Bull

AIX 5L Guide de l’utilisateur
Système d’exploitation et unités

AIX

REFERENCE
86 F2 44EM 01

Bull

AIX 5L Guide de l'utilisateur
Système d'exploitation et unités

AIX

Logiciel

Février 2005

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
86 F2 44EM 01

Imprimé en France

Vos suggestions sur la forme et le fond de ce manuel seront les bienvenues.
Une feuille destinée à recevoir vos remarques se trouve à la fin de ce document.

Pour commander d’autres exemplaires de ce manuel ou d’autres publications techniques Bull, veuillez utiliser le bon de commande également fourni en fin de manuel.

Marques déposées

Toutes les marques déposées sont la propriété de leurs titulaires respectifs.

AIXR est une marque déposée d’IBM Corp. et est utilisée sous licence.

UNIX est une marque déposée aux Etats–Unis et dans d’autres pays, licenciée exclusivement par Open Group.

Linux est une marque déposée de Linus Torvalds.

Préface

Ce manuel s'adresse aux utilisateurs novices qui souhaitent approfondir leur connaissance du système d'exploitation. Il traite les thèmes suivants : exécution des commandes, traitement des processus, gestion des fichiers et répertoires et impression. Il présente, en outre, des tâches telles que la sécurisation de fichiers, l'utilisation de support de stockage, la personnalisation des variables d'environnement (**.profile**, **.Xdefaults**, **.mwmrc**), et l'écriture de scripts shell. Les utilisateurs DOS y trouveront les procédures permettant d'exploiter les fichiers DOS dans cet environnement.

Conventions typographiques

Les conventions typographiques suivantes sont utilisées dans ce guide :

Gras	Commandes, mots-clés, fichiers, répertoires et autres éléments dont le nom est prédéfini par le système.
Italique	Permet d'identifier les paramètres dont les noms ou les valeurs doivent être indiqués par l'utilisateur.
`Espacement fixe`	Permet d'identifier les exemples de données spécifiques, les exemples de textes similaires aux textes affichés, les exemples de parties de code similaires au code que vous serez susceptibles de rédiger en tant que programmeur, les messages système ou les informations que vous devez saisir.

Distinction majuscules/minuscules dans AIX

La distinction majuscules/minuscules s'applique à toutes les données entrées dans le système d'exploitation AIX. Vous pouvez, par exemple, utiliser la commande **ls** pour afficher la liste des fichiers. Si vous entrez `LS`, le système affiche un message d'erreur indiquant que la commande entrée est introuvable. De la même manière, **FICHEA**, **FiChea** et **fichea** sont trois noms de fichiers distincts, même s'ils se trouvent dans le même répertoire. Pour éviter toute action indésirable, vérifiez systématiquement que vous utilisez la casse appropriée.

ISO 9000

Des systèmes de qualité homologuée ISO 9000 ont été utilisés lors du développement et de la fabrication de ce produit.

Bibliographie

Les manuels suivants contiennent des informations pertinentes :

- *AIX 5L Version 5.3 System User's Guide: Communications and Networks*
- *AIX 5L Version 5.3 System Management Guide: Operating System and Devices*
- *AIX 5L Version 5.3 System Management Concepts: Operating System and Devices*
- *AIX 5L Version 5.3 Imprimantes et impression – Guide de l'utilisateur*
- *AIX 5L Version 5.3 Commands Reference*
- *AIX 5L Version 5.3 Files Reference*

Table des matières

Chapitre 1. Noms de connexion, ID système et mots de passe

Le système d'exploitation ne peut vous fournir un environnement correct que s'il a le moyen de vous identifier. C'est pourquoi vous êtes invité, pour vous connecter, à entrer votre *nom de connexion* (également appelé ID utilisateur ou nom utilisateur) et un *mot de passe*. Les mots de passe constituent une première sécurité : personne ne peut accéder à votre système sans connaître à la fois votre nom de connexion et votre mot de passe.

Si le système est multi–utilisateur, chaque utilisateur autorisé se voit attribuer un compte, un mot de passe et un nom de connexion. Le système d'exploitation conserve une trace des ressources utilisées par chaque utilisateur. On appelle cela *compte système*. Chaque utilisateur est également doté d'un espace de stockage sur le système, appelé *système de fichiers*. Lorsque vous vous connectez, ce système de fichiers semble ne contenir que vos propres fichiers, même si des milliers d'autres fichiers se trouvent sur le système.

Vous pouvez, sur un même système, disposer de plusieurs noms de connexion. Pour passer de l'un à l'autre, il est inutile de vous déconnecter. Vous pouvez les utiliser simultanément dans des shells distincts ou successivement dans un même shell. En outre, si votre système fait partie d'un réseau connecté à d'autres systèmes, vous avez accès à tous les systèmes sur lesquels un nom de connexion vous a été attribué. On appelle cela une *connexion éloignée*.

A la fin de votre travail, déconnectez–vous : vous serez ainsi assuré que vos fichiers et vos données sont en sécurité.

Ce chapitre comporte les sections suivantes :

Connexion et déconnexion : généralités

Pour accéder au système d'exploitation, votre système doit être lancé et vous devez y être connecté. Lorsque vous ouvrez une session, vous êtes invité à décliner votre identité : le système peut alors configurer votre environnement.

Cette section décrit les procédures suivantes :

Connexion au système d'exploitation

Le système peut être configuré pour que vous ne puissiez y accéder qu'à certaines heures de la journée ou certains jours de la semaine. L'accès vous sera refusé si vous tentez une connexion en dehors des plages autorisées. Renseignez–vous auprès de votre administrateur système.

Vous vous connectez à l'affichage de l'invite de connexion. La connexion établie, vous êtes automatiquement placé dans votre répertoire personnel (également appelé *répertoire de connexion*).

Après avoir mis votre système sous tension, établissez une connexion pour démarrer une session.

1. Entrez votre nom de connexion après l'invite `login`: invite :

```
login : NomConnexion
```

   Par exemple, si votre nom utilisateur est denise :

```
login : denise
```

2. Si l'invite `password:` est affichée, entrez votre mot de passe.
   (Ce dernier ne s'affiche pas.)

```
PASSWORD : [votre mot de passe]
```

Si l'invite password ne s'affiche pas, c'est que vous n'avez pas de mot de passe défini. Vous pouvez commencer à travailler.

Si votre machine n'est pas mise sous tension, effectuez les étapes suivantes avant de vous connecter :

1. Mettez sous tension toutes les unités raccordées.
2. Mettez l'unité centrale sous tension (**I**).
3. Consultez l'afficheur. Si les autotests ont abouti, il doit être vide.

Si une erreur s'est produite, un code (de 3 chiffres) s'affiche et le système s'arrête. Veuillez contacter votre administrateur système pour plus d'informations sur les codes d'erreur et la reprise.

Les autotests terminés, l'invite de connexion s'affiche :

```
login:
```

Une fois la connexion établie, le système appelle, selon la configuration définie, une interface ligne de commande (shell) ou une interface graphique (par exemple, AIXwindows ou Common Desktop Environment (CDE)).

Si vous avez des questions concernant la configuration de votre mot de passe ou de votre nom d'utilisateur, veuillez vous adresser à votre administrateur système.

Connexions multiples (commande login)

Si vous travaillez sur plusieurs projets et souhaitez conserver des comptes séparés, vous pouvez ouvrir plusieurs sessions. Pour cela, spécifiez le même nom ou des noms de connexion différents.

> **Remarque :** A chaque système est associé un nombre maximal de noms de connexion pouvant être simultanément actifs. Ce nombre dépend de votre licence et varie selon les installations.

Par exemple, si vous vous connectez en tant que `denise1` et que votre deuxième nom de connexion est `denise2`, quand l'invite s'affiche, entrez :

```
login denise2
```

Si l'invite `password:` est affichée, entrez votre mot de passe. (Ce dernier ne s'affiche pas.) Vous disposez à présent de deux sessions.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **login** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Passage à un autre nom d'utilisateur (commande su)

Vous pouvez modifier l'ID utilisateur associé à une session (si vous connaissez le nom de connexion de l'utilisateur), en utilisant la commande **su** (switch user).

Si, par exemple, vous voulez devenir l'utilisateur `joyce`, entrez à l'invite :

```
su joyce
```

Si l'invite `password:` est affichée, entrez le mot de passe de l'utilisateur `joyce`. Votre ID utilisateur est désormais `joyce`. Si vous ne connaissez pas le mot de passe, l'accès vous est refusé.

Pour vérifier que votre ID utilisateur est `joyce`, utilisez la commande **id**. Pour plus d'informations sur la commande **id**, reportez–vous à la section Affichage de l'ID d'un utilisateur (commande id), page 1-6.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **su** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Suppression des messages de connexion

A l'ouverture d'une session, la commande **login** affiche la date du jour, le message du jour, la date et l'heure de la dernière tentative de connexion (aboutie ou non), ainsi que le nombre total de connexions inabouties, depuis la dernière mise à jour des données d'authentification (mot de passe, en général). Vous pouvez supprimer ces messages en ajoutant un fichier **.hushlogin** dans votre répertoire personnel.

A l'invite, dans votre répertoire utilisateur, tapez :

```
touch .hushlogin
```

La commande **touch** crée le fichier vide, **.hushlogin** s'il n'existe pas encore. A l'ouverture de session suivante, les messages de connexion seront éliminés. Vous pouvez aussi ne conserver que l'affichage de la date du jour à l'exclusion des autres messages.

Reportez–vous à la commande **touch** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Déconnexion du système d'exploitation (commandes exit et logout)

Pour déconnecter le système d'exploitation, effectuez l'une des opérations suivantes à l'invite du système :

Appuyez sur la combinaison de touches de contrôle fin de fichier (Ctrl–D).

OU

Entrez `exit`.

OU

Entrez `logout`.

Une fois la session fermée, le système affiche l'invite `login`.

Arrêt du système d'exploitation (commande shutdown)

Avertissement : Ne mettez pas le système hors tension avant d'avoir arrêté le système d'exploitation. En effet, vous interrompriez tous les process en cours. Si d'autres utilisateurs sont connectés ou que des travaux sont traités en arrière–plan, des données peuvent être perdues. Exécutez la procédure d'arrêt appropriée avant d'arrêter le système.

Si vous possédez les droits d'utilisateurs root, vous pouvez utiliser la commande **shutdown** pour arrêter le système. Si vous n'êtes pas autorisé à utiliser la commande **shutdown**, déconnectez–vous simplement du système d'exploitation et laissez–le en marche.

A l'invite, entrez ce qui suit :

```
shutdown
```

Lorsque la commande **shutdown** est exécutée, le système s'arrête avec le message :

```
....Shutdown completed....
```

Reportez–vous à la commande **shutdown** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Identification utilisateur et système

Cette section traite des commandes permettant d'afficher des informations identifiant les utilisateurs du système et le système utilisé.

Affichage du nom de connexion (commandes whoami et logname)

Si vous ouvrez plusieurs sessions, il n'est pas toujours évident de se souvenir des noms de connexion utilisés, notamment pour la session active.

Utilisation de la commande whoami

Pour afficher le nom de connexion utilisé, à l'invite, entrez ce qui suit :

```
whoami
```

Le système affiche un écran semblable à celui-ci :

```
denise
```

Ici, le nom de connexion est `denise`.

Reportez–vous à la commande **whoami** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Utilisation de la commande who am i

Une variante de la commande **who**, la commande **who am i**, permet d'afficher le nom de connexion, le nom du terminal et l'heure de la connexion. A l'invite, entrez ce qui suit :

```
who am i
```

Le système affiche un écran semblable à celui-ci :

```
denise  pts/0  Jun 21 07:53
```

Ici, le nom de connexion est `denise`, le nom du terminal est `pts/0` et l'utilisateur s'est connecté à 7:53, le 21 juin.

Reportez–vous à la commande **who**, dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Utilisation de la commande logname

Etant une autre variante de la commande **who** la commande **logname** affiche les mêmes informations que la commande **who**.

A l'invite, entrez ce qui suit :

```
logname
```

Le système affiche un écran semblable à celui-ci :

```
denise
```

Ici, le nom de connexion est `denise`.

Affichage du nom du système d'exploitation (commande uname)

Pour afficher le nom du système d'exploitation, utilisez la commande **uname**.

Par exemple, à l'invite, entrez :

```
uname
```

Le système affiche un écran semblable à celui-ci :

```
AIX
```

Ici, le nom du système d'exploitation est AIX.

Reportez–vous à la commande **uname** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Affichage du nom de votre système (commande uname)

Pour afficher le nom de votre système si vous êtes sur un réseau, utilisez la commande **uname** avec l'indicateur **–n**. Ce nom, qui identifie le système vis–à–vis du réseau, est distinct de votre ID de connexion.

Par exemple, à l'invite, entrez :

```
uname -n
```

Le système affiche un écran semblable à celui-ci :

```
barnard
```

Ici, le nom de connexion est `barnard`.

Reportez–vous à la commande **uname** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Affichage des utilisateurs connectés (commande who)

Pour afficher des informations sur tous les utilisateurs en cours sur le système local, utilisez la commande **who**. Les informations suivantes s'affichent : nom de connexion, nom système et date/heure de la connexion.

Remarque : Cette commande n'identifie que les utilisateurs se trouvant le nœud local.

Pour obtenir des informations sur les utilisateurs connectés au nœud local, entrez :

```
who
```

Le système affiche un écran semblable à celui-ci :

```
joe   lft/0 Jun 8 08:34
 denise   pts/1 Jun 8 07:07
```

Dans cet exemple, l'utilisateur `joe`, sur le terminal `lft/0`, s'est connecté à 8:34, le 8 juin.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **who** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage de l'ID d'un utilisateur (commande id)

Pour afficher les identificateurs système (ID) pour un utilisateur spécifié, utilisez la commande **id**. Les ID système sont des chiffres identifiant utilisateurs et groupes d'utilisateurs vis-à-vis du système. La commande **id** affiche les informations suivantes, le cas échéant :

- le nom et l'ID utilisateur réel ;
- le nom du groupe de l'utilisateur et l'ID groupe réel ;
- le nom et l'ID des groupes complémentaires, le cas échéant.

Par exemple, à l'invite, entrez :

```
id
```

Le système affiche un écran semblable à celui-ci :

```
uid=1544(sah) gid=300(build) euid=0(root) egid=9(printq)
groups=0(system),10(audit)
```

Dans cet exemple, l'utilisateur possède le nom `sah` avec un numéro ID de `1544`; un nom de groupe principal `build` avec un numéro ID de `300`; un nom d'utilisateur effectif `root` avec un numéro ID de `0`; un nom de groupe effectif `printq` avec un numéro ID de `9`; et deux noms de groupes supplémentaires `system` et `audit`, de numéros ID respectifs `0` et `10`.

Par exemple, à l'invite, entrez :

```
id denise
```

Le système affiche un écran semblable à celui-ci :

```
uid=2988(denise) gid=1(staff)
```

Dans cet exemple, l'utilisateur `denise` a un numéro ID de `2988` et seulement un nom de groupe principal, `staff` de numéro ID égal à `1`.

Reportez–vous à la commande **id** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Mots de passe

Votre système associe un mot de passe à chaque compte. Ce mot de passe unique offre une certaine sécurité système à vos fichiers. La sécurité est une partie importante des systèmes informatiques parce qu'elle évite que des personnes non autorisées n'accèdent au système et n'exploitent de manière frauduleuse les fichiers des autres utilisateurs. La sécurité permet également d'accorder des privilèges exclusifs à certains utilisateurs, en définissant les commandes qu'ils peuvent exécuter et les fichiers auxquels ils peuvent accéder. Par mesure de protection, certains administrateurs système ne permettent aux utilisateurs d'accéder qu'à certaines commandes ou fichiers.

Cette section décrit les procédures suivantes :

- Consignes relatives aux mots de passe, page 1-8
- Changement des mots de passe (commande passwd), page 1-9
- Paramétrage des mots de passe sur nul (commande passwd), page 1-9

Directives concernant les mots de passe

Vous devez disposer d'un mot de passe exclusif. *Les mots de passe ne doivent pas être partagés.* Protégez les mots de passe comme toute autre valeur de la société. Lors de la création de mots de passe, veillez à ce qu'ils soient difficiles à deviner, mais pas au point que vous deviez les écrire pour vous en souvenir.

L'utilisation de mots de passe obscurs garantit la sécurité de votre ID utilisateur. Les mots de passe reposant sur des informations personnelles, telles que votre nom ou votre date de naissance, sont de mauvais mots de passe. Les mots courants également peuvent être facilement devinés.

Les bons mots de passe comportent au moins six caractères et comprennent des caractères non alphabétiques. Les combinaisons de mots étranges et les mots sciemment mal orthographiés conviennent également.

> **Remarque :** Si votre mot de passe est si difficile à mémoriser que vous devez l'écrire, ce n'est pas un bon mot de passe.

Suivez les directives ci–après lors de la sélection d'un mot de passe :

- N'utilisez pas votre ID utilisateur comme mot de passe. Ne l'utilisez pas inversé, doublé ou modifié de toute autre façon.
- Ne réutilisez pas les mots de passe. Le système peut être configuré pour refuser la réutilisation d'un mot de passe.
- N'utilisez pas le nom d'une autre personne comme mot de passe.
- N'utilisez pas comme mot de passe des termes figurant dans le dictionnaire de vérification orthographique en ligne.
- N'utilisez pas de mots de passe inférieurs à six caractères.
- N'utilisez pas de mots obscènes ; ce sont les premiers recherchés lorsque l'on tente de deviner les mots de passe.
- Utilisez un mot de passe facile à mémoriser, de manière à ne pas être obligé de l'écrire.
- Utilisez des mots de passe dans lesquels lettres et chiffres sont combinés, et comportant des majuscules et des minuscules.
- Utilisez deux mots, séparés par un chiffre, comme mot de passe.
- Vous pouvez utiliser des mots de passe prononçables. Ils sont plus faciles à mémoriser.
- N'écrivez pas les mots de passe. Toutefois, si vous devez les écrire, placez–les dans un lieu protégé, comme une armoire fermant à clé.

Modification du mot de passe (commande passwd)

Pour modifier votre mot de passé, utilisez la commande **passwd**.

1. A l'invite, entrez ce qui suit :

```
passwd
```

   Si vous ne possédez pas de mot de passe, ignorez l'étape 2.

2. Le message suivant s'affiche :

```
Modification du mot de passe pour l'ID utilisateur ancien mot de passe
de IDutilisateur :
```

   Cette demande évite qu'un utilisateur non autorisé ne modifie votre mot de passe lorsque vous n'êtes pas à proximité du système. Tapez votre mot de passe actuel et appuyez sur Entrée.

3. Le message suivant s'affiche :

```
Nouveau mot de passe
```

   Tapez le nouveau mot de passe que vous souhaitez définir et appuyez sur Entrée.

4. L'invite ci–après est affichée et vous demande de confirmer le nouveau mot de passe.

```
Enter the new password again:
```

   Cette demande vous évite de définir comme mot de passe une chaîne de caractères mal saisie que vous ne seriez plus en mesure de reproduire.

Reportez–vous à la commande **passwd** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Annulation du mot de passe (commande passwd)

Si vous ne voulez pas saisir votre mot de passe lors de chaque connexion, il suffit de lui attribuer la valeur NULL (vierge).

Pour ce faire, saisissez :

```
passwd
```

Lorsque vous êtes invité à saisir le nouveau mot de passe, appuyez sur Entrée ou sur Ctrl–D.

La commande **passwd** ne vous invite pas à nouveau à saisir un mot de passe. Un message de vérification de l'annulation du mot de passe s'affiche.

Reportez–vous à la commande **passwd** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour obtenir de plus amples informations ainsi que la syntaxe exacte.

Récapitulatif des commandes relatives aux noms de connexion, aux ID système et aux mots de passe

Commandes de connexion et de déconnexion

login	Ouvre la session.
logout	Arrête tous les process.
shutdown	Met fin aux opérations du système
su	Modifie l'ID utilisateur associé à une session
touch	Met à jour l'heure d'accès et de modification d'un fichier, ou crée un fichier vide

Commandes d'identification utilisateur et système

id	Affiche les identifications système d'un utilisateur
logname	Affiche le nom de connexion.
uname	Affiche le nom du système d'exploitation
who	Identifie les utilisateurs actuellement connectés
whoami	Affiche votre nom de connexion

Commande de mot de passe

passwd	Modifie le mot de passe de l'utilisateur

Chapitre 2. Système et environnement utilisateur

Chaque nom de connexion est associé à un environnement système. Il s'agit d'une zone où sont enregistrées les informations communes à tous les processus en cours dans une session. Vous pouvez utiliser plusieurs commandes pour afficher des informations sur votre système.

Ce chapitre décrit les procédures suivantes relatives à l'affichage des informations relatives à votre environnement :

Affichage de la liste des unités système (commande lscfg)

La commande **lscfg** affiche le nom, la position et la description de chaque unité de la configuration courante. La liste est triée en fonction de l'emplacement des unités.

Par exemple, pour répertorier les unités configurées dans votre système, saisissez à l'invite de commande :

```
lscfg
```

Le système affiche une sortie semblable à l'exemple suivant :

```
LISTE DES RESSOURCES INSTALLEES

Les ressources suivantes sont installées sur la machine.

+/- = Ajoutée/Supprimée de la Liste des ressources testées.
*  = NON prise en charge par le programme de diagnostics.


   Architecture du modèle : chrp
   Mise en oeuvre du modèle : Processeur multiple, bus PCI

 +  sysplanar0   00-00           CPU Planar
 +  fpa0         00-00         Processeur calcul virgule flottante
 +  mem0         00-0A        Carte mémoire
 +  mem1         00-0B         Carte mémoire
 +  ioplanar0    00-00         Carte d'E/S
 +  rs2320       00-01         Carte RS232
 +  tty0         00-01-0-01    Port carte RS232
 -  tty1         00-01-0-02    Port carte RS232
    ..
    ..
    ..
```

La liste des unités n'est pas seulement triée en fonction de l'emplacement des unités. L'ordre fait également intervenir la notion de hiérarchie parent/enfant. Si le parent a plusieurs enfants, ces derniers sont triés en fonction de l'emplacement de l'unité. Si les enfants ont le même emplacement d'unité, ils sont affichés dans l'ordre dans lequel ils ont été obtenus par les logiciels. Pour afficher des informations sur une unité particulière, vous disposez de l'indicateur **–l**. Ainsi, pour afficher les informations sur l'unité **sysplanar0**, entrez ce qui suit à l'invite :

```
lscfg -l sysplanar0
```

Le système affiche une sortie semblable à l'exemple suivant :

```
DEVICE          LOCATION     DESCRIPTION

 sysplanar0   00-00            CPU Planar
```

Lancez également la commande **lscfg** pour afficher les données techniques essentielles (VPD), telles que références des pièces, numéros de série et niveaux d'EC. Dans le cas de certaines unités, les données techniques essentielles sont collectées automatiquement, puis sont ajoutées à la configuration du système. Pour d'autres unités, ces données sont saisies manuellement. Les données sont alors précédées de l'abréviation `ME` (saisie).

Par exemple, pour répertorier les données techniques essentielles (VPD) des unités configurées dans votre système, saisissez à l'invite de commande :

```
lscfg -v
```

Le système affiche une sortie semblable à l'exemple suivant :

```
LISTE DES RESSOURCES INSTALLEES AVEC VPD

Les ressources suivantes sont installées sur la machine.

  Architecture du modèle : chrp
   Mise en oeuvre du modèle : Processeur multiple, bus PCI
```

```
sysplanar0   00-00   CPU Planar

   Part Number.........342522
    Niveau modif technique.....254921
    Numéro de série............353535

fpa0   00-00   Floating Point Processor
mem0   00-0A   Carte mémoire

   EC Level............990221
.
.
.
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **lscfg** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage du nom de votre console (commande lscons)

La commande **lscons** envoie le nom de l'unité de console en cours vers la sortie standard (généralement votre écran).

Par exemple, à l'invite, entrez :

```
lscons
```

Le système affiche une sortie semblable à l'exemple suivant :

```
/dev/lft0
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **lscons** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage du nom de votre terminal (commande tty)

Pour afficher le nom de votre terminal, utilisez la commande **tty**.

Le système affiche des informations similaires à l'exemple suivant :

```
tty
```

Le système affiche des informations similaires à l'exemple suivant :

```
/dev/tty06
```

Dans cet exemple, `tty06` est le nom du terminal et **/dev/tty06** est le fichier du périphérique qui contient l'interface avec ce terminal.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **tty** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Liste des écrans disponibles (commande lsdisp)

La commande **lsdisp** permet de répertorier les écrans en cours disponibles sur le système et fournit le nom d'identification, le numéro d'emplacement, le nom de l'écran et la description des différents écrans.

Par exemple, pour répertorier tous les écrans disponibles, entrez :

```
lsdisp
```

La sortie est semblable à l'exemple ci–après. La liste s'affiche dans l'ordre croissant des numéros d'emplacement.

```
Name    Slot     Name         Description
ppr0    00-01    POWER_G4     Midrange Graphics Adapter
gda0    00-03    colorgda     Color Graphics Display Adapter
ppr1    00-04    POWER_Gt3    Midrange Entry Graphics Adapter
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **lsdisp** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Liste des polices disponibles (commande lsfont)

La commande **lsfont** permet d'afficher la liste des polices disponibles pour l'affichage.

Par exemple, pour répertorier toutes les polices disponibles, entrez :

```
lsfont
```

La sortie est semblable à l'exemple ci–après. L'identificateur de police, le nom du fichier, la taille de glyphe et le codage de la police sont indiqués :

```
FONT  FILE            GLYPH  FONT
ID    NAME            SIZE   ENCODING
====  ==============  =====  =========
0     Erg22.iso1.snf  12x30  ISO8859-1
1     Erg11.iso1.snf   8x15  ISO8859-1
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **lsfont** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Liste des mappes de clavier programmable en cours chargées sur le système (commande lskbd)

La commande **lskbd** permet d'afficher le nom du chemin d'accès absolu de la mappe de clavier programmable en cours chargée sur le système.

Par exemple, pour répertorier les mappes de clavier en cours, saisissez la commande :

```
lskbd
```

La liste affichée par la commande **lskbd** est similaire à l'exemple suivant :

```
The current software keyboard map = /usr/lib/nls/loc/C.lftkeymap
```

Liste des logiciels disponibles (commande lslpp)

La commande **lslpp** affiche des informations relatives aux logiciels disponibles sur le système.

Par exemple, pour répertorier tous les logiciels du système, entrez :

```
lslpp -l -a
```

Voici un exemple de sortie :

```
Fileset                 Level  State     Description
--------------------  -------  --------  -----------------
Path: /usr/lib/objrepos
   X11_3d.gl.dev.obj             APPLIED   AIXwindows/3D GL
                                           Development Utilities
Fonts
  X11fnt.oldX.fnt               APPLIED   AIXwindows Miscellaneous
                                           X Fonts
X11mEn_US.msg                   APPLIED   AIXwindows NL Message
                                           files

.

.

.
```

Si la liste est très longue, la première partie peut défiler hors de l'écran. Pour afficher la liste une page (écran) à la fois, associez la commande **lslpp** à la commande **pg**. A l'invite, entrez ce qui suit :

```
lslpp | pg
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **lslpp** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Liste des affectations de touches du terminal (commande stty)

Pour afficher les paramètres de votre terminal, utilisez la commande **stty**. Cela vous permet ainsi de savoir quelles touches le terminal utilise comme touches de contrôle.

Par exemple, à l'invite, entrez :

```
stty –a
```

Le système affiche des informations similaires à l'exemple suivant :

```
.
.
.
intr = ^C; quit = ^ erase = ^H; kill = ^U; eof = ^D;
eol = ^@ start = ^Q; stop = ^S; susp = ^Z; dsusp = ^Y;
reprint = ^R discard = ^O; werase = ^W; lnext = ^V
.
.
.
```

Dans cet exemple, les lignes `intr = ^C; quit = ^\; erase = ^H;` représentent les paramètres des touches de contrôle. La touche `^H` est la touche Retour arrière, qui exécute la fonction de suppression.

Si la liste est très longue, la première partie peut défiler hors de l'écran. Pour afficher la liste une page (écran) à la fois, associez la commande **stty** à la commande **pg**. A l'invite, entrez ce qui suit :

```
stty –a | pg
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **stty** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Liste de toutes les variables d'environnement (commande env)

Pour afficher les variables d'environnement courantes, utilisez la commande **env**.
Une variable accessible à tous les process est dite *variable globale.*

Par exemple, pour afficher la liste de toutes les variables d'environnement et leurs valeurs associées, entrez :

```
env
```

Voici un exemple de sortie :

```
TMPDIR=/usr/tmp
myid=denise
LANG=En_US
UNAME=barnard
PAGER=/bin/pg
VISUAL=vi

PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/
u/bin1
MAILPATH=/usr/mail/denise?denise has mail !!!
MAILRECORD=/u/denise/.Outmail
EXINIT=set beautify noflash nomesg report=1 showmode showmatch
EDITOR=vi
PSCH=>
HISTFILE=/u/denise/.history
LOGNAME=denise
MAIL=/usr/mail/denise
PS1=denise@barnard:${PWD}>
PS3=#
PS2=>
epath=/usr/bin
USER=denise
SHELL=/bin/ksh
HISTSIZE=500
HOME=/u/denise
FCEDIT=vi
TERM=lft
MAILMSG=**YOU HAVE NEW MAIL. USE THE mail COMMAND TO SEE YOUR
PWD= /u/denise
ENV=/u/denise/.env
```

Si la liste est très longue, la première partie défile hors de l'écran. Pour afficher la liste une page (écran) à la fois, associez la commande **env** à la commande **pg**. A l'invite, entrez ce qui suit :

```
env | pg
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **env** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage de la valeur d'une variable d'environnement (commande printenv)

Pour afficher les variables d'environnement courants, utilisez la commande **printenv**. Si vous définissez le paramètre *Nom*, le système affiche uniquement la valeur de la variable concernée. Si vous ne définissez pas le paramètre *Nom*, la commande **printenv** affiche toutes les variables d'environnement courantes, présentant une séquence *Nom* **=** *Valeur* par ligne.

Ainsi, pour afficher la valeur de la variable d'environnement **MAILMSG**, entrez :

```
printenv MAILMSG
```

La commande renvoie la valeur de la variable **MAILMSG**. Par exemple :

```
YOU HAVE NEW MAIL
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **printenv** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Utilisation des langues bidirectionnelles (commande aixterm)

La commande **aixterm** prend en charge les langues bidirectionnelles que sont l'arabe et l'hébreu. Les langues bidirectionnelles peuvent être lues et écrites dans les deux sens : de gauche à droite et de droite à gauche. Vous pouvez travailler avec des applications en arabe ou en hébreu en ouvrant une fenêtre dans laquelle vous spécifiez un environnement local arabe ou hébreu.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **aixterm** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Récapitulatif des commandes relatives à l'environnement utilisateur et aux informations système

aixterm	Permet de travailler dans des langues bidirectionnelles.
env	Affiche l'environnement courant ou définit l'environnement d'exécution d'une commande.
lscfg	Affiche des informations de diagnostic sur une unité.
lscons	Affiche le nom de la console courante.
lsdisp	Liste les écrans disponibles sur le système.
lsfont	Liste les polices disponibles sur l'écran.
lskbd	Liste les mappes clavier chargées sur le système.
lslpp	Liste les logiciels disponibles.
printenv	Affiche la valeur des variables d'environnement.
stty	Affiche les paramètres du système.
tty	Affiche le chemin d'accès complet à votre terminal.

Chapitre 3. L'environnement Common Desktop

Avec l'environnement CDE AIX, vous avez accès aux unités et aux outils en réseau sans vous soucier de leur emplacement. Vous pouvez échanger des données entre applications tout simplement en faisant glisser et en déplaçant des objets. Nombre de commandes complexes qui nécessitaient une syntaxe de ligne de commande complexe sont à présent aisément exécutables et sont en outre identiques d'une plate–forme à l'autre. Vous pouvez par exemple configurer de manière centralisée des applications et les distribuer aux utilisateurs. Vous pouvez également centraliser la gestion de la sécurité, de la disponibilité et des échanges d'informations entre les applications.

Remarque : Les manuels d'aide Common Desktop Environment (CDE) 1.0, la documentation basée sur le Web et les manuels imprimés peuvent désigner le Desktop comme l'environnement Common Desktop, le bureau AIXwindows, le CDE 1.0 ou encore le Desktop.

Les sujets traités dans ce chapitre sont les suivants :

Démarrage et arrêt de l'environnement Common Desktop

Vous pouvez configurer le lancement automatique de l'environnement Common Desktop au démarrage du système ou le lancer manuellement. Pour exécuter ces tâches, vous devez être utilisateur racine.

Activation/désactivation du démarrage automatique

Vous pouvez configurer le lancement automatique de l'environnement Common Desktop au démarrage du système. Vous pouvez utiliser Web–based System Manager (entrez `wsm` et sélectionnez `System`) via SMIT (System Management Interface Tool) ou la ligne de commande.

Préalables

Vous devez être utilisateur racine pour activer ou désactiver le démarrage automatique.

Démarrage et arrêt automatique de l'environnement Common Desktop

Tâche	Raccourci SMIT	Commande ou fichier
Activation du démarrage automatique [1]	**smit dtconfig**	**/usr/dt/bin/dtconfig –e**
Désactivation du démarrage automatique [1]	**smit dtconfig**	**/usr/dt/bin/dtconfig –d**

[1] **Remarque :** après exécution de la tâche, redémarrez le système.

Démarrage manuel de l'environnement Common Desktop

Pour démarrer manuellement l'environnement Common Desktop :

1. Connectez-vous en tant qu'utilisateur racine.
2. Sur une ligne de commande, entrez :

```
/usr/dt/bin/dtlogin –daemon
```

L'écran **Desktop Login** s'affiche. Une session Desktop débute lorsque vous vous connectez.

Arrêt manuel de l'environnement Common Desktop

Quand vous arrêtez manuellement le gestionnaire de connexion, tous les serveurs X et les sessions Desktop démarrés par le gestionnaire de connexion sont arrêtés.

Pour arrêter manuellement l'environnement Common Desktop :

1. Ouvrez une fenêtre d'émulation de terminal et connectez–vous en tant qu'utilisateur racine.
2. Accédez à l'ID processus du gestionnaire de connexion en tapant :

```
cat /var/dt/Xpid
```

3. Arrêtez le gestionnaire de connexion en tapant :

```
kill –term   process_id
```

Modification des profils du Desktop

Quand vous vous connectez au Desktop, la lecture du fichier d'environnement shell (**.profile** ou **.login**) n'est pas automatique. Le Desktop exécute le X Server avant votre connexion, ainsi la fonction fournie par le fichier **.profile** ou le fichier **.login** doit être fournie par le gestionnaire de connexion du Desktop.

Les variables d'environnement spécifiques de l'utilisateur sont définies dans */Home Directory/***.dtprofile**. Un modèle de ce fichier se trouve dans **/usr/dt/config/sys.dtprofile**. Placez les variables et les commandes shell dans le **.dtprofile** applicable au seul Desktop. Ajoutez les lignes à la fin de **.dtprofile** pour incorporer le fichier d'environnement shell.

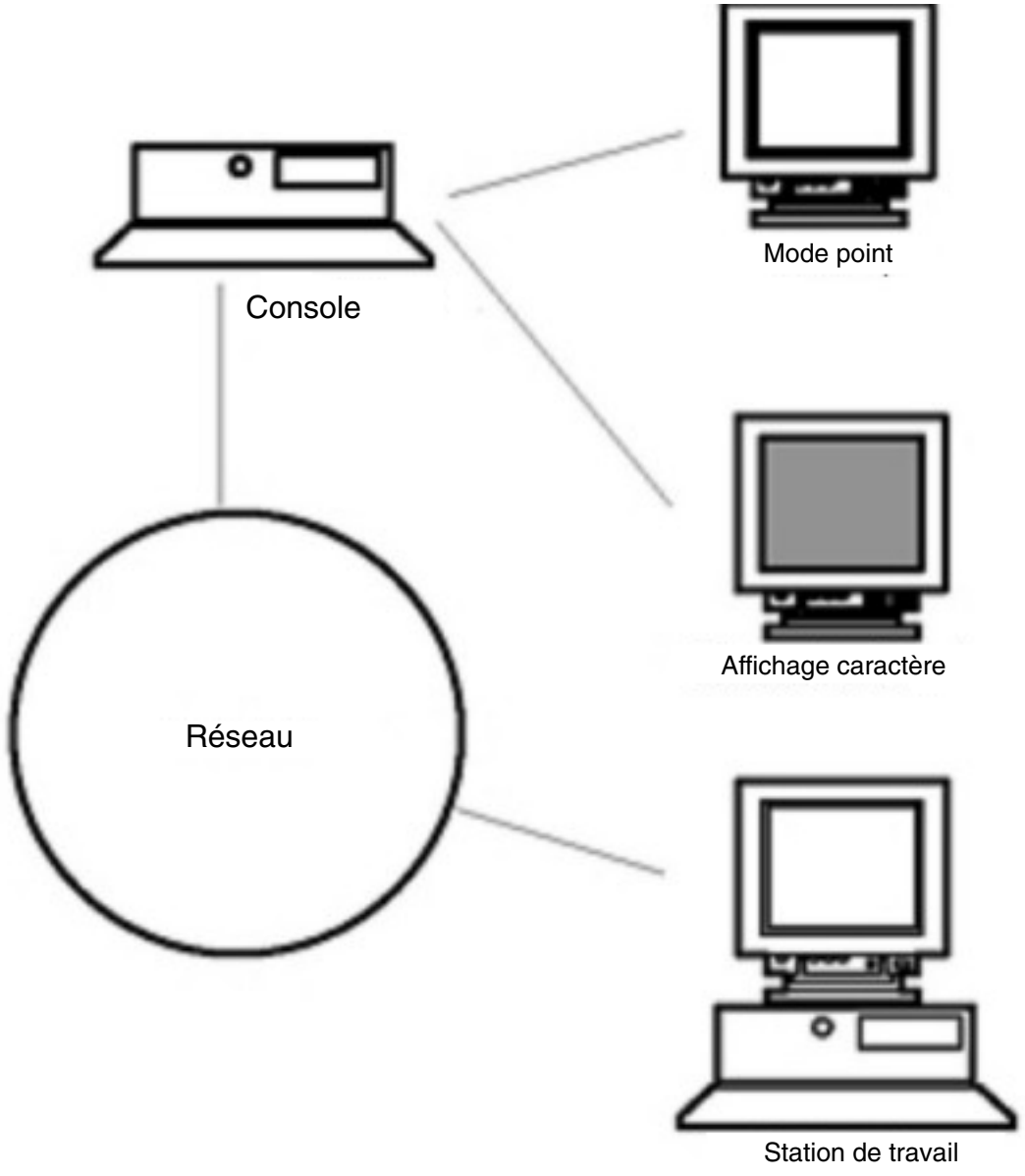
Les variables d'environnement à l'échelle du système peuvent être définies dans les fichiers de configuration du gestionnaire de connexion. Pour plus d'informations sur la configuration des variables d'environnement, reportez–vous à *Common Desktop Environment 1.0 : Advanced User's and System Administrator's Guide*.

Ajout/suppression d'écrans et de terminaux dans l'environnement Common Desktop

Le gestionnaire de connexion peut être lancé depuis un système doté d'une console locale graphique ou de type bitmap (mode point). Il existe nombre de méthodes de démarrage (voir la figure suivante). Vous pouvez démarrer l'environnement Common Desktop depuis les :

- Consoles locales
- Consoles distantes
- Systèmes de terminaux X d'affichage caractère et d'affichage en mode point fonctionnant sur un système hôte du réseau

Figure 1. Points d'interface CDE Cette illustration montre les points de liaison entre une console, un réseau, un affichage en mode point, un affichage caractère et une station de travail.



Un terminal X comprend une unité d'affichage, un clavier et une souris qui ne s'exécute que sur le serveur X. Les clients, notamment l'environnement Common Desktop, s'exécutent sur un ou plusieurs systèmes hôte des réseaux. Les sorties client sont dirigées sur le terminal X.

Pour les tâches suivantes de configuration du gestionnaire de connexion, plusieurs configurations sont admises :

- Suppression d'un écran local, page 3-5
- Ajout d'un terminal ASCII ou caractères, page 3-5

Utilisation d'une station comme terminal X

Sur la ligne de commande, entrez :

```
/usr/bin/X11/X –query hostname
```

Le serveur X de la station que vous souhaitez exploiter comme un terminal X doit :

- Prendre en charge XDMCP et l'option ligne de commande **–query**,
- Fournir à l'hôte du terminal l'autorisation hôte X (dans **/etc/X*.hosts**).

Suppression d'un écran local

Pour supprimer un écran local, supprimez l'enregistrement correspondant dans le fichier **Xservers** du répertoire */usr/dt/config*.

Ajout d'un terminal ASCII ou caractères

Un *terminal caractères* ou *terminal ASCII* est une configuration dans laquelle le terminal n'est pas une unité bitmap (mode point).

Ajout d'une console ASCII ou caractères sans affichage bitmap

1. Si le fichier **/etc/dt/config/Xservers** est inexistant, copiez le fichier **/usr/dt/config/Xservers** dans le répertoire **/etc/dt/config**.
2. Si vous devez copier le fichier **Xservers** dans **/etc/dt/config**, modifiez ou ajoutez la ligne **Dtlogin.servers:** de **/etc/dt/config/Xconfig** comme suit :

```
Dtlogin*servers: /etc/dt/config/Xservers
```

3. Modifiez dans **/etc/dt/config/Xservers** la ligne qui démarre le serveur X. Cette opération désactive le menu des options de connexion.

```
# * Local local@console /path/X :0
```

4. Lisez les fichiers de configuration du gestionnaire de connexion.

Ajout d'une console caractères avec affichage bitmap

1. Si le fichier **/etc/dt/config/Xservers** est inexistant, copiez le fichier **/usr/dt/config/Xservers** dans le répertoire **/etc/dt/config**.
2. Si vous devez copier le fichier **Xservers** dans **/etc/dt/config**, modifiez ou ajoutez la ligne **Dtlogin.servers:** de **/etc/dt/config/Xconfig** comme suit :

```
Dtlogin*servers: /etc/dt/config/Xservers
```

3. Modifiez dans **/etc/dt/config/Xservers** la ligne qui démarre le serveur X comme suit :

```
* Local local@none /path/X :0
```

4. Lisez les fichiers de configuration du gestionnaire de connexion.

Personnalisation des écrans Common Desktop Environment

Vous pouvez configurer le gestionnaire de connexion Common Desktop Environment pour l'exploiter sur des systèmes dotés de deux écrans ou plus.

Sur un système comportant plusieurs écrans :

- Un serveur doit être démarré sur chacun d'eux.
- Le mode Windows ne doit pas être configuré.

Pour chaque écran, il peut s'avérer nécessaire d'utiliser des ressources dtlogin distinctes.

Pour chaque unité d'affichage, il peut également s'avérer nécessaire d'utiliser des variables d'environnement distinctes à l'échelle du système.

Démarrage du serveur sur chaque écran

1. Si le fichier **/etc/dt/config/Xservers** est inexistant, copiez le fichier **/usr/dt/config/Xservers** dans le répertoire **/etc/dt/config**.
2. Si vous devez copier **Xservers** dans **/etc/dt/config**, modifiez la ligne **Dtlogin.servers:** de **/etc/dt/config/Xconfig** comme suit :

```
Dtlogin*servers: /etc/dt/config/Xservers
```

3. Modifez **/etc/dt/config/Xservers** pour démarrer un serveur X sur chaque terminal.

Syntaxe

La syntaxe de démarrage de serveur est la suivante :

```
NomEcran ClasseEcran TypeEcran [ @ite ] Commande
```

Les écrans connectés à un émulateur ITE (Internal Terminal Emulator) sont les seuls écrans exploitables en *mode No Windows*. Ce mode désactive temporairement le Desktop sur l'écran et exécute un process *getty* si un tel process n'est pas déjà en cours. Un process getty est un programme UNIX® qui définit le type de terminal et qui est utilisé dans le process de connexion.

Ainsi, vous pouvez vous connecter et exécuter des tâches qu'il est normalement impossible d'effectuer sous l'environnement CDE. Quand vous vous déconnectez, le Desktop est redémarré sur l'écran. Si le processus getty n'est pas déjà en cours sur un terminal, le gestionnaire de connexion en lance un au démarrage du mode No Windows.

Configuration par défaut

Si ITE n'est pas défini, display : 0 lui est associé (/dev/console).

Définition d'un écran différent d'ITE

- Sur l'écran ITE, donnez à ITE la valeur character device.
- Sur tous les autres écrans, donnez à ITE la valeur none.

Exemples

Dans **Xservers**, les entrées suivantes lancent un serveur sur trois écrans locaux sur `sysaaa:0`. L'écran `:0` correspond à la console (ITE).

```
sysaaa:0 Local local /usr/bin/X11/X :0
sysaaa:1 Local local /usr/bin/X11/X :1
sysaaa:2 Local local /usr/bin/X11/X :2
```

Sur l'hôte sysbbb, l'écran en mode point `:0` n'est pas un écran ITE ; l'ITE est associé à **/dev/ttyi1**. Dans **Xservers**, les entrées suivantes lancent des serveurs sur les deux écrans en mode point avec le mode No Windows activé sur `:1`.

```
sysaaa:0 Local local@none /usr/bin/X11/X :0
sysaaa:1 Local local@ttyi1 /usr/bin/X11/X :1
```

Définition d'un nom d'écran dans Xconfig

Pour spécifier le nom de l'écran dans Xconfig :

- Remplacez les deux points (:) par un tiret de soulignement (_).
- Pour un nom d'hôte entièrement qualifié, remplacez les points par des caractères soulignés.

Exemple

```
Dtlogin.claaa_0.resource: value
Dtlogin.sysaaa_prsm_ld_edu_0.resource: value
```

Utilisation d'un gestionnaire de connexion distinct pour chaque écran

Pour utiliser un gestionnaire de connexion distinct pour chaque écran :

1. Si le fichier **/etc/dt/config/Xconfig** est inexistant, copiez le fichier **/usr/dt/config/Xconfig** dans le **répertoire /etc/dt/config**.
2. Utilisez les ressources qui se trouvent dans **/etc/dt/config/Xconfig** pour spécifier un autre fichier de ressources pour chaque affichage :

```
Dtlogin.DisplayName.resources:   chemin  /   fichier
```

   où *chemin* désigne le nom du chemin d'accès aux fichiers Xresource à utiliser et *fichier*, le nom de fichier des fichiers Xresource à utiliser.

3. Créez chaque fichier de ressource spécifié dans le fichier **Xconfig**. Un fichier Xresources relatif à la langue utilisée est installé dans **/usr/dt/config/<LANG>**.
4. Placez les ressources dtlogin pour cet écran dans chaque fichier.

Exemple

Dans **Xconfig**, les lignes suivantes spécifient un fichier de ressource différent par écran :

```
Dtlogin.sysaaa_0.resources: /etc/dt/config/Xresources0
Dtlogin.sysaaa_1.resources: /etc/dt/config/Xresources1
Dtlogin.sysaaa_2.resources: /etc/dt/config/Xresources2
```

Script distinct pour chaque écran

1. Si le fichier **/etc/dt/config/Xconfig** est inexistant, copiez le fichier **/usr/dt/config/Xconfig** dans le répertoire **/etc/dt/config**.
2. Utilisez les ressources startup, reset et setup dans **/etc/dt/config/Xconfig** pour spécifier un script différent par écran (qui seront exécutés à la place des fichiers **Xstartup**, **Xreset** et **Xsetup**) :

```
Dtlogin*DisplayName*startup: /   chemin  /   fichier
Dtlogin*DisplayName*reset: /   chemin  /   fichier
Dtlogin*DisplayName*setup: /   chemin  /   fichier
```

où *chemin* désigne le nom du chemin d'accès du fichier à utiliser et *fichier*, le nom de fichier du fichier à utiliser. Le script de démarrage est défini comme racine une fois que l'utilisateur s'est connecté et avant le lancement de la session Common Desktop Environment.

Le script `/usr/dt/config/Xreset` peut être utilisé pour inverser la définition du fichier **Xstartup**. Le fichier **Xreset** est exécuté quand l'utilisateur se déconnecte.

Exemple

Dans le fichier **Xconfig**, les lignes suivantes spécifient différents scripts pour les deux écrans.

```
Dtlogin.sysaaa_0*startup:   /etc/dt/config/Xstartup0
Dtlogin.sysaaa_1*startup:    /etc/dt/config/Xstartup1
Dtlogin.sysaaa_0*setup:      /etc/dt/config/Xsetup0
Dtlogin.sysaaa_1*setup:      /etc/dt/config/Xsetup1
Dtlogin.sysaaa_0*reset:      /etc/dt/config/Xreset0
Dtlogin.sysaaa_1*reset:      /etc/dt/config/Xreset1
```

Utilisation de variables d'environnement, propres à l'ensemble du système, distinctes pour chaque écran

Pour définir des variables d'environnement, propres à l'ensemble du système, distinctes pour chaque écran :

1. Si le fichier **/etc/dt/config/Xconfig** est inexistant, copiez le fichier **/usr/dt/config/Xconfig** dans le répertoire **/etc/dt/config**.
2. Donnez une définition distincte pour chaque écran dans la ressource environnement de **/etc/dt/config/Xconfig** :

```
Dtlogin*DisplayName*environment: value
```

Les règles suivantes s'appliquent aux variables d'environnement pour chaque écran :

- Séparez les affectations de variables par un espace ou une tabulation.
- Ne vous servez pas de la ressource d'environnement pour définir TZ et LANG.
- Dans **Xconfig**, le traitement de shell n'existe pas.

Exemple

Dans **Xconfig**, les lignes suivantes définissent des variables différentes pour deux écrans.

```
Dtlogin*syshere_0*environment:EDITOR=vi SB_DISPLAY_ADDR=0xB00000
Dtlogin*syshere_1*environment: EDITOR=emacs \
         SB_DISPLAY_ADDR=0xB00000
```

Chapitre 4. Commandes et processus

Une *commande* est une demande d'exécution d'une opération ou d'un programme. L'utilisation des commandes permet d'indiquer au système d'exploitation les tâches à effectuer. Les commandes entrées sont déchiffrées par un interpréteur de commandes (également appelé *shell*) et ces tâches sont traitées.

Un programme ou une commande en cours d'exécution est appelé un *process*. Le système peut exécuter simultanément plusieurs process.

Le système d'exploitation permet de gérer les entrées et les sorties (E/S) par le biais de commandes et de symboles spécifiques. Vous pouvez contrôler les entrées en indiquant où lire les données. Vous pouvez ainsi définir l'origine des entrées : clavier (entrée standard) ou fichier. Vous pouvez aussi contrôler la sortie en spécifiant l'emplacement d'affichage ou de stockage des données : par exemple, l'écran (sortie standard) ou un fichier.

Cette section décrit les points suivants :

Présentation des commandes

Certaines commandes peuvent être entrées en tapant simplement un mot. Vous pouvez également combiner des commandes pour que la sortie d'une commande devienne l'entrée d'une autre commande. On appelle cela *traitement pipeline*. Pour plus d'informations sur le traitement pipeline, reportez–vous à Fonctions du Shell, page 12-3.

Les indicateurs définissent plus précisément les actions des commandes. L' *indicateur* est un modificateur utilisé avec le nom de commande sur la ligne de commande, généralement précédé d'un tiret.

Les commandes peuvent être regroupées dans un fichier. Ces fichiers sont appelés *procédures shell* ou *scripts shell*. Au lieu d'exécuter les commandes une par une, vous exécutez le fichier qui contient ces commandes. Pour plus d'informations sur les scripts et procédures, reportez–vous à la section Création et exécution d'un script shell, page 12-7.

Pour entrer une commande, à l'invite, entrez le nom de la commande et appuyez sur Entrée.

```
$ NomCommande
```

Cette section décrit les procédures suivantes :

- Syntaxe des commandes, page 4-3
- Lecture des lignes de syntaxe, page 4-5
- Démarrage de Web–based System Manager, page 4-6
- Exécution d'autres commandes avec la commande smit, page 4-6
- Utilisation de la commande smit, page 4-6
- Localisation d'une commande ou d'un programme (commande whereis), page 4-6
- Affichage des informations sur une commande (commande man), page 4-6
- Affichage de la fonction d'une commande (commande whatis), page 4-7
- Liste des commandes précédemment entrées (commande history du Shell), page 4-7
- Répétition des commandes à l'aide de la commande history du Shell, page 4-8
- Substitution des chaînes à l'aide de la commande history du Shell, page 4-9
- Edition de la commande history, page 4-9
- Création d'une commande alias (commande alias du Shell), page 4-10
- Travail avec les commandes de formatage de texte, page 4-10

Syntaxe des commandes

Bien que certaines commandes puissent être entrées en tapant simplement un mot, d'autres utilisent des indicateurs et des paramètres. Chaque commande possède une syntaxe désignant les indicateurs et les paramètres requis et en option. Voici le format général d'une commande :

```
NomCommande   indicateur(s) paramètre(s)
```

Voici quelques règles générales concernant les commandes :

- Les espaces entre les commandes, les indicateurs et les paramètres sont significatifs.
- Deux commandes peuvent être entrées sur la même ligne en les séparant par un point–virgule (;). Par exemple :

```
$   Commande1 ;   Commande2
```

  Le shell exécute les commandes de manière séquentielle.

- Les commandes respectent la distinction minuscules–majuscules. Le shell fait la distinction entre les lettres majuscules et les lettres minuscules. Pour le shell, `print` est différent de `PRINT` ou `Print`.
- Une commande très longue peut être entrée sur plusieurs lignes à l'aide du caractère barre oblique inversée (\). Pour le shell, une barre oblique inversée signifie une suite de ligne. L'exemple suivant est une commande qui s'étend sur deux lignes :

```
$ ls Mail info temp \
(appuyez sur Entrée)

> diary
(l'invite > s'affiche)
```

  Le caractère > est l'invite secondaire ($ étant l'invite principale par défaut de l'utilisateur non root), indiquant que la ligne courante est la suite de la précédente. Veuillez noter que **csh** (C shell) ne génère pas d'invite secondaire, que la coupure doit intervenir à la fin d'un mot, et que l'invite principale est **%**.

Nom d'une commande

Le premier mot d'une commande est obligatoirement son nom. Pour certaines commandes, ce nom est unique.

Indicateurs

Plusieurs indicateurs peuvent suivre le nom de la commande. Les indicateurs modifient l'opération d'une commande et sont parfois appelés *options*. Un indicateur est encadré d'espaces ou de tabulations, et il est généralement précédé d'un tiret (–). Les exceptions sont **ps**, **tar** et **ar** qui ne nécessitent pas de tiret avant certains indicateurs. Par exemple, dans la commande :

```
ls –a –F
```

`ls` est le nom de la commande et `ls –a –F` sont les indicateurs.

Lorsque des indicateurs sont associés à une commande, ils doivent suivre immédiatement son nom. Les indicateurs comportant un seul caractère peuvent être combinés à un tiret, dans une commande. Ainsi, la commande précédente peut également s'écrire :

```
ls –aF
```

Certains paramètres doivent parfois être également précédés d'un tiret (–). Dans ce cas, utilisez le délimiteur constitué de deux tirets (`--`) avant le paramètre. Le signe `--` indique à la commande qu'il s'agit d'un paramètre et non d'un indicateur.

Par exemple, si vous souhaitez créer un répertoire`-tmp` et que vous entrez la commande suivante :

```
mkdir –tmp
```

Un message d'erreur semblable au suivant s'affiche :

```
mkdir : n'est pas un indicateur reconnu : t
 Syntaxe : mkdir [-p] [-m mode]   Répertoire   ...
```

Il convient en effet de spécifier :

```
mkdir -- -tmp
```

Votre nouveau répertoire `-tmp` est à présent créé.

Paramètres des commandes

A la suite du nom de la commande, peuvent se trouver un ou plusieurs indicateurs, suivis de paramètres. Les paramètres sont parfois appelés *arguments* ou *opérandes.* Les paramètres précisent les informations requises pour l'exécution de la commande. Certains paramètres sont dotés d'une valeur par défaut, adoptée si vous n'en précisez pas d'autre. Par exemple, dans la commande :

```
ls –a temp
```

`ls` est le nom de la commande, `–a` est l'indicateur et `temp` est le paramètre. La commande affiche tous (`–a`) les fichiers du répertoire `temp`. Dans l'exemple :

```
ls –a
```

en l'absence de paramètres, ce sont les fichiers du répertoire courant qui sont affichés par défaut. Dans l'exemple :

```
ls temp mail
```

aucun indicateur n'est donné, et `temp` et `mail` sont les paramètres. Dans ce cas, `temp` et `mail` sont deux noms de répertoires distincts. La commande **ls** affiche tous les fichiers de chacun de ces répertoires, mais pas les fichiers masqués.

Lorsqu'un paramètre ou l'argument–option est constitué ou contient une valeur numérique, le nombre est interprété comme un entier décimal – sauf spécification contraire. Les nombres dans l'intervalle de 0 à **INT_MAX**, comme défini dans le fichier **/usr/include/sys/limits.h**, sont syntaxiquement reconnus comme des valeurs numériques.

Si une commande que vous souhaitez utiliser accepte les nombres négatifs comme paramètre ou arguments–options, vous pouvez utiliser les nombres de l'intervalle **INT_MIN** à **INT_MAX**, comme défini dans le fichier **/usr/include/sys/limits.h**. Ce qui ne signifie pas que tous les nombres de cet intervalle sont sémantiquement corrects. Certaines commandes (commandes d'impression notamment) sont limitées à un intervalle plus restreint. Le message d'erreur généré dans ce cas précise qu'il s'agit d'un problème de dépassement des limites de l'intervalle, et non de syntaxe incorrecte.

Lecture des lignes de syntaxe

Les instructions d'usage permettent de représenter la syntaxe d'une commande et sont constituées de symboles, tels que les parenthèses ([ ]), les accolades ({ }) et les barres verticales (|). Voici un exemple concernant la commande **unget** :

unget [**–r** *SID*] [**–s**] [**–n**] *Fichier*...

En outre :

- Les éléments devant être entrés littéralement sur la ligne de commande sont en **gras**. Ces éléments comprennent le nom de la commande, les indicateurs et les caractères littéraux.
- Les éléments représentant les variables devant être remplacés par un nom sont en *italique*. Ces éléments comprennent les paramètres qui suivent les indicateurs et les paramètres que la commande lit, tels que *Fichiers* et *Répertoires*.
- Les paramètres entre crochets sont facultatifs.
- Les paramètres entre accolades sont obligatoires.
- Les paramètres isolés sont obligatoires.
- Une barre verticale signifie que vous ne pouvez sélectionner qu'un paramètre. Par exemple, [a | b] signifie que vous *pouvez* choisir a, b, ou rien du tout. De la même manière, { a | b] signifie que vous *devez* choisir soit a, soit b.
- Les points de suspension (. . .) signifient que le paramètre peut être répété sur la ligne de commande.
- Le tiret (`–`) représente l'entrée standard.

Lancement de Web–based System Manager

Web–based System Manager est une interface GUI de gestion du système, soit à partir d'un écran rattaché en local ou à distance, depuis un autre système AIX ou depuis un ordinateur personnel équipé d'un navigateur Web. Vous pouvez lancer Web–based System Manager via différentes manières :

Action à réaliser	**A partir de**
Entrez la commande **wsm**.	Un terminal à ligne de commande dans l'environnement Common Desktop Environment (CDE)
Entrez la commande **wsm**.	Un terminal à ligne de commande dans l'environnement AIXwindows
Allez sur le dossier **System_Admin** et cliquez sur l'icone **Management Console**	A partir du gestionnaire d'application CDE
Lancez un navigateur Web compatible à HTML 3.2	Sur un PC configuré comme décrit dans le manuel *AIX 5L Version 5.3 Web–based System Manager Administration Guide*

Exécution d'autres commandes avec la commande smit

La commande **smit** est un outil permettant de lancer d'autres commandes. Les noms des commandes sont dans ce cas spécifiés comme paramètres de la commande **smit**, laquelle vous amène alors souvent à un sous–menu ou à la boîte de dialogue correspondant à la commande spécifiée. Par exemple, **smit lsuser** appelle directement la boîte de dialogue **List All Users**, qui permet d'afficher la liste des attributs utilisateur sur votre système.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **smit** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Localisation d'une commande ou d'un programme (commande whereis)

La commande **whereis** localise la commande ou le programme spécifié dans les répertoires source et binaires, et les sections de manuels. Elle procède à la recherche à partir d'une liste d'emplacements standard.

Voir les exemple ci–après :

- Par exemple, pour rechercher les fichiers du répertoire courant non documentés, entrez :

```
whereis –m –u *
```

- Pour rechercher tous les fichiers contenant le mot `Mail`, entrez :

```
whereis Mail
```

Le système affiche un écran semblable à celui-ci :

```
Mail : /usr/bin/Mail /usr/lib/Mail.rc
```

Reportez–vous à la commande **whereis** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Affichage des informations sur une commande (commande man)

La commande **man** affiche des informations sur les commandes, les sous–programmes et les fichiers. Le format général de la commande **man** est le suivant :

```
man NomCommande
```

Par exemple, pour obtenir des informations sur la commande **pg**, entrez :

```
man pg
```

Appuyez sur Entrée.

Le système affiche un écran semblable à celui-ci :

```
Commande pg

 Utilisation

 Formate les fichiers à l'affichage.

 Syntaxe

 pg [ - Nombre ] [ -c ] [ -e ] [ -f ] [ -n ] [ -p chaîne ]
 [ -s ] [ +Ligne | +/Forme/ ] [ Fichier ... ]

 Description

 La commande pg lit un nom de fichier à partir du paramètre Fichier et
 écrit le fichier dans la sortie standard, un écran à la fois. Si vous
 spécifiez un - (tiret) comme paramètre Fichier ou exécutez la commande
 pg sans options, la commande pg lit l'entrée standard. Chaque écran
 est suivi d'une invite. Si vous appuyez sur Entrée, une autre page
 s'affiche. Les sous-commandes utilisées avec la commande pg vous
 permettent d'afficher le fichier et d'y effectuer des recherches.
```

Reportez–vous à la commande **man** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Affichage de la fonction d'une commande (commande whatis)

La commande **whatis** recherche la commande, l'appel système, la fonction de bibliothèque ou le nom de fichier spécial, spécifié par le paramètre *Commande*, dans la base de données que vous avez créée via la commande **catman –w**. La commande **whatis** affiche l'en–tête de section correspondant. Vous pouvez alors, en lançant la commande **man**, obtenir des informations complémentaires.

La commande **whatis** équivaut à la commande **man –f**.

Par exemple, pour connaître l'objet de la commande**ls**, entrez :

```
whatis ls
```

Appuyez sur Entrée.

Le système affiche un écran semblable à celui-ci :

```
ls(1)  -Affiche le contenu d'un répertoire.
```

Reportez–vous à la commande **whatis** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Liste des commandes précédemment entrées (commande history)

Utilisez la commande **history** pour afficher la liste de toutes les commandes que vous avez entrées précédemment. La commande **history** est une commande intégrée du shell Korn, qui affiche la liste des 16 dernières commandes entrées. Le shell Korn enregistre ces commandes dans un fichier d'historique de commande, généralement appelé **$HOME/.sh_history**. Cette fonction permet de gagner du temps lorsque vous devez répéter une commande.

Par défaut, le shell Korn enregistre le texte des 128 dernières commandes pour un utilisateur non racine et 512 commandes pour un utilisateur racine. La taille du fichier d'historique (spécifiée par la variable **HISTSIZE**) n'est pas limitée ; un fichier trop volumineux risque toutefois de ralentir le shell Korn.

Remarque : Le shell Bourne ne prend pas en charge l'historique des commandes.

Pour plus d'informations sur les shells, reportez–vous à la section Shells, page12-1.

Par exemple, pour afficher la liste des commandes précédentes, entrez, à l'invite :

```
history
```

Appuyez sur Entrée.

La commande **history** liste les 16 dernières commandes. Le système affiche un écran semblable à celui-ci :

```
928   ls
929   mail
930   printenv MAILMSG
931   whereis Mail
932   whatis ls
933   cd /usr/include/sys
934   ls
935   man pg
936   cd
937   ls | pg
938   lscons
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
```

La liste affiche d'abord la position de la commande dans le fichier **$HOME/.sh_history** suivie de la commande

Par exemple, pour afficher les 5 dernières commandes, entrez, à l'invite :

```
history -5
```

Un menu semblable au suivant s'affiche :

Une liste semblable à la suivante s'affiche :

```
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
944   history -5
```

La commande **history** suivie d'un numéro affiche les commandes émises, à partir de ce numéro.

Par exemple, pour afficher les dernières commandes depuis 938, entrez, à l'invite :

```
history 938
```

Une liste semblable à la suivante s'affiche :

```
938   lscons
939   tty
940   ls *.txt
941   printenv MAILMSG
942   pwd
943   history
944   history -5
945   history 938
```

Répétition de commandes à l'aide de l'alias r

Pour répéter une commande antérieure, lancez l'alias du shell Korn **r**. Tapez **r**, suivi éventuellement du numéro ou du (des) premier(s) caractère(s) de la commande.

Entrez **lsdisp** à l'invite si vous souhaitez afficher la liste des écrans actuellement disponibles sur le système. L'invite système affiche les informations à l'écran :
Si vous souhaitez qu'il les réaffiche, entrez, à l'invite :

```
r
```

Le système réexécute la dernière commande. Dans cet exemple, la commande **lsdisp** est exécutée.

Pour répéter la commande **ls *.txt**, entrez, à l'invite :

```
r ls
```

L'alias shell Korn **r** localise la dernière commande commençant par le(s) caractère(s) spécifié(s).

Substitution de commandes à l'aide de l'alias r

Vous pouvez également utiliser l'alias**r** du shell Korn pour modifier une commande avant de la lancer. Vous pouvez, dans ce cas, passer par un paramètre de substitution de la forme *Ancien*=*Nouveau* pour modifier la commande avant de l'exécuter.

L'exemple suivant montre comment utiliser l'alias **r** :

- Si la ligne de commande 940 spécifie **ls *.txt** et que vous souhaitez exécuter **ls *.exe**, entrez, à l'invite :

  ```
  r txt=exe 940
  ```

  La commande 940 est exécutée, avec substitution de **exe** par **txt**.

- Si la commande 940 est la dernière des commandes commençant par la lettre*l*, vous pouvez également entrer :

  ```
  r txt=exe l
  ```

  Remarque : Seule la première occurrence de l'*ancienne* chaîne est remplacée par la *nouvelle*. Entrer l'alias shell Korn **r** sans numéro de commande ou lettre d'identification effectue la substitution dans la dernière commande spécifiée.

Edition de la commande history

L'utilisation de la commande intégrée du shell Korn **fc** permet d'afficher et de modifier tout ou partie du fichier d'historique. Le cas échéant, précisez le numéro ou l'intervalle de numéros (ou encore les premiers caractères) des commandes qui vous intéressent. Vous pouvez spécifier une ou plusieurs commandes.

Si vous ne spécifiez pas un programme d'édition comme argument dans la commande intégrée du shell Korn **fc**, l'éditeur spécifié par la variable **FCEDIT** est utilisé. Si la variable **FCEDIT** n'est pas définie, l'éditeur **/usr/bin/ed** est utilisé. Les commandes modifiées sont affichées et exécutées dès que vous quittez l'éditeur. Utilisez la commande **printenv** pour afficher la valeur de la variable **FCEDIT**.

Les exemples suivants décrivent comment modifier l'historique de commande :

- Si vous souhaitez exécuter la commande :

  ```
  cd /usr/tmp
  ```

  très proche de la ligne de commande `933`, entrez, à l'invite :

  ```
  fc 933
  ```

  L'éditeur par défaut est appelé, affichant la ligne de commande 933. Modifiez include/sys en tmp, et la commande modifiée est exécutée lorsque vous quittez l'éditeur.

- Vous pouvez également choisir l'éditeur que vous souhaitez utiliser dans la commande **fc**. Par exemple, pour modifier une commande avec l'éditeur `/usr/bin/vi`, entrez, à l'invite :

  ```
  fc -e vi 933
  ```

  L'éditeur **vi** est chargé, affichant la commande 933.

- Vous pouvez également indiquer une suite de commandes à modifier.
  Par exemple, pour modifier les commandes 930 à 940, entrez, à l'invite :

  ```
  fc 930 940
  ```

  L'éditeur par défaut est appelé, affichant les lignes de commande 930 à 940. Lorsque vous quittez l'éditeur, toutes les commandes affichées sont exécutées en séquence.

Création d'un alias de commande (commande alias du Shell)

Créer un *alias* permet d'attribuer un mnémonique à une commande, à un nom de fichier ou à tout texte shell. Vous pouvez ainsi accélérer le lancement des tâches que vous exécutez fréquemment. La commande intégrée du shell Korn **alias** permet de définir un mot comme alias de certaines commandes. Vous pouvez utiliser les alias pour redéfinir des commandes intégrées, mais pas des mots réservés.

L'initiale d'un alias peut être n'importe quel caractère imprimable (non spécial).
Les autres caractères doivent respecter les règles applicables aux noms de fichiers.

Format de création d'un alias :

```
alias   Name  =   String
```

où *Name* indique le nom de l'alias et *String*, une chaîne de caractères.
Si *String* comporte des espaces, mettez–la entre guillemets.

Voici quelques exemples qui montrent comment créer un alias :

- Pour créer un alias de la commande **rm –i** (qui vous demande confirmation avant de supprimer des fichiers), entrez, à l'invite :

```
alias rm="/usr/bin/rm –i"
```

  A partir de là, chaque fois que vous entrez la commande **rm**, la commande **/usr/bin/rm–i** est exécutée.

- Pour créer un alias **dir** de la commande **ls –alF | pg** (qui affiche des informations complètes sur tous les fichiers du répertoire courant, fichiers invisibles compris, marque l'un des fichiers exécutables d'un `*` et les répertoires d'un / et défile écran par écran), entrez, à l'invite :

```
alias dir="/usr/bin/ls –alF | pg"
```

  A partir de là, chaque fois que vous entrez la commande **dir**, la commande **/usr/bin/ls –alF | pg** est exécutée.

- Pour afficher tous les alias définis, entrez, à l'invite :

```
alias
```

  Le système affiche des informations similaires à l'exemple suivant :

```
rm="/usr/bin/rm –i"
dir="/usr/bin/ls –alF | pg"
```

Utilisation des commandes de formatage de texte

Vous disposez de commandes de formatage pour les textes composés avec le jeu de caractères internationaux utilisé pour les langues européennes.

Formatage de texte en caractères internationaux

Le jeu étendu de caractères internationaux comporte les caractères et les symboles utilisés par nombre de langues européennes, avec un sous–ensemble ASCII composé de caractères, de chiffres et de signes de ponctuation anglais.

Tous les caractères du jeu étendu ont une forme ASCII, qui permet de représenter le caractères en entrée. Les caractères peuvent également être entrés directement au clavier (sous réserve que celui–ci prenne en charge le jeu étendu de caractères européens).

Les commandes suivantes permettent de traiter des textes dans différentes langues, si tous les caractères sont mono–octets. Ces commandes se trouvent dans **/usr/bin**. (Celles suivies d'un astérisque (*) permettent de traiter les textes composés dans les langues multi-octets). Pour de plus amples informations sur les langues multi-octets, reportez–vous à Formatage de texte en caractères multi-octets, page 4-11.)

addbib*	hyphen	pic*	pstext
checkmm	ibm3812	ps4014	refer*
checknr*	ibm3816	ps630	roffbib*
col*	ibm5587G*	psbanne	soelim*
colcrt	ibm5585H–T*	psdit	sortbib*
deroff*	indxbib*	psplot	tbl*
enscript	lookbib*	psrev	troff*
eqn*	makedev*	psroff	vgrind
grap*	neqn*	psrv	xpreview*
hplj	nroff*		

Les commandes de formatage et les macros non citées ne s'appliquent pas aux caractères internationaux.

Entrée de caractères étendus mono-octets

Si votre unité d'entrée prend en charge les caractères du jeu étendu de caractères européens, vous pouvez les entrer directement. Sinon, passez par la séquence d'échappement ASCII :

La forme \[*N*], où *N* est un code hexadécimal de 2 ou de 4 chiffres pour le caractère.

Remarque : La forme NCesc `\<xx>` n'est plus prise en charge.

Les textes comportant des caractères étendus sont émis en sortie selon les conventions de formatage de la langue utilisée. Les caractères non définis pour l'interface d'une unité ne génèrent ni sortie ni notification d'erreur.

Bien que la plupart des noms de demandes, des macros et des commandes soient basés sur l'anglais, ils acceptent presque toutes les entrées (noms de fichiers, paramètres, etc.) comportant des caractères du jeu européen étendu.

Quant aux commandes **nroff** et **troff**, et à leurs préprocesseurs, l'entrée de la commande doit être en ASCII, ou une erreur de syntaxe irrémédiable peut survenir. Les caractères internationaux, mono ou multi-octets, peuvent être entrés entre guillemets, éventuellement à l'intérieur de texte à formater. Par exemple, à l'aide des macros de la commande **pic** :

```
define foobar %   Texte   %
```

Après la directive `define`, le premier nom `foobar`doit être en ASCII. Toutefois, le texte de remplacement `Texte`peut contenir des caractères non ASCII.

Formatage de texte en caractères multi-octets

Certaines commandes permettent de traiter des textes dans des langues multi-octets. Ces commandes peuvent être identifiées par un astérisque (*) dans la liste sous Formatage de texte en caractères internationaux, page 4-11. Les commandes de formatage de texte qui ne sont pas répertoriées dans cette liste ne peuvent pas traiter les caractères internationaux.

Entrée de caractères multi-octets

Si votre unité d'entrée prend en charge les caractères multi-octets, vous pouvez les entrer directement. Sinon, vous pouvez entrer un caractère multi-octet sous forme ASCII \[*N*], où ***N*** est un code hexadécimal de 2, 4, 6, 7, ou 8 chiffres pour le caractère.

Bien que la plupart des noms de demandes, des macros et des commandes soient basées sur l'anglais, elles acceptent presque toutes des entrées (noms de fichiers, paramètres, etc.) comportant des caractères multi-octets.

Pour les familiers du formatage de texte en caractères mono-octets, la liste suivante récapitule les caractéristiques remarquables ou propres aux environnements multi-octets :

- Il n'y a pas de césure du texte.
- Des types de formats spéciaux sont requis pour une sortie numérique multi-octets. Des types de formats japonais sont disponibles.
- La sortie est effectuée en lignes horizontales, remplies de gauche à droite.
- Les caractères sont automatiquement alignés en colonnes.
- Les caractères non définis pour l'interface d'une unité ne génèrent ni sortie ni notification d'erreur.

Présentation des process

Un programme ou une commande en cours d'exécution est appelé un *process*. Les process sont structurés selon une hiérarchie parent–enfant. Un process lancé par un programme ou une commande est un *process parent*; le process résultant étant un *process enfant*. Un process parent peut engendrer plusieurs process enfant, tandis qu'un process enfant ne peut avoir qu'un seul parent.

Le système attribue un numéro d'identification (PID) à chaque process, au moment de son lancement : si vous lancez plusieurs fois le même programme, un PID différent lui est associé à chaque fois.

Un process lancé mobilise une partie des ressources système. Lorsque plusieurs process sont simultanément en cours, un répartiteur (partie intégrante du système d'exploitation) assure le partage du temps machine, en fonction des priorités définies. Pour modifier ces priorités, vous disposez des commandes **nice** et **renice**.

> **Remarque :** Pour attribuer à un process une priorité supérieure, vous devez détenir les droits de l'utilisateur root. Tout utilisateur peut, en revanche, attribuer à un process une priorité moindre, via la commande `nice` (ou `renice` pour un process déjà en cours).

Cette section décrit les procédures suivantes :

Process d'avant et d'arrière-plan

Les process qui nécessitent un utilisateur pour les lancer ou pour interagir ave eux sont appelés *process d'avant–plan*. Les process exécutés indépendamment de l'utilisateur sont appelés *process d'arrière–plan*. Par défaut, programmes et commandes sont exécutés en avant-plan. Pour exécuter un process en arrière-plan, placez une perluète (&) à la fin du nom de la commande de lancement du process.

Process démon

Les démons sont des process exécutés sans contrôle. Ils se trouvent en permanence à l'arrière–plan et sont toujours disponibles. Ils sont généralement lancés et arrêtés en même temps que le système. Un démon, qui exécute des tâches système, peut être appelé par plusieurs tâches ou utilisateurs. Les démons sont démarrés par l'utilisateur ou le shell racine et arrêtés exclusivement par l'utilisateur racine. Par exemple, le process **qdaemon**

donne accès aux ressources système, telles que les imprimantes. Le démon **sendmail** est un autre démon commun.

Process zombie

Un *process zombie* est un process mort qui n'est plus en cours d'exécution, mais qui est toujours reconnu dans la table des process (autrement dit, qui possède un PID). Aucun autre espace système ne lui est attribué. Ces process se sont ou ont été arrêtés, mais continuent d'exister jusqu'à l'arrêt de leur process parent ou jusqu'à l'arrêt puis la relance du système. Les process zombie s'affichent comme `<defunct>` lorsqu'ils sont affichés par la commande **ps**.

Lancement d'un process

Pour lancer un process d'avant–plan à partir d'une station de travail, entrez le nom du programme ou de la commande à l'invite du système. Une fois démarré, le process entre en interaction avec vous. Vous ne pouvez envisager d'autres interactions (entrer une autre commande, par exemple) tant que le process n'a pas pris fin.

Un utilisateur peut lancer simultanément plusieurs process (par défaut, 40 au maximum).

Lancement d'un process d'avant–plan

Pour lancer un process d'avant–plan, entrez le nom du programme ou de la commande, assorti des paramètres et indicateurs appropriés :

```
$   CommandName
```

Lancement d'un process d'arrière–plan

Pour lancer un process d'arrière plan, entrez le nom du programme ou de la commande, assorti des paramètres et indicateurs appropriés, et suivi d'une perluète (&) :

```
$   CommandName  &
```

Pendant que le process s'exécute en arrière–plan, vous pouvez tout à fait lancer d'autres commandes ou programmes depuis votre poste d'affichage.

En règle générale, les process en arrière–plan sont fort utiles pour les commandes longues à exécuter. Toutefois, dans la mesure où ils accroissent la charge du processeur, l'ensemble des opérations du système est ralenti.

La plupart des process, même exécutés en arrière–plan, dirigent leurs sorties vers la sortie standard. Sauf s'ils ont été explicitement réacheminés, ils se dirigent vers l'écran. Pour éviter toute interférence entre les résultats d'un process en arrière-plan et votre travail en cours, nous vous conseillons de réacheminer les sorties des process d'arrière-plan vers un fichier ou une imprimante. Vous pourrez les consulter à votre guise quand vous serez disponible.

> **Remarque :** Dans certains cas, la séquence des sorties d'un process peut différer selon que le process est exécuté en avant ou en arrière–plan. Les programmeurs voudront peut–être utiliser le sous–programme **fflush** pour s'assurer que la sortie s'affiche dans l'ordre correct indépendamment du fait que le process est exécuté en avant ou en arrière-plan.

Tant qu'un process d'arrière–plan est en cours, vous pouvez vérifier son état via la commande **ps**.

Vérification des process (commande ps)

Lorsque le système est en cours de fonctionnement, plusieurs process sont également en cours d'exécution. Vous pouvez également utiliser la commande **ps** pour savoir quels process sont en cours d'exécution et pour afficher les informations qui leur sont relatives.

Commande ps

La commande **ps** possède plusieurs indicateurs permettant d'indiquer quels process et quelles informations il faut afficher.

Ainsi, pour afficher tous les process en cours, entrez, à l'invite :

```
ps -ef
```

Le système affiche un écran semblable à celui-ci :

```
    USER   PID  PPID   C    STIME    TTY  TIME CMD
    root     1     0   0   Jun 28      -  3:23 /etc/init
    root  1588  6963   0   Jun 28      -  0:02 /usr/etc/biod 6
    root  2280     1   0   Jun 28      -  1:39 /etc/syncd 60
    mary  2413 16998   2 07:57:30      -  0:05 aixterm
    mary 11632 16998   0 07:57:31  lft/1  0:01 xbiff
    mary 16260  2413   1 07:57:35  pts/1  0:00 /bin/ksh
    mary 16469     1   0 07:57:12  lft/1  0:00 ksh /usr/lpp/X11/bin/xinit
    mary 19402 16260  20 09:37:21  pts/1  0:00 ps -ef
```

Les colonnes des sorties précédentes sont définies comme suit :

USER	nom de connexion de l'utilisateur
PID	ID process
PPID	ID process parent
C	utilisation CPU du process
STIME	heure de début du process
TTY	contrôle de la station de travail
TIME	durée d'exécution totale du process
CMD	commande

Dans l'exemple précédent, l'ID process de la commande **ps –ef** est `19402`.
Son ID process parent est `16260` et la commande est **/bin/ksh**.

Dans l'exemple précédent, l'ID process de la commande **ps –ef** est `19402`. Son ID process parent est `16260`, la commande **/bin/ksh**.

Si la liste est très longue, la première partie défile hors de l'écran. Pour afficher la liste une page (écran) à la fois, associez la commande **ps** à la commande **pg**. A l'invite, entrez ce qui suit :

```
ps -ef | pg
```

Pour afficher des informations sur tous les process en cours, entrez, à l'invite :

```
ps gv
```

Cette commande affiche des statistiques sur chaque process actif. La sortie de cette commande est similaire à celle–ci :

```
    PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
      0      - A     0:44    7     8     8    xx     0     0  0.0  0.0 swapper
      1      - A     1:29  518   244   140    xx    21    24  0.1  1.0 /etc/init
    771      - A     1:22    0    16    16    xx     0     0  0.0  0.0 kproc
   1028      - A     0:00   10    16     8    xx     0     0  0.0  0.0 kproc
   1503      - A     0:33  127    16     8    xx     0     0  0.0  0.0 kproc
   1679      - A     1:03  282   192    12 32768   130     0  0.7  0.0 pcidossvr
   2089      - A     0:22  918    72    28    xx     1     4  0.0  0.0 /etc/sync
   2784      - A     0:00    9    16     8    xx     0     0  0.0  0.0 kproc
   2816      - A     5:59 6436  2664   616     8   852   156  0.4  4.0 /usr/lpp/
   3115      - A     0:27  955   264   128    xx    39    36  0.0  1.0 /usr/lib/
   3451      - A     0:00    0    16     8    xx     0     0  0.0  0.0 kproc
   3812      - A     0:00   21   128    12 32768    34     0  0.0  0.0
usr/lib/lpd/
   3970      - A     0:00    0    16     8    xx     0     0  0.0  0.0 kproc
   4267      - A     0:01  169   132    72 32768    16    16  0.0  0.0 /etc/sysl
   4514  lft/0 A     0:00   60   200    72    xx    39    60  0.0  0.0 /etc/gett
   4776  pts/3 A     0:02  250   108   280     8   303   268  0.0  2.0 -ksh
   5050      - A     0:09 1200   424   132 32768   243    56  0.0  1.0 /usr/sbin
   5322      - A     0:27 1299   156   192    xx    24    24  0.0  1.0 /etc/cron
   5590      - A     0:00    2   100    12 32768    11     0  0.0  0.0 /etc/writ
   5749      - A     0:00    0   208    12    xx    13     0  0.0  0.0 /usr/lpp/
   6111      - T     0:00   66   108    12 32768    47     0  0.0  0.0 /usr/lpp/
```

Reportez–vous à la commande **ps** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Définition de la priorité initiale d'un process (commande nice)

Pour attribuer à un process une priorité inférieure, lancez–le via la commande **nice** pour démarrer le process.

> **Remarque :** Pour attribuer à un process une priorité supérieure, vous devez détenir les droits de l'utilisateur racine.

Commande nice

Pour définir la priorité initiale d'un process, entrez :

```
nice –n   Numéro ChaîneCommande
```

où *Numéro* se trouve dans l'intervalle de 0 à 39, 39 étant la priorité inférieure. La *valeur nice* est la valeur décimale de la priorité de planification du système d'un process. Plus la valeur est élevée, plus la priorité est faible. La valeur zéro conserve la priorité de planification initiale. *ChaîneCommande* est la chaîne de commande du process à exécuter.

Reportez–vous à la commande **nice** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Vous pouvez également utiliser la commande **smit nice** pour réaliser cette tâche.

Modification de la priorité d'un process en cours d'exécution (commande renice)

Pour modifier la priorité d'un process en cours, lancez la commande **renice** à partir de la ligne de commande. Cette commande modifie la valeur nice d'un process.

> **Remarque :** Pour attribuer à un process (que vous n'avez pas lancé) une priorité supérieure, vous devez détenir les droits de l'utilisateur root.

commande renice

Pour modifier la priorité d'un process en cours, entrez :

```
renice   Priorité   -p   IDProcess
```

où *Priorité* est un nombre compris entre –20 et 20 (plus le nombre est élevé, plus la priorité est basse). La valeur zéro conserve la priorité de planification initiale. *IDProcess* est l'ID process pour lequel vous souhaitez modifier la priorité.

Vous pouvez également utiliser la commande **smit renice** pour réaliser cette tâche.

Interruption d'un process d'avant-plan

Si vous lancez un process d'avant-plan et que vous ne voulez pas qu'il se termine, vous pouvez l'interrompre en appuyant sur INTERRUPTION. Pour cela, utilisez Ctrl–C ou Ctlr–Supp. Pour savoir quelle est votre touche d'INTERRUPTION, reportez–vous à Liste des affectations de touches du terminal (commande stty), page 2-10.

> **Remarque :** La touche d'INTERRUPTION (Ctrl–C) n'arrête pas les process d'arrière–plan. Pour annuler un process d'arrière–plan, vous devez utiliser la commande **kill**.

La plupart des commandes simples s'exécutent en un laps de temps tellement bref qu'elles se terminent avant même que vous n'ayez eu le temps de les annuler. C'est pourquoi les exemples qui suivent portent sur une commande dont le temps d'exécution est un peu plus long : **find / –type f**. Cette commande affiche le nom des chemins d'accès de tous les fichiers du système. Vous n'avez pas besoin d'approfondir l'étude de la commande **find** pour poursuivre : elle est citée uniquement pour illustrer la manière de travailler avec les process.

Dans l'exemple suivant, la commande **find** démarre un process. Quelques secondes après le démarrage du process, vous pouvez l'interrompre en appuyant sur la touche d'INTERRUPTION :

```
$ find / -type f
/usr/sbin/acct/lastlogin
/usr/sbin/acct/prctmp
/usr/sbin/acct/prdaily
/usr/sbin/acct/runacct
/usr/sbin/acct/sdisk
/usr/sbin/acct/shutacct INTERRUPT (Ctrl-C)
$ _
```

L'invite système est réaffichée. Vous pouvez lancer d'autres commandes.

Arrêt d'un process d'avant-plan

Il est possible d'arrêter un process sans que son PID soit supprimé de la table des process. Le process peut être arrêté par Ctrl–Z, par exemple.

> **Remarque :** Ctrl–Z fonctionne avec les shells Korn (**ksh**) et C (**csh**), mais pas avec le shell Bourne (**bsh**).

Relance d'un process

Cette procédure explique comment relancer un process arrêté par Ctrl-Z.

> **Remarque :** Ctrl–Z fonctionne avec les shells Korn (**ksh**) et C (**csh**), mais pas avec le shell Bourne (**bsh**). Pour relancer un process arrêté, vous devez soit l'avoir vous–même lancé, soit être un utilisateur racine.

1. Pour afficher tous les process en cours ou arrêtés, mais non ceux tués (commande kill), entrez :

```
ps  -ef
```

Vous pouvez chaîner cette commande avec une commande **grep** pour mieux cibler les process qui vous intéressent. Ainsi, pour relancer une session **vi**, entrez :

```
ps -ef | grep vi
```

Appuyez sur Entrée. Cette commande affichera uniquement les lignes de la sortie de la commande **ps** contenant le mot `vi`. Une sortie semblable à la suivante est affichée :

```
UID    PID   PPID   C      STIME       TTY  TIME  COMMAND
 root   1234  13682  0      00:59:53    -    0:01  vi test
 root  14277  13682  1      01:00:34    -    0:00  grep vi
```

2. Dans la sortie de la commande **ps**, recherchez le process que vous souhaitez relancer et notez son ID process. Dans cet exemple, l'ID process est `1234`.
3. Pour transmettre le signal de continuation au process arrêté, entrez :

```
kill -19 1234
```

Modifiez l'ID de votre process, `1234`. La commande `-19` indique le signal CONTINUE, qui relance le process en arrière-plan. Si le process peut être exécuté en arrière-plan, vous avez terminé la procédure. Si le process doit être exécuté en avant-plan (comme dans une session **vi**), vous devez réaliser l'étape suivante.

4. Pour amener le process à l'avant–plan, entrez :

```
fg 1234
```

Modifiez à nouveau l'ID de votre process, `1234`. Votre process doit à présent être exécuté en avant-plan. (Vous êtes à présent dans votre session d'édition **vi**).

Planification d'un process pour une opération ultérieure (commande at)

Vous pouvez configurer un process comme étant un *batch process* (traitement par lots) pour qu'il fonctionne en arrière-plan à une heure définie. Les commandes **at** et **smit** vous permettent d'entrer des noms de commandes à exécuter à un moment ultérieur et de spécifier quand ces commandes doivent être lancées.

Remarque : Les fichiers **/var/adm/cron/at.allow** et **/var/adm/cron/at.allow** vérifient si vous pouvez utiliser la commande **at**. Un utilisateur root peut créer, éditer ou supprimer ces fichiers. Leurs entrées comprennent un nom de connexion utilisateur par ligne. Voici un exemple d'ACL d'un fichier **at.allow** :

```
root
nick
dee
sarah
```

Si le fichier **at.allow** exite, seuls les utilisateurs dont les noms de connexion y sont répertoriés peuvent utiliser la commande **at**. Un administrateur système peut explicitement interdire à un utilisateur d'utiliser la commande **at** en répertoriant le nom de connexion de cet utilisateur dans le fichier **at.deny**. Si seul le fichier **at.deny** existe, n'importe quel utilisateur dont le nom ne figure pas dans ce fichier peut utiliser la commande **at**.

Vous pouvez également utiliser la commande **at** si l'une des conditions suivantes est vraie :

- Le fichier **at.allow** et le fichier **at.deny** n'existent pas (utilisateur root autorisé uniquement).
- Le fichier **at.allow** existe mais le nom de connexion utilisateur n'y figure pas.
- Le fichier **at.deny** existe et le nom de connexion utilisateur y apparaît.

Si le fichier **at.allow** n'existe pas et que le fichier **at.deny** n'existe pas non plus ou qu'il est vide, seul un utilisateur root peut soumettre un travail avec la commande **at**.

La syntaxe de la commande **at** vous permet de spécifier une chaîne date et heure, ou encore une chaîne incrément pour définir le moment d'exécution du process. Elle vous autorise également à préciser le shell ou la file d'attente à utiliser. Voici quelques exemples.

Commande at

1. Entrez l'heure de lancement du programme :

Si votre nom de connexion est joyce et que vous disposez d'un script appelé WorkReport que vous voulez exécuter à minuit, procédez comme suit.

```
at midnight
```

2. Entrez le nom des programmes à exécuter, en appuyant sur Entrée après chaque nom. Appuyez ensuite sur Ctrl-D (fin de fichier) pour terminer.

```
WorkReport^D
```

Un message semblable au suivant s'affiche :

```
job joyce.741502800.a at Fri Jul  6 00:00:00 CDT 2002.
```

Un numéro de travail est attribué au programme WorkReport `joyce.741502800.a` et le travail sera exécuté à minuit le 6 juillet.

3. Pour afficher la liste des programmes qui seront exécutés en différé, entrez :

```
at –l
```

Le système affiche un écran semblable à celui-ci :

```
joyce.741502800.a       Fri Jul  6 00:00:00 CDT 2002
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **at** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Liste de tous les process planifiés (commande at ou atq)

Utilisez l'indicateur **–l** avec la commande **at** ou avec la commande **atq** pour afficher la liste des process planifiés. Les deux commandes génèrent la même sortie, la commande **atq** pouvant en outre trier les process selon l'heure de lancement de la commande **at** et n'afficher que les seuls process en file d'attente.

Pour afficher tous les process planifiés, vous lancez :

- la commande **at** à partir de la ligne de commande.
- la commande **atq**.

Pour limiter les utilisateurs de la commande **at**, reportez–vous à la **Remarque** dans Planification d'un process pour une opération ultérieure (commande at), page 4-18.

Commande at

Pour afficher la liste des process planifiés, entrez :

```
at –l
```

Cette commande affiche la liste des process planifiés dans votre file d'attente. Si vous êtes utilisateur root, elle affiche tous les process planifiés de tous les utilisateurs. Pour en savoir plus sur la syntaxe, reportez-vous à la commande **at**.

commande atq

Les exemples ci–après illustrent les différentes utilisations de la commande **atq** :

- Pour afficher la liste des process planifiés de la file d'attente, entrez :

```
atq
```

- Si vous êtes utilisateur root, vous pouvez afficher les process planifiés d'un utilisateur en entrant :

```
atq UserName
```

- Pour afficher le nombre de process de la file d'attente, entrez :

```
atq –n
```

Suppression d'un process du planning (commande at)

Vous pouvez supprimer un process planifié avec la commande **at** en utilisant l'indicateur **–r**. Pour limiter les utilisateurs de la commande **at**, reportez–vous à la **Remarque** dans Planification d'un process pour une opération ultérieure (commande at), page 4-18.

commande at ou atq

Les exemples ci–après illustrent les différentes utilisations de la commande **at** ou **atq** :

1. Pour déprogrammer un process, vous devez connaître son numéro. Utilisez les commandes **at –l** ou **atq** pour obtenir le numéro d'un process. Pour plus d'informations, reportez–vous à la section Liste de tous les process planifiés (commande at ou atq), page 4-19.
2. Lorsque vous connaissez le numéro du process à déprogrammer, entrez :

```
at –r NuméroProcess
```

Vous pouvez également utiliser la commande **smit rmat** pour réaliser cette tâche.

Suppression d'un process d'arrière–plan (commande kill)

Si la touche d'INTERRUPTION n'a pas eu l'effet escompté sur le process d'avant-plan ou si vous souhaitez arrêter un process d'arrière-plan, vous pouvez recourir à la commande **kill**. Avant d'arrêter un process à l'aide de la commande **kill**, vous devez connaître son ID process. Le format général de la commande **kill** est le suivant :

```
kill   IDProcess
```

Remarque : Pour supprimer un process, vous devez soit l'avoir lancé, soit être un utilisateur root. Le signal par défaut d'un process suite à la commande **kill** est –15 (SIGTERM).

Commande kill

Remarque : Pour supprimer un process zombie, vous devez supprimer son parent.

1. Pour déterminer le PID du process à supprimer, lancez la commande **ps**. Vous pouvez chaîner cette commande avec une commande **grep** pour mieux cibler les process qui vous intéressent. Ainsi, pour obtenir le PID d'une session vi, entrez :

```
ps –l | grep vi
```

2. Dans l'exemple suivant, la commande **find** permet de faire une exécution en arrière–plan. Vous pouvez alors décider d'arrêter le process. Emettez la commande **ps** pour lister les numéros de PID.

```
$ find / –type f > dir.paths &
 [1]    21593
 $ ps
    PID   TTY  TIME  COMMAND
   1627  pts3  0:00  ps
   5461  pts3  0:00  ksh
  17565  pts3  0:00  –ksh
  21593  pts3  0:00  find / –type f
 $ kill 21593
 $ ps
    PID   TTY  TIME  COMMAND
   1627  pts3  0:00  ps
   5461  pts3  0:00  ksh
  17565  pts3  0:00  –ksh
 [1]  +  Terminated 21593     find  /  –type f > dir.paths  &
```

La commande **kill 21593** arrête le process d'arrière–plan **find** et la deuxième commande **ps** renvoie un état sans informations sur le PID 21593. Le système n'affichera pas le message d'arrêt avant que vous n'entriez la commande suivante, à moins que la commande soit **cd**.

La commande **kill** vous permet d'arrêter les process d'arrière–plan. Vous voulez peut–être réaliser cette tâche si vous avez, par erreur, lancé un process en arrière-plan ou que vous vous apercevez qu'il est trop long, par exemple.

Reportez–vous à la commande **kill** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

La commande **kill** peut aussi être utilisée dans **smit** en entrant :

```
smit kill
```

Récapitulatif des commandes pour les commandes et les process

Commandes

alias	La commande shell imprime une liste d'alias sur une sortie standard
history , page 12-96	La commande shell affiche la liste des événements de l'historique
man	Affiche les informations sur les commandes, les sous–programmes et les fichiers en ligne
wsm	Effectue une gestion de système à partir d'un navigateur web
whatis	Décrit la fonction d'une commande
whereis	Localise la source, binaire ou manuelle, des programmes installés

Process

at	Exécute des commandes à un moment ultérieur, liste tous les process planifiés ou supprime un process du planning
atq	Affiche la file de travaux en attente de l'exécution
kill	Envoie un signal aux process en cours d'exécution
nice	Exécute une commande à un niveau de priorité inférieur ou supérieur
ps	Affiche l'état en cours des process.
renice	Modifie la priorité des process en cours d'exécution

Chapitre 5. Réacheminement des entrées/sorties

Le système d'exploitation AIX permet la gestion des entrées et sorties (E–S) de données de et vers votre système en utilisant des commandes et symboles spécifiques E–S. Vous pouvez gérer les entrées en indiquant l'emplacement à partir duquel les données sont rassemblées. Vous pouvez par exemple indiquer de lire les entrées quand les données sont entrées sur le clavier (entrées standard) ou de lire les sorties à partir d'un fichier. Vous pouvez gérer les sorties en indiquant où les données s'affichent et sont enregistrées. Vous pouvez indiquer d'envoyer les données de sortie à l'écran (sortie standard) ou de les envoyer vers un fichier.

Le système est multitâche, ce qui signifie qu'il traite les process en combinaison les uns avec les autres. Ce chapitre aborde les avantages du réacheminement des entrées et sorties et de l'association des process.

Cette section décrit les points suivants :

Entrées, sorties et erreurs standard

Lorsqu'une commande est lancée, elle attend normalement que les fichiers suivants soient ouverts : entrées, sorties et erreurs standards (appelés parfois *sortie erreur* ou sortie *diagnostic* Un *descripteur* de *fichier* est un nombre associé à chacun de ces fichiers comme suit :

Descripteur de fichier 0	Entrée standard
Descripteur de fichier 1	Sortie standard
Descripteur de fichier 2	Sortie erreur standard (diagnostic)

Un processus enfant hérite normalement de ces fichiers de son parent. Les trois fichiers sont initialement attribués au poste de travail (0 pour le clavier, 1 et 2 pour l'écran). Le shell leur permet d'être réacheminé vers un autre emplacement avant que le contrôle ne soit passé sur une commande.

Lorsque vous entrez une commande, si aucun nom de fichier n'est précisé, votre clavier représente l'*entrée standard*, parfois appelée *stdin*. A la fin d'une commande, les résultats s'affichent sur votre écran.

Votre écran représente la *sortie standard*, parfois appelée *stdout*. Par défaut, les commandes gardent leur entrée à partir de l'entrée standard et envoie les résultats à la sortie standard.

Les messages d'erreur sont acheminés aux erreur standard, parfois appelées *stderr*. Par défaut, il s'agit de votre écran.

Ces actions d'entrées et de sorties par défaut peuvent varier. Vous pouvez utiliser un fichier comme entrées et envoyer les résultats d'une commande à un fichier. Il s'agit du *réacheminement des entrées/sorties*.

La sortie d'une commande qui normalement s'affiche à l'écran peut être aisément réacheminée vers un fichier. il s'agit du *réacheminement des sorties*. Cette fonction est très utile lorsque le résultat d'une commande est trop long pour être aisément lu à l'écran ou que vous souhaitez regrouper plusieurs petits fichiers dans un seul.

Bien qu'elle ne soit pas autant utilisée que le réacheminement des sorties, les entrées d'une commande normalement issues du clavier peuvent être également réacheminées à partir d'un fichier. Il s'agit du *réacheminement des entrées*. Le réacheminement des entrées permet de préparer un fichier à l'avance ; ainsi la commande peut lire le fichier.

Réacheminement des sorties standard

Lorsque la mention *>nomfichier* est indiquée à la fin d'une commande, la sortie de la commande est ajoutée au nom du fichier indiqué. Le `>` symbole est connu comme l'*opérateur de réacheminement* des *sorties*.

Toute commande dont les résultats sont normalement affichés à l'écran peut être réacheminée de la sorte.

Réacheminement des sorties vers un fichier

Il est possible de réacheminer la sortie d'un processus vers un fichier en saisissant la commande suivie de l'opérateur de réacheminement des sorties et du nom de fichier. Par exemple, pour réacheminer les résultats de la commande **who** dans un fichier nommé **users**, entrez :

```
who > users
```

Remarque : Si le fichier **users** existe déjà, il est supprimé et remplacé sauf si l'option **noclobber** des commandes (shell C) intégrées **set ksh** (Korn shell) ou **csh** est indiquée.

Pour afficher le contenu du fichier **users**, entrez :

```
cat users
```

Une liste similaire à l'exemple ci–après s'affiche :

```
denise    lft/0 May 13 08:05
marta     pts/1 May 13 08:10
endrica   pts/2 May 13 09:33
```

Réacheminement des sorties avec ajout à un fichier

Lorsque la mention > > *nomfichier* est indiquée à la fin d'une commande, les sorties de la commande sont ajoutées au nom du fichier indiqué, plutôt que d'écraser les données existantes. Le >> symbole est connu comme l'*opérateur de réacheminement*.

Par exemple, pour ajouter **file2** à **file1**, entrez :

```
cat file2 >> file1
```

Appuyez sur Entrée.

Remarque : Si le fichier **file1** n'existe pas, il est créé sauf si l'option **noclobber** de la commande (shell C) intégrée **set ksh** (Korn shell) ou **csh** est indiquée.

Création d'un fichier texte avec réacheminement à partir du clavier

Utilisée seule, la commande **cat** considère comme entrées toutes les saisies clavier. Vous pouvez rediriger ces entrées vers un fichier. Appuyez sur Ctrl–D (sur une nouvelle ligne) pour signaler la fin du texte.

A l'invite du système, entrez ce qui suit :

```
cat > nomfichier
 This is a test.
 ^D
```

Concaténation de fichiers texte

L'association de plusieurs fichiers en un seul fichier s'appelle la *concaténation*.

L'exemple suivant génère **file4**, composé de **file1**, **file2** et **file3**.

Voir les exemple ci–après :

- A l'invite du système, entrez ce qui suit :

```
cat file1 file2 file3 > file4
```

- L'exemple suivant affiche une erreur communément rencontrée à la concaténation de fichiers :

```
cat file1 file2 file3 > file1
```

Attention : Dans cet exemple, la commande **cat** peut ajouter **file1**, **file2** et **file3** dans **file1**. La commande **cat** crée tout d'abord le fichier de sorties, elle supprime donc le contenu de **file1** et lui ajoute **file2** et **file3**.

Réacheminement des entrées standard

Lorsque la mention < *nomfichier* est indiquée à la fin d'une commande, les entrées de la commande sont ajoutées au nom du fichier indiqué. Le < symbole est connu comme l'*opérateur réacheminement des entrées*.

> **Remarque :** Seules les commandes qui normalement gardent leurs entrées à partir du clavier peuvent subir un réacheminement de leurs entrées.

Par exemple, pour envoyer le fichier **letter1** comme message à l'utilisateur `denise`, via la commande **mail**, entrez :

```
mail denise < letter1
```

Elimination des sorties via le fichier /dev/null

Le fichier **/dev/null** est un fichier spécial. Ce fichier a une propriété unique : il est toujours vide. Toutes les données que vous envoyez vers **/dev/null** sont supprimées. Ceci est une fonction utile lorsque vous exécutez un programme dont les sorties n'ont aucun intérêt pour vous.

Soit, par exemple, le programme **myprog** qui lit les données à l'écran et renvoie des messages que vous souhaitez ignorer. Pour qu'il lise les entrées dans le fichier **monscript** et rejette les messages de sortie standard, entrez :

```
myprog < myscript >/dev/null
```

Dans cet exemple, `myprog` utilise le fichier `myscript` comme entrées et toutes les sorties standard sont éliminées.

Réacheminement des erreurs standard et autres sorties

Outre l'entrée standard et la sortie standard, les commandes génèrent souvent d'autres types de sortie comme les messages d'erreur ou d'état connus sous le nom de sortie de diagnostic. Tout comme la sortie standard, la sortie erreur standard apparaît à l'écran, à moins qu'elle ne soit réacheminée.

Pour réacheminer l'erreur standard ou une autre sortie, utilisez un descripteur de fichier. Un *descripteur de fichier* est un nombre associé à chacun des fichiers d'entrée–sortie généralement utilisés par une commande. Il est également possible de spécifier des descripteurs de fichiers pour réacheminer l'entrée standard et la sortie standard, mais elles disposent déjà de valeurs par défaut. Les nombres ci–après sont associés à l'entrée standard, la sortie standard et la sortie erreur standard :

0	Entrée standard (clavier)
1	Sortie standard (écran)
2	Erreur standard (écran)

Pour réacheminer la sortie erreur standard, saisissez le descripteur de fichier numéro 2 devant la sortie ou ajoutez les symboles de réacheminement (> ou > >) suivis d'un nom de fichier. Par exemple, la commande suivante prend la sortie d'erreur standard de la commande **cc** à l'emplacement où elle est utilisée pour compiler le fichier **testfile.c** et l'ajoute à la fin du fichier **ERRORS** :

```
cc testfile.c 2 >> ERRORS
```

Il est également possible de réacheminer d'autres types de sortie à l'aide des descripteurs de fichiers allant de 0 à 9. Ainsi, si la commande **cmd** consigne la sortie vers le descripteur de fichier numéro 9, vous pouvez réacheminer cette sortie vers le fichier **savedata** à l'aide de la commande suivante :

```
cmd 9> savedata
```

Si une commande génère plusieurs sorties, vous pouvez les réacheminer indépendamment les unes des autres. Ainsi, si cette commande envoie sa sortie standard vers le descripteur de fichier numéro 1, sa sortie erreur standard vers le descripteur de fichier numéro 2 et génère un fichier de données sur le descripteur de fichier numéro 9, saisissez la ligne de commande suivante pour réacheminer chacune de ces sorties vers un fichier différent :

```
command > standard 2> error 9> data
```

Réacheminement des sorties vers des documents entrée en ligne (here)

Si une commande se présente sous la forme suivante :

```
command << eofstring
```

et *eofstring* est une chaîne qui ne contient pas de métacaractères, alors le shell considère la ligne suivante comme entrée standard de la *commande* jusqu'à ce que le shell envoie une ligne comprenant uniquement *eofstring* (précédé si possible d'une ou plusieurs tabulations). Les lignes entre la première occurrence de *eofstring* et la seconde sont souvent appelées document *d'entrée en ligne* ou *document here*. Si un tiret (–) suit immédiatement les << caractères de réacheminement, le shell supprime les tabulations au début de chaque ligne du document **here** avant de passer la ligne à *command*.

Le shell crée un fichier temporaire contenant le document here et remplace les variables et les commandes dans le contenu avant de renvoyer le fichier à la commande. Il effectue le contrôle de forme sur les noms de fichiers qui font partie des lignes de commande dans les substitutions de commandes. Pour empêcher toutes substitutions, indiquez tous les caractères de *eofstring* :

```
command << \eofstring
```

Le document **Here** est particulièrement utile pour un faible volume de données, qu'il est plus judicieux de placer directement dans une procédure shell plutôt que dans un fichier distinct (scripts d'édition, par exemple). Par exemple, vous pouvez taper :

```
cat <<- xyz
    Ce message s'affiche sur
    l'écran ; les tabulations principales sont supprimées.
   xyz
```

Réacheminement des sorties à l'aide de canaux et filtres

Vous pouvez connecter deux ou plusieurs commandes pour que la sortie standard d'une commande soit utilisée comme entrée standard d'une autre commande. Les commandes connectées de telle sorte s'appellent une *pipeline*. La connexion qui accompagne les commandes s'appelle un *tube*. Les tubes sont utiles car ils permettent d'associer plusieurs commandes à but unique en une commande puissante.

Un pipeline permet d'acheminer la sortie à partir d'une commande pour devenir l'entrée d'une autre commande. Les commandes sont connectées par un symbole tube (|).

Lorsqu'une commande garde son entrée à partir d'une autre commande, la modifie et envoie les résultats à la sortie standard, elle s'appelle un *filtre*. Les filtres peuvent être utilisés seuls mais ils s'avèrent très utiles dans les pipelines. Les filtres les plus fréquents sont les suivants :

- sort

- plus
- pg

Voir les exemple ci–après :

- La commande **ls** affiche le contenu du répertoire courant en un flot ininterrompu. Si les informations ne tiennent pas sur un écran, certaines ne sont pas visibles. Pour obtenir un affichage page par page, chaînez les commandes **ls** et **pg**. Par exemple, tapez la commande suivante :

```
ls | pg
```

  Dans cet exemple, la sortie de la commande **ls** devient l'entrée de la commande **pg**. Pour afficher l'écran suivant, appuyez sur Entrée.

  Les pipelines fonctionnent dans une seule direction (de gauche à droite). Chaque commande est exécutée comme un process distinct, tous les process pouvant être exécutés simultanément. Un process est suspendu lorsqu'il n'a plus de données à lire ou que le canal vers le process suivant est plein.

- Les canaux peuvent également être utilisés avec la commande **grep**. La commande **grep** cherche dans un fichier les lignes qui contiennent des chaînes d'un format particulier. Pour afficher tous les fichiers créés ou modifiés en juillet, entrez :

```
ls -l | grep Jul
```

  Dans cet exemple, la sortie de la commande **ls** devient l'entrée de la commande **grep**.

Affichage de la sortie d'un programme avec copie dans un fichier (commande tee)

La commande **tee**, utilisée en combinaison avec une autre commande, lit l'entrée standard, consigne la sortie d'un programme vers la sortie standard et la copie simultanément dans le ou les fichiers spécifiés. La commande **tee** permet de visualiser immédiatement la sortie et de la conserver en vue d'utilisations ultérieures.

Par exemple, tapez la commande suivante :

```
ps -ef | tee program.ps
```

La sortie standard de la commande **ps –ef** s'affiche à l'écran et une copie est sauvegardée dans le fichier **program.ps**. Si le fichier **program.ps** existe déjà, il est supprimé et remplacé, sauf si l'option **noclobber** de la commande intégrée **set** est activée.

Pour visualiser et sauvegarder la sortie d'une commande dans un fichier existant, entrez :

```
ls -l | tee -a program.ls
```

La sortie standard de la commande **ls –l** s'affiche à l'écran et une copie est sauvegardée dans le fichier **program.ps**.

Le système affiche des informations similaires à l'exemple ci–après. Ces mêmes informations figurent dans le fichier **program.ls**.

```
-rw-rw-rw-   1 jones   staff   2301   Sep 19    08:53 161414
 -rw-rw-rw-   1 jones   staff   6317   Aug 31    13:17 def.rpt
 -rw-rw-rw-   1 jones   staff   5550   Sep 10    14:13 try.doc
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **tee** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Effacement de l'écran (commande clear)

Utilisez la commande **clear** pour supprimer de l'écran tous les messages et l'entrée clavier.

A l'invite, entrez ce qui suit :

```
clear
```

Le système efface l'écran et affiche l'invite.

Envoi d'un message vers la sortie standard (commande echo)

Utilisez la commande **echo** pour afficher les messages à l'écran.

Par exemple, pour inscrire un message sur la sortie standard, entrez, à l'invite :

```
echo Please insert diskette. . .
```

Le message suivant s'affiche :

```
Please insert diskette. . .
```

Pour combiner commande **echo** et caractères joker, entrez, à l'invite :

```
echo The back-up files are: *.bak
```

Le système affiche le message `Les fichiers de sauvegarde sont :` suivi des noms de fichier dans le répertoire courant terminé par `.bak`.

Ajout d'une ligne de texte à un fichier (commande echo)

Utilisez la commande **echo**, utilisée avec l'opérateur de réacheminement d'ajout, pour ajouter une ligne de texte à un fichier.

Par exemple, à l'invite, entrez :

```
echo Remember to backup mail files by end of week. >

 >notes
```

Ceci ajoute le message `Remember to backup mail files by end of week.` à la fin du fichier **notes**.

Copie de l'écran dans un fichier (commandes capture et script)

Utilisez la commande **capture** qui émule un terminal VT100, pour copier toutes les informations affichées sur votre terminal dans un fichier que vous spécifiez.

Utilisez la commande **script** pour copier toutes les informations affichées sur votre terminal, sans émulation du terminal VT100, dans un fichier que vous spécifiez.

Les deux commandes sont utiles pour imprimer les enregistrements des boîtes de dialogue affichées à l'écran.

Par exemple, pour prendre une capture d'écran avec une émulation d'un terminal VT100, entrez la commande suivante sur la ligne de commande :

```
capture screen.01
```

Le système affiche des informations similaires à l'exemple suivant :

```
Capture command is started. The file is screen.01.
Use ^P to dump screen to file screen.01.
You are now emulating a vt100 terminal.
Press Any Key to continue.
```

Après avoir saisi les données souhaitées et vidé le contenu de l'écran, mettez fin à la commande **capture** en appuyant sur les touches Ctrl–D ou en saisissant `exit`. Appuyez ensuite sur la touche Entrée. Le système affiche des informations similaires à l'exemple suivant :

```
Capture command is complete. The file is screen.01.
You are NO LONGER emulating a vt100 terminal.
```

La commande **cat** permet d'afficher le contenu de votre fichier.

Pour prendre une capture d'écran, saisissez la commande suivante sur la ligne de commande :

```
script
```

Le système affiche des informations similaires à l'exemple suivant :

```
Script command is started The file is typescript
```

Tout ce qui s'affiche sur cet écran est maintenant copié dans le fichier **typescript**.

Pour mettre fin à la commande **script**, appuyez sur `exit`, puis sur Entrée.
Le système affiche des informations similaires à l'exemple suivant :

```
Script command is complete. The file is typescript.
```

La commande **cat** permet d'afficher le contenu de votre fichier.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description des commandes **capture** et **script** dans le manuel *AIX 5L Version 5.3 Commands Reference.*

Affichage de texte en gros caractères (commande banner)

La commande **banner** affiche les caractères ASCII en gros caractères. Chaque ligne de sortie peut compter 10 chiffres (ou des caractères en majuscules ou en minuscules).

Par exemple, à l'invite, entrez :

```
banner GOODBYE!
```

Le système affiche `GOODBYE!` en gros caractères.

Récapitulatif des commandes relatives au réacheminement des entrées/sorties

>	Réacheminement des sorties standard, page 5-2
<	Réacheminement des entrées standard, page 5-4
> >	Réacheminement des sorties avec ajout à un fichier, page 5-3
|	Réacheminement des sorties à l'aide de canaux et filtres, page 5-5
banner	Permet d'envoyer des chaînes de caractères ASCII en gros caractères vers la sortie standard.
capture	Permet de vider le contenu de l'écran dans un fichier.
clear	Permet d'effacer l'écran du terminal.
echo	Permet d'envoyer des chaînes de caractères vers la sortie standard.
script	Permet de copier des sorties et entrées standard dans un fichier.
tee	Permet d'afficher la sortie standard d'un programme et d'en effectuer une copie dans un fichier.

Chapitre 6. Systèmes de fichiers et répertoires

Un *système de fichiers* est constitué d'un groupe de répertoires et des fichiers qu'ils contiennent. Il est généralement représenté par une arborescence : le répertoire racine, symbolisé par une barre oblique(/), définit un système de fichiers et apparaît au sommet de l'arborescence. Les branches descendantes, issues du répertoire racine, peuvent contenir des fichiers ou des sous–répertoires. Ces ramifications définissent des chemins d'accès uniques à tous les objets du système de fichiers.

Les fichiers sont regroupés dans des *répertoires*. Ces derniers sont rarement indépendants les uns des autres : les structurer en système de fichiers permet de les organiser.

Un *fichier* est un ensemble de données, susceptible d'être lu et dans lequel il est possible d'écrire. Un programme, un texte, des données, une unité, etc., constituent des fichiers. Commandes, imprimantes, terminaux, courrier et programmes d'application y sont stockés. L'accès à ces divers éléments est de ce fait uniformisé: le système y gagne en souplesse et en simplicité.

Ce chapitre traite des points suivants :

Système de fichiers

Un *système de fichiers* est une structure hiérarchique (arborescence de fichiers) de fichiers et de répertoires. Ce type de structure ressemble à un arbre renversé avec les racines au sommet et les branches en bas. Cette arborescence des fichiers organise les données et programmes en groupes, ce qui permet de manipuler simultanément plusieurs fichiers et répertoires.

Certaines tâches sont exécutées de façon plus efficace sur un système de fichiers que sur chaque répertoire d'un système de fichiers. Vous pouvez, par exemple, sauvegarder, déplacer ou sécuriser l'ensemble d'un système de fichiers.

Le système de fichiers de base s'appelle *système de fichiers journalisé (JFS)*. Ce système de fichiers utilise des techniques de journalisation de base de données pour maintenir la cohérence structurelle. Cela permet d'éviter d'endommager le système de fichiers lorsque le système s'arrête de manière anormale.

Nombre de tâches de gestion du système sont liées aux systèmes de fichiers, notamment :

- l'affectation d'espace aux systèmes de fichiers sur les volumes logiques,
- la création de systèmes de fichiers,
- la mise à la disposition des utilisateurs de l'espace du système de fichiers,
- le contrôle de l'utilisation de l'espace du système de fichiers,
- la sauvegarde des systèmes de fichiers pour protéger les données en cas de panne du système,
- le maintien de la cohérence des systèmes de fichiers.

Ces tâches incombent à l'administrateur du système.

Types de systèmes de fichiers

Prend en charge plusieurs types de systèmes de fichiers. Par exemple :

Système de fichiers journalisé (JFS)	Le type de base de système de fichiers accepte l'ensemble des commandes relatives aux systèmes de fichiers.
Système de fichiers journalisé amélioré (JFS2)	Le type de base de système de fichiers accepte l'ensemble des commandes relatives aux systèmes de fichiers.
Système de fichiers réseau	Type de système de fichiers permettant d'accéder aux fichiers résidant sur des machines distantes comme s'il s'agissaient de fichiers locaux.
Système de fichiers CD-ROM	Type de système de fichiers permettant d'accéder au contenu d'un CD-ROM via les interfaces habituelles du système (ouverture, lecture et fermeture).

Structure des systèmes de fichiers

Sur les systèmes autonomes, les systèmes de fichiers suivants résident, par défaut, sur les unités associées :

/ **Système de fichiers**	/ **Unité**
/ **dev** / **hd1**	/ **home**
/ **dev** / **hd2**	**/ usr**
/ **dev** / **hd3**	/ **tmp**
/ **dev** / **hd4**	/ **(racine)**
/ **dev** / **hd9var**	/ **var**
/ **proc**	/ **proc**
/ **dev** / **hd10opt**	/ **opt**

L'arborescence des fichiers présente les caractéristiques suivantes :

- Les fichiers partageables par les machines de la même architecture matérielle se trouvent dans le système de fichiers **/usr**.
- Les fichiers texte partageables, indépendants de l'architecture (pages de manuel, par exemple) se trouvent dans le répertoire **/var**.
- Le système de fichiers **/(root)** contient les fichiers et les répertoires essentiels à l'exploitation du système. Par exemple :
  - répertoire des unités (**/dev**) ;
  - points de montage des systèmes de fichiers sur le système de fichiers racine, **/mnt**, par exemple.
- Le système de fichiers **/home** est le point de montage des répertoires personnels de l'utilisateur.
- Pour les serveurs, le répertoire **/export** contient les fichiers de pagination d'espace, les systèmes de fichiers racine client (non partagés), les répertoires de vidage, personnels et **/usr/share** des clients sans disque, ainsi que les répertoires **/usr** exportés.
- Le système de fichiers **/proc** contient des informations sur l'état des processus et des unités d'exécution dans le système.
- Le système de fichiers **/opt** contient les logiciels en option, comme les applications.

La liste suivante propose des informations sur le contenu de certains sous–répertoires du système de fichiers **/ (racine)**.

/bin Lien symbolique au répertoire **/usr/bin**.

/dev Contient les noeuds unité pour les fichiers spéciaux des unités locales. Le répertoire **/dev** contient les fichiers spéciaux des unités de bande, des imprimantes, des partitions de disque et des terminaux.

/etc Contient les fichiers de configuration propres à chaque machine. Exemples :
- **/etc/hosts**
- **/etc/passwd**

/export Contient les répertoires et les fichiers d'un serveur, réservés aux clients distants.

/home	Sert de point de montage au système de fichiers contenant les répertoires personnels de l'utilisateur. Le système de fichiers **/home** contient les fichiers et les répertoires utilisateur. Sur les machines autonomes, un système de fichiers distinct est monté sur le répertoire **/home**. Dans un réseau, un serveur peut contenir des fichiers utilisateur qui doivent être accessibles par plusieurs machines. Dans ce cas, la copie serveur du répertoire **/home** est télémontée sur un système de fichiers **/home** local.
/lib	Lien symbolique au répertoire **/usr/lib**- qui comprend les bibliothèques indépendantes de l'architecture au format **lib*.a**.
/sbin	Contient les fichiers requis pour l'amorçage de la machine et le montage du système de fichiers **/usr**. La plupart des commandes utilisées au moment de l'amorçage proviennent du système de fichiers du disque RAM de l'image d'amorçage. De ce fait, très peu de commandes se trouvent dans le répertoire **/sbin**.
/tmp	Sert de point de montage du système de fichiers contenant les fichiers temporaires générés par le système.
/u	Lien symbolique au répertoire **/home**.
/usr	Sert de point de montage au système de fichiers contenant les fichiers invariables partageables par les machines (fichiers exécutables et documentation ASCII, par exemple). Les machines autonomes montent un système de fichiers local séparé sur le répertoire **/usr**. Les machines sans disque ou à faible capacité disque montent un répertoire, à partir d'un serveur distant, sur le système de fichiers **/usr**.
/var	Sert de point de montage des fichiers variables (dépendant de la machine). Le système de fichiers **/var** est configuré comme un système de fichiers dans la mesure où les fichiers qu'il contient tendent à grossir. Par exemple, c'est un lien symbolique au répertoire **/usr/tmp** qui contient les fichiers de travail temporaires.

Affichage de l'espace disponible sur un système de fichiers (commande df)

Utilisez la commande **df** pour afficher des informations sur l'espace total et l'espace disponible d'un système de fichiers. Le paramètre *FileSystem* spécifie le nom de l'unité où réside le système de fichiers, le répertoire sur lequel il est monté ou son chemin d'accès relatif. Si vous ne spécifiez pas le paramètre *FileSystem*, la commande **df** affiche des informations sur les systèmes de fichiers actuellement montés. Si vous indiquez un fichier ou un répertoire, la commande **df** affiche des informations sur le système de fichiers sur lequel réside le fichier.

Normalement, la commande **df** se sert des comptes libres du superbloc. Dans certaines circonstances exceptionnelles, ces comptes peuvent être en erreur. C'est ce qui se produit si un système de fichiers est activement modifié pendant l'exécution de la commande **df**, par exemple.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **df** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Remarque : Dans certains systèmes de fichiers éloignés, tels que les NFS (Network File Systems), les colonnes représentant l'espace disponible sur l'écran restent vierges si le serveur ne fournit pas l'information demandée.

Les exemples ci–après illustrent les différentes utilisations de la commande **df** :

- Pour afficher les informations relatives à tous les systèmes de fichiers montés, entrez :

```
df
```

  Si votre système est configuré de sorte que les répertoires **/**, **/usr**, **/site** et **/usr/venus** résident sur des systèmes de fichiers distincts, la commande **df** génère un résultat semblable au suivant :

```
Filesystem 512-blocks  free   %used   Iused %Iused   Mounted on
/dev/hd4    20480     13780    32%      805   13%     /
/dev/hd2   385024     15772    95%    27715   28%     /usr
/dev/hd9var 40960     38988     4%      115    1%     /var
/dev/hd3    20480     18972     7%       81    1%     /tmp
/dev/hd1     4096      3724     9%       44    4%     /home
```

- Pour afficher l'espace disponible du système de fichiers dans lequel réside le répertoire en cours, saisissez :

```
df.
```

Répertoires : généralités

Un *répertoire* est un fichier qui ne contient que les informations d'accès à d'autres fichiers et répertoires. Par conséquent, il occupe moins d'espace que les autres types de fichiers. Les répertoires vous permettent de regrouper des fichiers et d'autres répertoires dans une hiérarchie modulaire, ce qui confère à la structure de systèmes de fichiers à la fois flexibilité et profondeur.

Les répertoires contiennent des entrées. Chacune d'entre elles contient le nom et le numéro d'*i–node* (index node) du fichier. Pour optimiser l'utilisation de l'espace disque, les données d'un fichier sont réparties sur plusieurs zones de la mémoire. L'i–node indique les adresses et l'enchaînement de tous les blocs de données associés à un fichier. L'i–node enregistre également d'autres informations, comme la date et l'heure de la dernière ouverture/modification, les droits d'accès, le nombre de liens, le propriétaire, le type de fichier, etc.

Un ensemble spécial de commandes contrôle les répertoires. Vous pouvez, via la commande **ln**, associer plusieurs noms d'un fichier à un même i–node.

Les répertoires contenant souvent des informations réservées à certains utilisateurs, l'accès aux répertoires peut être protégé. Vous pouvez, en définissant des droits d'accès, contrôler et limiter l'accès aux répertoires, et désigner les utilisateurs autorisés à en modifier le contenu. Pour plus d'informations, reportez–vous à la section Modes d'accès aux fichiers et aux répertoires, page 10-4.

Ce chapitre traite des points suivants :

Types de répertoires

Les répertoires peuvent être définis par le système d'exploitation, l'administrateur ou les utilisateurs. Les répertoires système contiennent des fichiers système particuliers (commandes, par exemple). Le répertoire **/(root)**, défini par le système, constitue le sommet de la hiérarchie du système de fichiers. Il est en général composé des répertoires suivants :

/dev	Fichiers spéciaux pour unités d'E–S.
/etc	Fichiers d'initialisation et d'administration du système.
/home	Répertoires de connexion des utilisateurs du système.
/tmp	Fichiers temporaires, supprimés à l'issue d'un nombre de jours donné.
/usr	Répertoires **lpp, include** et programmes système.
/usr/bin	Programmes exécutables utilisateur.

Certains répertoires, comme le répertoire de connexion ou le répertoire personnel (**$HOME**), sont définis et personnalisés par l'administrateur. Lorsque vous vous connectez au système, vous vous trouvez dans le répertoire de connexion (qui est alors le répertoire courant).

Les répertoires que vous créez sont appelés des répertoires utilisateur. Ces répertoires vous permettent d'organiser et de maintenir vos fichiers.

Organisation des répertoires

Les répertoires contiennent des fichiers, des sous–répertoires ou une combinaison des deux. Un sous–répertoire est un répertoire qui se trouve dans un autre répertoire. Le répertoire contenant le sous–répertoire s'appelle le *répertoire parent*.

Chaque répertoire possède une entrée pour le répertoire parent dans lequel il a été créé, .. (point point), et une entrée pour le répertoire lui–même, . (point). Dans la plupart des listes de répertoires, ces fichiers sont cachés.

Arborescence de répertoire
: La structure d'un système de fichiers peut rapidement devenir complexe. Essayez de ne pas la surcharger. Veillez en outre à donner aux fichiers et aux répertoires des noms évocateurs. Votre travail en sera facilité.

Répertoire parent
: Chaque répertoire, à l'exception de **/(racine)**, possède un répertoire parent et peut avoir des répertoires enfants.

Répertoire personnel
: Lorsque vous vous connectez, vous vous trouvez dans votre *répertoire personnel* (répertoire de connexion). Il est défini par l'administrateur pour chaque utilisateur. C'est dans ce répertoire que vous conservez vos fichiers. Normalement, les répertoires que vous créez pour votre usage personnel sont des sous–répertoires de votre répertoire personnel. Pour revenir à tout moment à votre répertoire personnel, entrez la commande **cd** et appuyez sur Entrée à l'invite.

Répertoire de travail
: Vous travaillez toujours dans le cadre d'un répertoire. Ce répertoire, quel qu'il soit, est appelé répertoire *courant* ou *de travail*. La commande **pwd** (present working directory) affiche le nom du répertoire courant. Utilisez la commande **cd** pour changer les répertoires de travail.

Conventions d'appellation des répertoires

Le nom d'un répertoire doit être unique dans le répertoire qui le contient. Ainsi, à chaque répertoire ne correspond qu'un seul chemin d'accès dans le système de fichiers. Les noms de répertoires suivent les mêmes conventions que les fichiers, explicitées à la section Conventions d'appellation des fichiers, page 7-4.

Abréviations des répertoires

Les abréviations sont un moyen pratique de spécifier certains répertoires. En voici la liste :

Abréviation	Signification
.	Répertoire de travail courant.
..	Répertoire au–dessus (parent) du répertoire de travail courant.
~	Votre répertoire personnel. (Ceci n'est pas vrai pour le shell Bourne. Pour plus d'informations, reportez–vous à la section Shell Bourne : généralités, page 12-70.)
$HOME	Votre répertoire personnel. (Ceci est vrai pour tous les shells.)

Chemins d’accès au répertoire

L’accès à chaque fichier ou répertoire d’un système de fichiers se fait par un chemin unique, appelé *chemin d’accès*, qui en indique l’emplacement. Le chemin d’accès spécifie l’emplacement d’un répertoire ou d’un fichier dans le système de fichiers.

Remarque : Les chemins d’accès ne peuvent dépasser 1023 caractères.

Les types de chemins d’accès suivants sont associés au système de fichiers :

chemin d’accès absolu	chemin à partir du répertoire **/(root)**. Les chemins d’accès absolus commencent toujours par le signe / (barre oblique).
chemin d’accès relatif	chemin à partir du répertoire courant vers les sous–répertoires et fichiers en aval.

Un chemin d’accès absolu est constitué de la liste complète des noms de répertoire ou de fichier situés en amont, à partir du répertoire **/(root)**. Quel que soit l’endroit où vous vous trouvez dans le système de fichiers, vous pouvez atteindre un répertoire ou un fichier en spécifiant son chemin d’accès absolu. Ce chemin doit obligatoirement commencer par le signe / (barre oblique), symbole du répertoire racine. Le chemin d’accès **/A/D/9** est le chemin absolu vers **9**. La première **/** (barre oblique) représente le répertoire **/(root)**, point de départ de la recherche. Le reste du chemin oriente la recherche vers **A**, puis **D** et enfin **9**.

Il peut y avoir deux fichiers nommés **9** car le chemin d’accès absolu donne à chaque fichier un nom unique à l’intérieur du système de fichiers. Les chemins **/A/D/9** et **/C/E/G/9** spécifient deux fichiers distincts appelés **(9)**.

En revanche, les chemins d’accès relatifs indiquent l’accès à un répertoire ou à un fichier basé sur le répertoire de travail courant. Pour ce type de chemin d’accès, vous pouvez utiliser la notation .. (point point) pour remonter dans la hiérarchie du système de fichiers. Ces deux points successifs représentent le répertoire parent. De ce fait, les chemins d’accès relatifs ne commencent pas par le signe/ (barre oblique). Ils donnent accès à un fichier du répertoire courant ou à un fichier ou répertoire situé en amont ou en aval du répertoire courant dans le système de fichiers. Si **D** est le répertoire actuel, le nom de chemin relatif pour accéder à **10** est **F/10**. Néanmoins, le chemin d’accès absolu est toujours **/A/D/F/10**. De la même manière, le chemin d’accès relatif pour accéder à **3** est**../../B/3**.

Vous pouvez également utiliser la notation . (point). La notation par point est couramment employée pour exécuter des programmes qui doivent lire le nom du répertoire courant.

Procédures de gestion des répertoires

Vous pouvez utiliser les répertoires et leur contenu de plusieurs façons.

Cette section présente les différentes commandes, avec des exemples:

Création de répertoires (commande mkdir)

Utilisez la commande **mkdir** pour créer un ou plusieurs répertoires définis par le paramètre *Répertoire*. Chaque répertoire ainsi créé contient les entrées standard (.) (point) et (..) (point point). Vous pouvez définir des droits d'accès au répertoire via l'indicateur **–m** *mode*.

Le répertoire est créé dans le répertoire en cours ou le répertoire de travail, à moins que vous ne spécifiez un nom de chemin d'accès absolu pour désigner un autre emplacement du système de fichiers.

Les exemples ci–après illustrent les différentes utilisations de la commande **mkdir** :

- Pour créer un répertoire appelé **Test** dans le répertoire de travail en cours et disposant des droits d'accès par défaut, entrez la commande :

```
mkdir Test
```

- Pour créer un répertoire appelé **Test** doté des droits `rwxr–xr–x` dans le répertoire **/home/demo/sub1** créé au préalable, entrez :

```
mkdir –m 755 /home/demo/sub1/Test
```

- Pour créer un répertoire appelé **Test** doté des droits d'accès par défaut, dans le répertoire **/home/demo/sub2**, entrez :

```
mkdir –p /home/demo/sub2/Test
```

  L'indicateur **–p** crée les répertoires **/home**, **/home/demo** et **/home/demo/sub2** s'ils n'existent pas.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **mkdir** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Déplacement ou changement de nom d'un répertoire (commande mvdir)

Pour déplacer ou renommer un répertoire, utilisez la commande **mvdir**.

Les exemples ci–après illustrent les différentes utilisations de la commande **mkdir** :

- Pour déplacer un répertoire, entrez :

```
mvdir book manual
```

  Le répertoire **book** est déplacé dans le répertoire **manual**, si celui-ci existe. Sinon, le répertoire **book** est renommé en **manuel**.

- Pour déplacer et renommer un répertoire, entrez :

```
mvdir book3 proj4/manual
```

Le répertoire **book3** est déplacé vers le répertoire **proj4** qui est renommé en **manual** (si ce nom de répertoire n'est pas déjà utilisé).

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **mvdir** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage du répertoire courant (commande pwd)

La commande **pwd** écrit, sur l'unité de sortie standard, le chemin d'accès absolu du répertoire courant (en commençant par le répertoire **/(racine)**). Tous les répertoires sont séparés par une barre oblique (/). Le répertoire **/(racine)** est représenté par la première barre /, le dernier répertoire indiqué étant le répertoire courant.

Par exemple, pour afficher le répertoire en cours, entrez :

```
pwd
```

Le nom du chemin d'accès complet du répertoire en cours s'affiche de la manière suivante :

```
/home/thomas
```

Changement de répertoire (commande cd)

Utilisez la commande **cd** pour passer du répertoire actuel à un autre répertoire. Pour cela, vous devez disposer des droits d'exécution (recherche) dans le répertoire spécifié.

Si vous omettez de définir le paramètre *Répertoire*, la commande **cd** vous ramène à votre répertoire de connexion (**$HOME** dans les environnements **ksh** et **bsh** ou **$home** dans l'environnement **csh**). Si le nom du répertoire spécifié est un nom de chemin complet, il devient le répertoire en cours. Un chemin d'accès complet commence par une barre oblique (/) indiquant le répertoire **/(racine)**, par un point (.) indiquant le répertoire courant ou par point point (..) indiquant le répertoire parent. Si le nom du répertoire n'est pas un nom de chemin entier, la commande **cd** le recherche en fonction d'un des chemins précisés dans la variable shell **$CDPATH** (ou la variable **$cdpath csh**). Cette variable possède la même syntaxe et sensiblement la même valeur sémantique que la variable shell **$PATH** (ou la variable **$path csh**).

Les exemples ci–après illustrent les différentes utilisations de la commande **cd** :

- Pour accéder à votre répertoire personnel, entrez :

```
cd
```

- Pour passer au répertoire **/usr/include**, entrez :

```
cd /usr/include
```

- Pour passer au sous–répertoire **sys**, entrez :

```
cd sys
```

  Si le répertoire courant **/usr/include** contient le sous–répertoire **sys**, alors **/usr/include/sys** devient le répertoire courant.

- Pour remonter d'un niveau dans l'arborescence, entrez :

```
cd ..
```

  Le nom de fichier spécial, point point (..), désigne le répertoire se trouvant immédiatement au–dessus du répertoire en cours, à savoir son répertoire parent.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **cd** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Copie de fichiers (commande cp)

Utilisez la commande **cp** pour copier le contenu du fichier ou du répertoire spécifié dans les paramètres *FichierSource* ou *RépertoireSource* dans le fichier ou dans le répertoire spécifié par les paramètres *FichierCible* ou *RépertoireCcible*. Si le fichier défini comme *FichierCible* existe, son contenu est écrasé lors de la copie. Pour copier plusieurs *fichiers source*, la cible doit être un répertoire.

Pour copier le *FichierSource* dans un répertoire, indiquez, dans le paramètre *RépertoireCible*, le chemin d'accès à ce répertoire. Les fichiers conservent leurs noms respectifs lorsqu'ils sont copiés dans un répertoire, à moins que vous n'indiquiez un nouveau nom de fichier à la fin du chemin. Si vous lui associez l'indicateur **–r** ou **–R**, la commande **cp** copie également des répertoires entiers dans d'autres répertoires.

Les exemples ci–après illustrent les différentes utilisations de la commande **cp** :

- Pour copier tous les fichiers du répertoire **/home/accounts/customers/orders** dans le répertoire cible **/home/accounts/customers/shipments**, entrez la commande :

```
cp /home/accounts/customers/orders/* /home/accounts/customers/shipments
```

  Cette commande copie les fichiers, mais pas les répertoires, du répertoire **orders** dans le répertoire **shipments**.

- Pour copier un répertoire, ainsi que tous ses fichiers et sous–répertoires, dans un autre répertoire, saisissez la commande suivante :

```
cp –R /home/accounts/customers /home/accounts/vendors
```

  Le répertoire **customers**, avec ses fichiers et ses sous–répertoires (et les fichiers que ces derniers contiennent), est copié dans le répertoire **vendors**.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **cp** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage du contenu d'un répertoire (commande ls)

Utilisez la commande **ls** pour afficher le contenu d'un répertoire.

La commande **ls** écrit, sur l'unité de sortie standard, le contenu des *Répertoires* ou le nom des *Fichiers* spécifiés, ainsi que les informations demandées via les indicateurs. A défaut de paramètre *Fichier* ou *Répertoire*, la commande **ls** affiche le contenu du répertoire courant.

Par défaut, la commande **ls** affiche toutes les informations alphabétiquement par nom de fichier. Si la commande est lancée par un utilisateur racine, l'indicateur par défaut est **–A** et affiche toutes les entrées sauf `.` (point) et `..` (point point). Pour afficher toutes les entrées des fichiers, y compris celles qui commencent par un point (`.`), utilisez la commande **ls –a**.

Vous pouvez formater la sortie de plusieurs façons :

- une entrée par ligne, avec l'indicateur**–l** ;
- plusieurs entrées dans plusieurs colonnes, avec l'indicateur **–C** ou **–x**. L'indicateur **–C** est le format par défaut quand la sortie se fait vers un terminal tty.
- des entrées séparées par des virgules, avec l'indicateur **–m**.

Pour déterminer le nombre de caractères par ligne sur l'unité de sortie, la commande **ls** utilise la variable d'environnement **$COLUMNS**. Si cette variable n'est pas définie, la commande lit le fichier **terminfo**. Si aucune de ces méthodes ne donne de résultat, elle adopte par défaut la valeur 80.

Les informations affichées avec les indicateurs **–e** et **–l** sont interprétées comme suit :

Le premier caractère peut être :

d l'entrée est un répertoire.

b l'entrée est un fichier bloc spécial.

c l'entrée est un fichier caractères spécial.

l l'entrée est un lien symbolique.

c l'entrée est un fichier FIFO (first in, first out) spécial.

s l'entrée est un socket local.

– l'entrée est un fichier ordinaire.

Les neuf caractères suivants sont répartis en trois ensembles de trois caractères chacun. Les trois premiers caractères indiquent les droits du propriétaire. Les trois caractères suivants indiquent les droits des autres utilisateurs du groupe. Les trois derniers caractères indiquent les droits de toute autre personne ayant accès au fichier. Chaque caractère d'un groupe de 3 correspond aux droits d'accès en lecteur, écriture et exécution. Le droit d'exécution sur un répertoire confère le droit d'y rechercher un fichier donné.

Les droits sont indiqués comme suit :

r accès en lecture.

t seul le propriétaire du répertoire ou du fichier peut supprimer ou renommer un fichier dans ce répertoire, même si les autres utilisateurs ont un droit d'écriture sur ce répertoire.

w accès en écriture (édition).

x accès en exécution (recherche).

– accès correspondant refusé.

Les informations affichées avec l'indicateur **–e** sont identiques à celles affichées avec l'indicateur **–l**, à l'exception de l'ajout du 11ème caractère suivant :

+ Il indique un fichier doté d'informations de sécurité étendues. Par exemple, le fichier peut comporter les attributs **ACL**, **TCB** ou **TP** dans le mode.

– Il indique un fichier non doté d'informations de sécurité étendues.

Lorsque la taille des fichiers d'un répertoire est affichée, la commande **ls** affiche le décompte total de blocs, blocs indirects compris.

Voir les exemple ci–après :

- Pour afficher la liste de tous les fichiers du répertoire courant, entrez :

```
ls –a
```

  Cette liste comporte tous les fichiers, y compris : les fichiers .

  – (point),

  – les fichiers .. (point point),

  – les autres fichiers commençant ou non par un . (point).

- Pour afficher des informations détaillées, entrez :

```
ls–l chap1.profile
```

  Une liste exhaustive fournissant des informations complètes sur **chap1** et **.profile** s'affiche.

- Pour afficher des informations détaillées sur un répertoire, entrez :

```
ls –d –l . manual manual/chap1
```

  Une liste exhaustive concernant les répertoires **.** (point) et **manual** s'affiche avec le fichier **manual/chap1**. Sans indicateur **–d**, la commande afficherait la liste des fichiers des répertoires **.** (point) et **manual** et non des informations complètes sur les répertoires eux–mêmes.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **ls** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Suppression ou retrait d'un répertoire (commande rmdir)

La commande **rmdir** supprime du système le répertoire spécifié par le paramètre *Répertoire*. Pour pouvoir être retiré, ce répertoire doit être vide (il ne peut contenir que . et ..) et vous devez disposer des droits en écriture sur le répertoire parent correspondant. Utilisez la commande **ls –a** *Répertoire* pour vérifier s'il est vide.

Les exemples ci–après illustrent les différentes utilisations de la commande **rmdir** :

- Pour vider et supprimer un répertoire, entrez :

```
rm mydir/* mydir/.*
rmdir mydir
```

  Le contenu de **mydir** est supprimé, puis le répertoire lui–même. La commande **rm** affiche un message d'erreur concernant la suppression des répertoires . (point) et .. (point point). La commande **rmdir** supprime ensuite le répertoire.

  Remarque : La commande **rm mydir/* mydir/.*** supprime d'abord les fichiers dont le nom ne commence pas par un point, puis ceux dont le nom commence par un point. La commande **ls** n'affiche pas la liste des fichiers commençant par un point sauf si vous utilisez l'indicateur **–a**.

- Pour supprimer le répertoire **/tmp/jones/demo/mydir** et les sous–répertoires qu'il contient, entrez :

```
cd /tmp
rmdir –p jones/demo/mydir
```

  Le répertoire **jones/demo/mydir** est supprimé du répertoire **/tmp**. Si un répertoire n'est pas vide et que vous ne disposez pas des droits en écriture sur celui–ci lorsqu'il est supprimé, la commande génère les messages d'erreur appropriés.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **mkdir** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Comparaison de répertoires (commande dircmp)

La commande **dircmp** compare les deux répertoires spécifiés par *Répertoire1* et *Répertoire2* et écrit des informations relatives à leur contenu sur l'unité de sortie standard. Elle compare d'abord les noms des fichiers de chaque répertoire. Lorsqu'elle rencontre deux fichiers portant le même nom, elle compare leur contenu.

Dans la sortie, la commande **dircmp** répertorie les fichiers uniques figurant dans chaque répertoire. Elle dresse ensuite la liste des fichiers portant des noms identiques dans les deux répertoires, mais ayant un contenu différent. Si aucune option n'est spécifiée, la commande répertorie également les fichiers aux contenus et aux noms identiques dans les deux répertoires.

Les exemples ci–après illustrent les différentes utilisations de la commande **dircmp** :

- Pour récapituler les différences entre les fichiers des répertoires **proj.ver1** et **proj.ver2**, entrez :

```
dircmp proj.ver1 proj.ver2
```

  Le récapitulatif généré liste les différences entre les répertoires **proj.ver1** et **proj.ver2**. La synthèse répertorie séparément les fichiers figurant dans un répertoire, puis dans l'autre et enfin ceux figurant dans les deux répertoires. Ces derniers sont en outre différenciés selon que leur contenu est identique ou non.

- Pour afficher en détail les différences entre les fichiers des répertoires **proj.ver1** et **proj.ver2**, entrez :

```
dircmp -d -s proj.ver1 proj.ver2
```

  L'indicateur **–s** supprime les informations relatives à des fichiers identiques. L'indicateur **–d** affiche les fichiers différents trouvés dans les deux répertoires dans la liste **diff**.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **dircmp** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Récapitulatif des commandes pour les Systèmes de fichiers et les répertoires

Système de fichiers

df	Donne des informations sur l'espace disponible dans les systèmes de fichiers.

Abréviations des répertoires

.	Répertoire de travail courant
..	Répertoire "au–dessus" (parent) du répertoire de travail courant.
~	Votre répertoire personnel. (Ceci n'est pas vrai pour le shell Bourne. Pour plus d'informations, reportez–vous à la section Shell Bourne : généralités, page 12-70.)
$HOME	Répertoire personnel (valable pour tous les shells).

Procédures de gestion des répertoires

cd	Passe à un autre répertoire.
cp	Copie des fichiers ou des répertoires.
dircmp	Compare deux répertoires, ainsi que le contenu de leurs fichiers communs.
ls	Affiche le contenu d'un répertoire.
mkdir	Crée un ou plusieurs répertoires.
mvdir	Déplace (ou renomme) un répertoire.
pwd	Affiche le chemin d'accès au répertoire de travail.
rmdir	Supprime un répertoire.

Chapitre 7. Fichiers

Dans le système d'exploitation, toutes les entrées/sorties (E/S) d'informations passent par des fichiers. Ceci permet de standardiser l'accès aux logiciels et au matériel. On parle d'entrée lorsqu'il y a modification ou écriture dans le contenu d'un fichier. On parle de sortie lorsque le contenu d'un fichier est lu ou transféré vers un autre fichier. Par exemple, lorsque qu'une sortie papier d'un fichier est demandée, le système lit le contenu du fichier et le copie dans le fichier représentant l'imprimante.

Ce chapitre traite des points suivants :

Types de fichiers

Voici les types de fichiers de base :

standard	fichier de données (texte, binaire ou exécutable)
répertoire	fichier contenant des informations permettant d'accéder à d'autres fichiers
spécial	fichier de chaînage FIFO (first-in, first-out) ou unité physique

Les types de fichier reconnus par le système relèvent obligatoirement d'une de ces trois catégories. Il existe cependant à l'intérieur de ces catégories de nombreuses variantes.

Fichiers standard

Les fichiers standard sont les fichiers les plus courants, ils sont utilisés pour stocker des données. Les fichiers standard sont de la même forme que les fichiers texte ou binaires :

Fichiers texte

Les fichiers texte sont des fichiers réguliers stockés dans le format texte ASCII et lisibles par l'utilisateur. Vous pouvez les afficher et les imprimer. Les lignes d'un fichier texte ne doivent pas comporter de caractères **NUL**, et ne doivent pas excéder la longueur de **{LINE_MAX}** octets, caractère de nouvelle ligne compris.

Le terme **fichier texte** n'interdit pas la présence de caractères de contrôle et autres caractères non imprimables (autres que **NUL**). En conséquence, les utilitaires standard d'affichage des fichiers texte, en entrée ou en sortie, soit sont capables de traiter ces caractères spéciaux, soit décrivent explicitement les limites de leur action.

Fichiers binaires

Les fichiers binaires sont des fichiers standard, interprétés par le système. Les fichiers binaires peuvent être des fichiers exécutables donnant des instructions au système. Les commandes et les programmes sont stockées dans des fichiers binaires. La conversion textes ASCII/codes binaires est effectuée par des compilateurs.

Les fichiers texte et binaire diffèrent sur un unique point, les fichiers texte contiennent des lignes de longueur inférieure à **{LINE_MAX}** octets, pas de caractères **NUL** et chaque ligne se termine par un caractère de nouvelle ligne.

Fichiers répertoire

Les fichiers répertoire comportent des informations que le système requiert pour accéder à tous les types de fichiers, mais ils ne contiennent pas de données sur le fichier en cours. Par conséquent, il occupe moins d'espace que les autres types de fichiers et confère au système de fichiers souplesse et puissance. Chaque entrée de répertoire représente un fichier ou un sous–répertoire. Chaque entrée comporte le nom du fichier et le numéro de référence du nœud d'index du fichier (i–node number). Le numéro d'i–node désigne l'unique nœud index attribué au fichier. Le numéro d'i–node décrit l'emplacement des données associées au fichier. Les répertoires sont créés et contrôlés par un ensemble de commandes séparées.

Fichiers spéciaux

Les fichiers spéciaux définissent les unités pour le système, ou sont des fichiers temporaires générés par les process. Les types de fichiers spéciaux de base sont des fichiers FIFO (premier entré, premier sorti), bloc ou caractères. Les fichiers FIFO sont également appelés fichiers de *chaînage*. Les fichiers de chaînage sont créés par un process afin de permettre une communication temporaire avec un autre process. Ces fichiers cessent d'exister lorsque le premier process se termine. Les fichiers bloc et caractères définissent des unités.

A chaque fichier sont associés des droits d'accès (dits *modes d'accès*), déterminant les utilisateurs habilités à le lire, le modifier ou l'exécuter.

Reportez-vous à "Modes d'accès aux fichiers et aux répertoires", page10-4.

Conventions d'appellation des fichiers

Il ne peut y avoir, dans un même répertoire, deux fichiers enregistrés sous le même nom. Ainsi, à chaque fichier ne correspond qu'un seul chemin d'accès dans le système de fichiers. Pour nommer un fichier, voici quelques règles à respecter :

- Le nom ne peut excéder 255 caractères (lettres, chiffres et traits de soulignement).
- Le système d'exploitation respecte la disctinction entre les minuscules et les majuscules dans les noms de fichiers. Par conséquent, `FICHEA`, `FiChea` et `fichea` sont trois noms de fichiers distincts, même s'ils se trouvent dans le même répertoire.
- Choisissez des noms "parlants".
- Les noms de répertoires doivent respecter les mêmes conventions.
- Certaines caractères ont une signification spéciale pour le système d'exploitation. Evitez de les utiliser dans les noms de vos fichiers. En voici une liste (non exhaustive) :

```
/ \ " ' * ; – ? [ ] ( ) ~ ! $ { } < > # @ & |
```

- Un nom de fichier est caché de la liste normale d'un répertoire s'il commence par un point (.). Lorsque la commande **ls** est entrée avec l'indicateur **–a**, les fichiers cachés sont répertoriés avec les autres fichiers et répertoires.

Chemins d'accès aux fichiers

Le chemin d'accès à un fichier ou à un répertoire est constitué de la liste des répertoires situés en amont dans l'arborescence des répertoires.

Tous les chemins étant issus du répertoire racine /(**root**), chaque fichier est associé de façon unique à ce répertoire, par un *nom* de *chemin absolu*. Un chemin d'accès absolu commence toujours par le symbole / (barre oblique). Par exemple, le nom de chemin absolu du fichier **h** pourrait être **/B/C/h**. Veuillez noter que deux fichiers de nom **h** peuvent exister dans le système. Comme les chemins absolus des deux fichiers sont différents **/B/h** et **/B/C/h**, chaque fichier **h** possède un nom unique dans le système. Chaque composant d'un nom de chemin d'accès est un répertoire, excepté le dernier élément. Le dernier élément du nom de chemin d'accès peut être un nom de fichier.

Remarque : Les chemins d'accès ne peuvent dépasser 1023 caractères.

Concordance avec un modèle à l'aide de caractères de remplacement et de métacaractères

Les caractères de remplacement constituent une méthode pratique pour spécifier de multiples noms de fichiers ou répertoires. Les deux caractères de remplacement sont l'astérisque (`*`) et le point d'interrogation (`?`). Les métacaractères sont les crochets ouverts et fermés (`[ ]`), le tiret (`–`) et le point d'exclamation (`!`).

Recherche de concordances avec le caractère de remplacement *

L'astérisque (`*`) remplace une séquence ou une chaîne quelconque de caractères. L'astérisque (`*`) désigne tout caractère, y compris aucun caractère.

Voir les exemple ci–après :

- Si, par exemple, votre répertoire contient les fichiers :

```
1test 2test afile1 afile2 bfile1 file file1 file10 file2 file3
```

  et que vous souhaitez seulement vous reporter aux fichiers commençant par **file**, spécifiez :

```
file*
```

  La sélection portera sur les fichiers : **file file1 file10 file2 file3**.

- Pour rechercher les fichiers contenant le mot **file**, spécifiez :

```
*file*
```

  La sélection portera sur les fichiers : **afile1 afile2 bfile1 file file1 file10 file2 file3**.

Recherche de concordances avec le caractère de remplacement ?

Utilisez le point d'interrogation `?` pour remplacer un caractère quelconque. Le caractère `?` indique n'importe quel caractère unique.

Voir les exemple ci–après :

- Pour rechercher uniquement les fichiers commençant par **file**, suivis d'un seul caractère, spécifiez :

```
file?
```

  La sélection portera sur les fichiers : **file1 file2 file3**.

- Pour rechercher uniquement les fichiers commençant par **file**, suivis de deux caractères, spécifiez :

```
file??
```

  La sélection portera sur le fichier : **file10**.

Recherche de concordances avec les métacaractères shell [ ]

Les métacaractères sont une alternative aux caractères de remplacement. Il s'agit d'encadrer les caractères souhaités par des crochets `[ ]`. La méthode est semblable à l'utilisation de `?`, mais là, vous précisez les caractères qui doivent concorder. Vous pouvez aussi, à l'aide d'un tiret (`–`), spécifier une plage de valeurs. Pour spécifier toutes les lettres de l'alphabet, indiquez `[[:alpha:]]`. Pour spécifier toutes les lettres minuscules de l'alphabet, indiquez `[[:lower:]]`.

Voir les exemple ci–après :

- Pour rechercher uniquement les fichiers se terminant par `1` ou `2`, spécifiez :

```
*file[12]
```

  La sélection portera sur les fichiers : **afile1 afile2 file1 file2**.

- Pour référencer uniquement les fichiers commençant par un chiffre, spécifiez :

```
[0123456789]*   ou   [0-9]*
```

  La sélection portera sur les fichiers : **1test 2test**.

- Pour rechercher uniquement les fichiers ne commençant pas par un `a`, spécifiez :

```
[!a]*
```

  La sélection portera sur les fichiers : **1test 2test bfile1 file file1 file10 file2 file3**.

Concordance avec un modèle et expressions régulières

Les expressions régulières permettent de sélectionner des chaînes dans un ensemble spécifique de chaînes de caractères. Leur usage est généralement associé à un traitement de texte.

Les expressions régulières peuvent représenter un large éventail de chaînes possibles. Alors que ces expressions peuvent être diversement interprétées selon l'environnement local, les fonctions d'internationalisation assurent l'invariance du contexte au niveau des différents environnements locaux.

Voir les exemples dans les comparaisons suivantes :

Concordance avec un modèle	**Expression régulière**
*	.*
?	.
[!a]	[^a]
[abc]	[abc]
[[:alpha:]]	[[:alpha:]]

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **awk** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Procédures de gestion de fichier

Travailler sur des fichiers couvre de multiples tâches. Le plus souvent, vous créez un fichier texte à l'aide d'un éditeur. Les éditeurs les plus courants dans l'environnement UNIX sont les éditeurs vi et ed. Etant donné que plusieurs éditeurs de texte sont disponibles, choisissez celui qui vous convient le mieux.

Vous pouvez également créer des fichiers en réacheminant des entrées/sorties, comme décrit dans Réacheminement des entrées/sorties, page 5-1. La sortie d'une commande peut ainsi être envoyée dans un nouveau fichier ou ajoutée à un fichier existant.

Les fichiers créés et modifiés peuvent ensuite être copiés dans un autre répertoire et, renommés pour en distinguer les différentes versions. Vous serez sans doute également amené à créer des répertoires si vous travaillez sur plusieurs projets.

Il vous faudra aussi supprimer certains fichiers. Votre répertoire peut rapidement être encombré de fichiers contenant des informations inutiles ou anciennes. Pour libérer de l'espace sur votre système, assurez vous d'effacer les fichiers qui ne sont plus nécessaires.

Ce chapitre traite des points suivants :

Suppression de fichiers (commande rm)

Utilisez la commande **rm** pour supprimer des fichiers. La commande **rm** supprime les entrées du fichier spécifié, d'un groupe de fichiers ou de certains fichiers sélectionnés dans un répertoire. Pour utiliser la commande **rm**, il n'est pas nécessaire de disposer des droits d'accès en écriture et en lecture sur le fichier et aucune confirmation de l'utilisateur n'est demandée. Cependant, vous devez disposer des droits d'accès en écriture sur le répertoire contenant le fichier.

Les exemples ci–après illustrent les différentes utilisations de la commande **rm** :

- Pour supprimer le fichier nommé **myfile**, entrez :

```
rm myfile
```

- Pour supprimer, un par un, tous les fichiers du répertoire **mydir**, entrez :

```
rm –i mydir/*
```

Lorsque chaque nom de fichier apparaît, saisissez y et appuyez sur Entrée pour supprimer le fichier. Ou, pour conserver le fichier, appuyez seulement sur la touche Entrée.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **rm** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Déplacement et changement de nom d'un fichier (commande mv)

Utilisez la commande **mv** pour déplacer les fichiers et répertoires d'un répertoire dans un autre ou renommer un fichier ou un répertoire. Si vous déplacez un fichier ou un répertoire dans un nouveau répertoire sans spécifier de nouveau nom, l'élément déplacé conserve son nom d'origine.

Attention : Si vous omettez l'indicateur **–i**, la commande **mv** peut écraser de nombreux fichiers. L'indicateur **–i** vous invite à confirmer avant que le fichier ne soit écrasé. L'indicateur **–f** ne vous propose pas d'invite. Si les indicateurs **–f** et **–i** sont tous deux spécifiés, le dernier indicateur spécifié est prioritaire.

Déplacement d'un fichier avec la commande mv

Les exemples ci–après illustrent les différentes utilisations de la commande **mv** :

- Pour déplacer un fichier dans un autre répertoire et lui attribuer un nouveau nom, indiquez :

```
mv intro manuel/chap1
```

Le fichier **intro** est déplacé dans le répertoire**manual/chap1**. Le nom **intro** est supprimé du répertoire actuel et le même fichier apparaît sous le nom **chap1** dans le répertoire **manual**.

- Pour déplacer un fichier dans un autre répertoire en conservant son nom, entrez :

```
mv chap3 manual
```

Le fichier **chap3** est déplacé vers **manual/chap3**.

Changement du nom d'un fichier avec la commande mv

Utilisez la commande **mv** pour changer le nom d'un fichier sans le déplacer vers un autre répertoire.

Pour renommer un fichier, entrez :

```
mv appendix apndx.a
```

Le fichier **appendix** est renommé et s'appelle désormais **apndx.a**. Si un fichier de même nom existe déjà, il est écrasé par le contenu du fichier**appendix**.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **mv** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Copie de fichiers (commande cp)

Utilisez la commande **cp** pour copier le contenu du fichier ou du répertoire spécifié dans les paramètres *SourceFile* ou *SourceDirectory* dans le fichier ou dans le répertoire spécifié par les paramètres *TargetFile* ou *TargetDirectory*. Si le *TargetFile* existe, son contenu est écrasé lors de la copie, sans le moindre message d'avertissement. Pour copier plusieurs *SourceFile*, la cible doit être un répertoire.

Si un fichier portant le même nom existe dans le répertoire de destination, le fichier copié remplace le fichier existant du répertoire cible. Par conséquent, il est utile d'attribuer un *nouveau* nom à la copie du fichier pour s'assurer qu'un fichier du même nom n'existe pas encore dans le répertoire de destination.

Pour mettre une copie du *FichierSource* dans un répertoire, spécifiez un chemin d'accès à un répertoire existant pour le paramètre *RépertoireCible*. Les fichiers conservent leur nom d'origine, sauf si vous en spécifiez d'autres à la fin du chemin d'accès. La commande **cp**

copie également les répertoires entiers dans d'autres répertoires si vous spécifiez les indicateurs **–r** ou **–R**.

Vous pouvez également copier des fichiers spéciaux d'unités à l'aide de l'indicateur **–R**. En indiquant **–R**, les fichiers spéciaux sont recréés sous un nouveau nom de chemin d'accès. En indiquant l'indicateur **–r**, la commande **cp** essaie de copier des fichiers spéciaux dans des fichiers standard.

Les exemples suivants décrivent comment utiliser la commande **cp** :

- Pour copier un fichier du répertoire en cours, saisissez :

```
cp prog.c prog.bak
```

  Appuyez sur Entrée.

  Le fichier **prog.c** est copié dans le fichier **prog.bak**. Si le fichier **prog.bak** n'existe pas, alors la commande **cp** le crée. S'il existe déjà, alors la commande **cp** le remplace par une copie du fichier **prog.c**.

- Pour copier un fichier du répertoire en cours dans un autre répertoire, entrez :

```
cp jones /home/nick/clients
```

  Le fichier **jones** est copié dans le répertoire **/home/nick/clients/jones**.

- Pour copier tous les fichiers d'un répertoire dans un nouveau répertoire, entrez :

```
cp /home/janet/clients/* /home/nick/customers
```

  Seuls les fichiers du répertoire **clients** sont copiés dans le répertoire **customers**.

- Pour copier un ensemble spécifique de fichiers dans un autre répertoire, entrez :

```
cp jones lewis smith /home/nick/clients
```

  Les fichiers **jones**, **lewis** et **smith** de votre répertoire de travail actuel sont copiés dans le répertoire **/home/nick/clients**.

- Pour utiliser les métacaractères lors de la copie de fichiers, entrez :

```
cp programs/*.c .
```

  Les fichiers du répertoire **programs** suffixé .c sont copiés dans le répertoire courant, spécifié par le point (.). Notez que l'extension c et le point final doivent être séparés par un espace.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **cp** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Recherche de fichiers (commande find)

Vous pouvez également utiliser la commande **find** pour effectuer une recherche récursive dans l'arborescence des répertoires pour chaque *Chemin d'accès* spécifié, pour trouver les fichiers correspondant à l'expression booléenne écrite à l'aide des termes donnés dans le texte ci–après. La sortie de la commande **find** dépend des termes spécifiés par le paramètre *Expression*.

Les exemples suivants décrivent comment utiliser la commande **find** :

- Pour afficher tous les fichiers **.profile** du système de fichiers, entrez :

```
find / -name .profile
```

Cette commande permet d'effectuer une recherche dans tout le système de fichiers et écrit les noms de chemins complets de tous les fichiers `.profile`. La barre oblique indique à la commande **find** de rechercher dans le répertoire /(**root**) et dans tous ses sous–répertoires.

Afin de gagner du temps, il est conseillé de limiter la recherche aux répertoires susceptibles de contenir les fichiers recherchés.

- Pour répertorier les fichiers dotés d'un code de droit d'accès spécifique **0600** dans l'arborescence en cours, saisissez :

```
find . -perm 0600
```

Les noms de fichiers possédant *uniquement* les droits d'accès en lecture et en écriture du propriétaire sont alors affichés. Le point (.) indique à la commande **find** de rechercher dans le répertoire courant et dans ses sous–répertoires. Pour obtenir des explications sur les codes d'accès, veuillez vous reporter à la commande **chmod**.

- Pour rechercher dans plusieurs répertoires les fichiers dotés de certains codes de droit d'accès, entrez :

```
find manual clients proposals -perm -0600
```

La commande répertorie ainsi les noms des fichiers dont les droits d'accès en lecture et en écriture sont accordés à leur propriétaire et éventuellement dotés d'autres droits. La recherche est effectuée dans les répertoires **manual**, **clients** et **proposals** ainsi que dans leurs sous–répertoires. Dans l'exemple précédent, **–perm 0600** sélectionne uniquement les fichiers dont les codes de droits d'accès correspondent exactement à **0600**. Dans cet exemple, **–perm –0600** sélectionne les fichiers dont les codes de droits d'accès permettent l'accès indiqué par **0600** et d'autres accès au–dessus du niveau **0600**. Les codes 0622 et 2744 sont ainsi également concernés.

- Pour répertorier tous les fichiers du répertoire en cours ayant été modifiés au cours des dernières 24 heures, saisissez :

```
find . -ctime 1
```

- Pour rechercher les fichiers standard comportant plusieurs liens, entrez :

```
find . -type f -links +1
```

Les noms de fichiers ordinaires (`-type f`) possédant plusieurs liens (`-links +1`).

Remarque : A chaque répertoire sont associés au moins deux liens : l'entrée dans son répertoire parent et son entrée **.** (point). Pour plus d'informations sur des liens de fichiers multiples, reportez–vous à la commande **ln**.

- Pour rechercher tous les fichiers dont la longueur est exactement de 414 octets, saisissez :

```
find . -size 414c
```

Appuyez sur Entrée.

Reportez–vous à la commande **find** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Affichage du type d'un fichier (commande file)

Vous pouvez également utiliser la commande **file** pour lire les fichiers spécifiés par le paramètre *File* ou **–f** *FileList*, et réaliser une série de tests sur chacun de ces fichiers. La commande tente de classer les fichiers par type. Elle envoie ensuite les types de fichiers vers la sortie standard.

Si fichier semble s'afficher en ASCII, la commande **file** examine les 512 premiers octets et détermine la langue du fichier. Si fichier ne semble pas s'afficher en ASCII, la commande **file** tente de déterminer s'il s'agit d'un fichier de données binaires ou d'un fichier texte contenant des caractères étendus.

Si le paramètre *File* spécifie un exécutable ou un fichier de module d'objet et que le numéro de version est supérieur à 0, la commande **file** affiche la version.

La commande **file** utilise le fichier **/etc/magic** pour identifier des fichiers comportant un nombre magique, à savoir, tout fichier comportant une constante numérique ou une constante de type chaîne définissant le type.

Les exemples suivants décrivent comment utiliser la commande **file** :

- Pour afficher le type des informations du fichier **monfich**, entrez :

```
file monfichier
```

  Le type du fichier **monfichier** (par exemple, un répertoire, des données, du texte ASCII, une source de programme C, ou archive).

- Pour afficher le type de chacun des fichiers spécifiés dans le fichier **filenames.lst** (liste de noms de fichiers), entrez :

```
file –f filenames.lst
```

  Le type de chaque fichier nommé dans le fichier **filenames.lst** est affiché. Chaque nom de fichier doit être indiqué sur une ligne distincte.

- Pour créer le fichier **filenames.lst** qui contient tous les noms de fichiers du répertoire courant, entrez :

```
ls > filenames.lst
```

  Editez le fichier **nomsfichiers** si nécessaire.

Reportez–vous à la commande **file** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Affichage du contenu d'un fichier (commandes pg, more, page et cat)

Les commandes **pg**, **more** et **page** permettent de visualiser le contenu d'un fichier et d'en contrôler l'affichage. Vous pouvez également utiliser la commande **cat** pour afficher le contenu d'un ou de plusieurs fichiers sur votre écran. En combinant la commande **cat** avec la commande **pg**, vous pourrez lire le contenu d'un fichier en plein écran, un écran à la fois.

Vous pouvez également consulter le contenu des fichiers via les procédures de réacheminement. Reportez–vous à Réacheminement des entrées/sorties, page 5-1 pour plus de détails sur le réacheminement des entrées/sorties.

commande pg

Utilisez la commande **pg** pour lire les noms de fichiers spécifiés par le paramètre *File* et les afficher page par page sur l'unité de sortie standard. Si vous spécifiez le tiret (–) comme paramètre *File* ou exécutez la commande **pg** sans options, la commande **pg** lit les entrées standard. Chaque écran est suivi d'une invite. Si vous appuyez sur Entrée, un autre écran s'affiche. Des sous–commandes utilisées avec la commande **pg** permettent de revenir à un écran antérieur.

Par exemple, pour consulter page par page le contenu du fichier **monfich**, entrez :

```
pg monfichier
```

Reportez–vous à la commande **pg** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Commande more ou page

La commande **more** ou **page** affiche du texte continu écran par écran. Elle s'arrête après chaque écran et affiche le *nom de fichier* et le pourcentage réalisé (par exemple, `monfichier (7%)`) au bas de l'écran. Si vous appuyez sur Entrée, la commande **more** affiche une ligne supplémentaire. Si vous appuyez sur la barre d'espacement, la commande **more** affiche un autre écran de texte.

Remarque : Sur certains modèles de terminaux, la commande **more** vide l'écran au lieu de le faire défiler, avant d'afficher l'écran de texte suivant.

Par exemple, pour afficher le fichier **monfich**, entrez :

```
more monfichier
```

Appuyez sur la barre d'espacement pour passer à l'écran suivant.

Reportez–vous à la commande **more** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

commande cat

Utilisez la commande **cat** pour lire séquentiellement les paramètres *File* et les écrire sur l'unité de sortie standard.

Voir les exemple ci–après :

- Pour afficher le contenu du fichier **notes**, entrez :

```
cat notes
```

  Si le fichier dépasse 24 lignes, le début défile sans que vous ayez le temps de le lire. Pour obtenir un affichage page par page, utilisez la commande **pg**.

- Pour afficher le contenu des fichiers **notes**, **notes2** et **notes3**, entrez :

```
cat notes notes2 notes3
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **cat** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Recherche de chaînes dans un fichier texte (commande grep)

Utilisez la commande **grep** pour rechercher, dans le fichier spécifié, l'expression générique spécifiée par le paramètre *Pattern* et écrire chaque ligne correspondant à l'expression sur l'unité de sortie standard.

Les exemples ci–après illustrent les différentes utilisations de la commande **grep** :

- Pour rechercher dans le fichier **pgm.s** les lignes correspondant à une expression contenant des caractères de remplacement et des métacaractères (*, ^, ?, [,], , , { et }), ici les lignes commençant par une lettre (majuscule ou minuscule), entrez :

```
grep "^[a-zA-Z]" pgm.s
```

  Toutes les lignes du fichier **pgm.s** commençant par une lettre sont affichées.

- Pour afficher toutes les lignes d'un fichier **sort.c** qui ne correspondent pas à une expression générique particulière, entrez :

```
grep -v bubble sort.c
```

  Toutes les lignes du fichier **sort.c** ne contenant pas le mot **bubble** sont affichées.

- Pour afficher les lignes de la sortie de la commande **ls** contenant la chaîne **staff**, entrez :

```
ls -l | grep staff
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **grep** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Tri des fichiers texte (commande sort)

Utilisez la commande **sort** pour trier par ordre alphabétique les lignes du (des) fichier(s) spécifié(s) par le paramètre *File* et écrire le résultat sur l'unité de sortie standard. Si le paramètre *File* spécifie plusieurs fichiers, la commande **sort** les concatène et trie le fichier résultant.

Remarque : La commande **sort** est sensible à la casse et place généralement les majuscules avant les minuscules (cela peut varier en fonction des paramètres régionaux).

Dans les exemples suivants, le contenu du fichier **names** est :

```
marta
denise
joyce
endrica
melanie
```

et le contenu du fichier **states** est :

```
texas
colorado
ohio
```

- Pour trier le contenu du fichier **names**, entrez :

```
sort names
```

  Le système affiche des informations similaires à l'exemple suivant :

```
denise
endrica
joyce
marta
melanie
```

- Pour trier le contenu des deux fichiers **names** et **states**, entrez :

```
sort names states
```

  Le système affiche des informations similaires à l'exemple suivant :

```
colorado
denise
endrica
joyce
marta
melanie
ohio
texas
```

- Pour remplacer le contenu du fichier **names** d'origine par son contenu trié, entrez :

```
sort –o names names
```

  Le contenu du fichier **names** est remplacé par son contenu trié.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **sort** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Comparaison de fichiers (commande diff)

Utilisez la commande **diff** pour comparer des fichiers texte. Cette commande peut comparer des fichiers isolés ou le contenu de répertoires.

Lorsque la commande **diff** est exécutée sur des fichiers standard ou lorsqu'elle compare des fichiers texte dans des répertoires différents, la commande **diff** indique les lignes qui présentent des différences dans les fichiers.

Les exemples ci–après illustrent les différentes utilisations de la commande **diff** :

- Pour comparer deux fichiers, entrez :

```
diff chap1.bak chap1
```

Les différences entre les chapitres **chap1.bak** et **chap1** s'affichent.

- Pour comparer deux fichiers en ignorant les différences dues au nombre d'espaces, entrez :

```
diff –w prog.c.bak prog.c
```

Deux lignes ne se distinguant que par le nombre d'espaces et de tabulations sont considérées comme identiques par la commande **diff–w**.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **diff** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Décompte des mots, des lignes et des octets d'un fichier (commande wc)

Utilisez la commande **wc** pour compter le nombre de lignes, de mots et d'octets des fichiers spécifiés par le paramètre *File*. Si un fichier n'est pas spécifié pour le paramètre *File*, l'entrée standard est utilisée. La commande écrit le résultat sur l'unité de sortie standard et conserve un décompte total de tous les fichiers nommés. Si des indicateurs sont spécifiés, leur ordre détermine celui de la sortie. Un *mot* est défini par une chaîne de caractères délimitée par des espaces, des tabulations ou des caractères nouvelle ligne.

Si vous indiquez des fichiers sur la ligne de commande, leur nom est inscrit avec le décompte.

Voir les exemple ci–après :

- Par exemple, pour afficher les décomptes en lignes, en mots et en octets du fichier **chap1**, entrez :

```
wc chap1
```

Le nombre de lignes, de mots et d'octets du fichier **chap1** est affiché.

- Pour afficher uniquement le nombre d'octets et de mots, entrez :

```
wc –cw chap*
```

Le nombre d'octets et de mots s'affiche dans chaque fichier dont le nom commence par **chap**, et affiche les totaux.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **wc** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage des premières lignes d'un fichier (commande head)

Utilisez la commande **head** pour écrire sur sortie standard les premières nouvelles lignes des fichiers spécifiés ou de l'entrée standard. En l'absence d'indicateur, ce sont les 10 premières lignes qui sont affichées par défaut.

Par exemple, pour afficher les 5 premières lignes du fichier **Test**, entrez :

```
head –5 Test
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **head** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage des dernières lignes d'un fichier (commande tail)

Utilisez la commande **tail** pour écrire le fichier spécifié par le paramètre *File* sur une sortie standard, en commençant à un point spécifié.

Voir les exemple ci–après :

- Pour afficher les 10 dernières lignes du fichier **notes**, entrez :

```
tail notes
```

- Pour préciser la ligne (en commençant par la fin) à partir de laquelle lire le fichier **notes**, entrez :

```
tail -20 notes
```

- Pour afficher le fichier **notes** page par page, à partir du 200ème octet, entrez :

```
tail -c +200 notes | pg
```

- Pour surveiller la taille du fichier **accounts**, entrez :

```
tail -f accounts
```

  Les 10 dernières lignes du fichier **accounts** s'affichent. La commande **tail** affiche les lignes au fur et à mesure qu'elles s'ajoutent au fichier de comptabilité. L'affichage se poursuit tant que vous n'appuyez pas sur Ctrl–C pour l'interrompre.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **tail** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Coupe de sections de fichiers texte (commande cut)

Utilisez la commande **cut** pour écrire sur l'unité de sortie standard les octets, les caractères ou les champs spécifiés, à partir de chaque ligne d'un fichier.

Voir les exemple ci–après :

- Pour afficher plusieurs champs de chaque ligne d'un fichier, entrez :

```
cut -f1,5 -d: /etc/passwd
```

  Cette commande affiche les champs nom de connexion et nom utilisateur complet du fichier mot de passe du système. Il s'agit des 1er et 5ème champs (–f1,5) séparés par des deux points (–d:).

- Pour un fichier **/etc/passwd** semblable à :

```
su:*:0:0:User with special privileges:/:/usr/bin/sh
daemon:*:1:1::/etc:
bin:*:2:2::/usr/bin:
sys:*:3:3::/usr/src:
adm:*:4:4:System Administrator:/var/adm:/usr/bin/sh
pierre:*:200:200:Pierre Harper:/home/pierre:/usr/bin/sh
joan:*:202:200:Joan Brown:/home/joan:/usr/bin/sh
```

  la commande **cut** génère :

```
su:User with special privileges
daemon:
bin:
sys:
adm:System Administrator
pierre:Pierre Harper
joan:Joan Brown
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **cut** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Collage de sections de fichiers texte (commande paste)

Utilisez la commande **paste** pour fusionner les lignes de plusieurs fichiers (12 au maximum) dans un fichier unique.

Voir les exemple ci–après :

- Soit le fichier **names** contenant le texte :

```
rachel
jerry
mark
linda
scott
```

et un autre fichier nommé **places** contenant le texte suivant :

```
New York
Austin
Chicago
Boca Raton
Seattle
```

et un autre fichier nommé **dates** contenant le texte suivant :

```
February 5
March 13
June 21
July 16
November 4
```

Pour regrouper les fichiers **names**, **places** et **dates**, entrez :

```
paste names places dates > npd
```

Un fichier nommé **npd** est créé ; il contient les données du fichier **names** en première colonne, celles du fichier **places** en deuxième colonne et celles du fichier **dates** en troisième colonne. Le fichier **npd** contient maintenant les données suivantes :

```
rachel          New York        February 5
jerry           Austin          March 13
mark            Chicago         June 21
linda           Boca Raton      July 16
scott           Seattle         November 4
```

Un caractère de tabulation sépare le nom, l'emplacement et la date sur chaque ligne. Ces colonnes ne sont pas alignées car les marques de tabulation sont définise à chaque huitième colonne.

- Pour séparer les colonnes par un caractère autre qu'une tabulation, entrez :

```
paste -d"!@" names places dates > npd
```

Les caractères ! et @ séparent alternativement les données. Si les fichiers **names**, **places** et **dates** sont les mêmes que dans l'exemple 1, le fichier **npd** apparaît sous la forme :

```
rachel!New York@February 5
jerry!Austin@March 13
mark!Chicago@June 21
linda!Boca Raton@July 16
scott!Seattle@November 4
```

- Pour afficher le répertoire actuel en quatre colonnes, entrez :

```
ls | paste - - - -
```

Chaque tiret (–) indique à la commande **paste** de créer une colonne contenant des données issues de l'unité d'entrée standard. La première ligne est affichée dans la première colonne, la deuxième ligne dans la deuxième colonne, etc.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **paste** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Numérotation des lignes d'un fichier texte (commande nl)

Utilisez la commande **nl** pour lire le fichier spécifié (entrée standard par défaut), numéroter les lignes et écrire celles–ci sur la sortie standard.

Voir les exemple ci–après :

- Pour numéroter uniquement les lignes qui ne sont pas vides, entrez :

```
nl chap1
```

  Les lignes non vides du fichier **chap1** s'affichent, numérotées.

- Pour numéroter toutes les lignes, entrez :

```
nl –ba chap1
```

  Toutes les lignes du fichier **chap1** sont numérotées, y compris les lignes vides.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **nl** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Suppression de colonnes dans un fichier texte (commande colrm)

Utilisez la commande **colrm** pour supprimer d'un fichier les colonnes spécifiées. L'entrée est lue sur l'unité d'entrée standard. La sortie est envoyée vers l'unité de sortie standard.

Lancée avec un seul paramètre, la commande supprime de chaque ligne toutes les colonnes à partir de la colonne spécifiée. Lancée avec deux paramètres, ce sont les colonnes comprises entre les deux colonnes spécifiées qui sont supprimées.

Remarque : La numérotation des colonnes commence à 1.

Voir les exemple ci–après :

- Pour supprimer les colonnes du fichier **text.fil**, entrez :

```
colrm 6 < text.fil
```

  Si **text.fil** contient :

```
123456789
```

  la commande **colrm** affiche :

```
12345
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **colrm** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Par exemple, le fichier **/etc/passwd** a la forme suivante :

Liaison entre fichiers et répertoires

Un lien est une connexion entre un nom de fichier et un numéro de référence de nœud d'index (inode), représentation interne d'un fichier. Les entrées de répertoire pouvant comporter des noms de fichiers appariés avec des i-nodes, chaque entrée de répertoire constitue un lien. C'est son numéro d'i-node, et non son nom, qui permet d'identifier un fichier. En utilisant les liens, tout numéro d'i-node ou fichier peut avoir plusieurs noms différents.

Par exemple, l'i-node 798 contient un mémo sur les ventes de Juin de l'agence Omaha. L'entrée de répertoire correspondante est la suivante:

Numéro d'i–node	Nom de fichier
798	**memo**

Comme ces informations font référence à celles stockées dans les répertoires `sales` et `omaha`, la liaison permet de partager les informations si nécessaire. A l'aide de la commande **ln**, des liens sont créés vers ces répertoires. Le fichier est alors référencé sous trois noms :

Numéro d'i–node	Nom de fichier
798	**memo**
798	**sales/june**
798	**omaha/junesales**

Lorsque vous utilisez la commande **pg** ou **cat** pour visualiser le contenu d'un des trois noms de fichier, les mêmes informations sont affichées. Si vous modifiez l'un de ces trois fichiers, le contenu des deux autres est modifié en conséquence.

Types de liens

Les liens sont créés avec la commande **ln** et sont de types suivants :

lien fixe Donne accès aux données d'un fichier par le biais d'un nouveau nom de fichier. Un lien fixe assure l'existence d'un fichier. Lorsqu'un lien fixe est supprimé, le numéro d'i–node et ses données sont également supprimés. Les liens fixes peuvent uniquement être créés entre des fichiers qui sont dans le même système de fichiers.

lien symbolique Donne accès aux données d'autres systèmes de fichiers par le biais d'un nouveau nom de fichier. Un lien symbolique est un type particulier de fichier, contenant un chemin d'accès. Lorsqu'un process rencontre un lien symbolique, il recherche éventuellement ce chemin. Les liens symboliques ne protègent pas les fichiers de leur suppression du système de fichiers.

Remarque : L'utilisateur qui crée un fichier en reste propriétaire quel que soit le nombre de liens qui y sont rattachés. Seul le propriétaire du fichier ou l'utilisateur root peut définir le mode d'accès à ce fichier. Cependant, des modifications peuvent être apportées au fichier à partir d'un nom de fichier lié avec le mode d'accès approprié.

Un fichier ou un répertoire existe tant qu'il existe un lien fixe avec son i–node. Dans la longue liste affichée par la commande **ls –l**, le nombre de liens fixes vers chaque fichier et sous–répertoire est fourni. Tous les liens fixes sont équivalents pour le système : l'ordre de leur création est indifférent.

Liaison de fichiers (commande ln)

Lier des fichiers via la commande **ln** permet de travailler sur les mêmes données en plusieurs endroits. Les liaisons sont générées via l'attribution de noms différents au fichier d'origine. L'utilisation de liaisons permet à plusieurs utilisateurs de partager un même fichier de grande taille (une base de données ou une liste de diffusion, par exemple) sans qu'il soit nécessaire de faire des copies de ce fichier. Les liaisons permettent d'économiser de l'espace disque et présentent également l'avantage suivant : les modifications intégrées dans un fichier sont automatiquement répercutées dans tous les fichiers liés.

La commande **ln** relie le fichier désigné sous le paramètre *SourceFile* au fichier désigné sous le paramètre *TargetFile* ou au fichier de même nom dans un autre répertoire *y* spécifié par le paramètre *TargetDirectory*. Par défaut, la commande **ln** crée des liens fixes. Pour utiliser la commande **ln** afin de créer des liens symboliques, ajoutez l'indicateur **–s**.

Remarque : Vous devez absolument spécifier l'indicateur **–s** pour lier des fichiers entre systèmes de fichiers.

Vous ne pouvez associer qu'un fichier à un nouveau nom. Vous pouvez cependant en associer plusieurs à un répertoire.

Le paramètre *TargetFile* est facultatif. Si vous ne désignez pas de fichier cible, la commande **ln** crée un fichier dans votre répertoire actuel. Le nouveau fichier hérite du nom du fichier désigné sous le paramètre *SourceFile*.

Voir les exemple ci–après :

- Pour créer une liaison vers un fichier nommé **chap1**, entrez :

```
ln –f chap1 intro
```

  Le fichier **chap1** est relié au nom **intro**. Avec l'indicateur **–f**, le nom de fichier **intro** est créé s'il n'existe pas. Si **intro** existe, le fichier est remplacé par une liaison vers **chap1**. Les noms de fichiers **chap1** et **intro** désignent le même fichier.

- Pour relier un fichier nommé **index** au même nom dans un autre répertoire nommé **manual**, entrez :

```
ln index manual
```

  Le fichier **index** est relié au nom **manual/index**.

- Pour lier plusieurs fichiers à des noms dans d'autres répertoires, entrez :

```
ln chap2 jim/chap3 /home/manual
```

  chap2 est lié à **/home/manual/chap2** et **jim/chap3** à **/home/manual/chap3**.

- Pour utiliser la commande **ln** avec des caractères de remplacement, entrez :

```
ln manual/* .
```

  Remarque : Vous devez séparer l'astérisque du point par un espace. Tous les fichiers contenus dans le répertoire **manual** sont liés dans le répertoire actuel, dot (.). Les fichiers ont le même nom que dans le répertoire **manual**.

- Pour créer un lien symbolique, entrez :

```
ln –s /tmp/toc toc
```

  Le lien symbolique **toc** est créé dans le répertoire courant. **Le fichier** toc pointe sur le fichier **/tmp/toc**. Si le fichier **/tmp/toc** existe, la commande **cat toc** affiche son contenu.

- Pour obtenir le même résultat sans préciser le paramètre *TargetFile*, entrez :

```
ln –s /tmp/toc
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **ln** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Suppression de fichiers liés

Utilisez la commande **rm** pour supprimer le lien du nom de fichier indiqué. Lorsque l'un des noms de fichiers à lien fixe est supprimé, le fichier n'est pas complètement supprimé puisqu'il reste sous l'autre nom. Ce n'est qu'à la suppression du dernier lien à l'i–node que les données sont supprimées. Le numéro d'i–node correspondant est alors libéré.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **rm** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Fichiers DOS

Le système d'exploitation AIX vous permet de gérer des fichiers DOS sur votre système. Il vous suffit de copier les fichiers DOS voulus sur une disquette. Votre système peut lire ces fichiers dans un répertoire de système d'exploitation de base dans le format approprié, et sur la disquette sous format DOS.

Remarque : Les caractères génériques * et ? (astérisque et point d'interrogation) ne sont pas admis pour les commandes concernées dans cette section (bien qu'ils soient acceptés par le shell). Si vous ne spécifiez pas d'extension de nom de fichier, vous aurez une extension vide.

Copie les fichiers DOS dans des fichiers de système d'exploitation de base

La commande **dosread** copie le fichier DOS spécifié dans le fichier de système d'exploitation de base indiqué.

Remarque : Les conventions d'appellation des fichiers s'appliquent, à une exception près. Comme la barre oblique inversée (\) peut avoir une signification particulière pour le système d'exploitation de base, utilisez la barre oblique (/) comme délimiteur pour spécifier des noms de sous–répertoire dans les noms de chemin DOS.

Voir les exemple ci–après :

- Pour copier un fichier texte nommé **chap1.doc** à partir d'une disquette DOS vers le système de fichiers du système d'exploitation de base, entrez :

```
dosread –a chap1.doc chap1
```

Le fichier texte DOS **\CHAP1.DOC** de l'unité par défaut **/dev/fd0** est copié sur le fichier système de base **chap1** dans le répertoire actuel.

- Pour copier un fichier binaire à partir d'une disquette DOS sur le système de fichiers du système d'exploitation de base, entrez :

```
dosread –D/dev/fd0 /survey/test.dta /home/fran/testdata
```

Le fichier de données DOS **\SURVEY\TEST.DTA** de **/dev/fd0** est copié sur le fichier du système d'exploitation de base **/home/fran/testdata**.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **dosread** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Copie de fichiers de système d'exploitation de base dans des fichiers DOS

Utilisez la commande **doswrite** pour copier le fichier de système d'exploitation de base dans le fichier DOS spécifié.

Remarque : Les conventions DOS d'appellation des fichiers s'appliquent, à une exception près : étant donné que la barre oblique inverse (\) peut avoir une significations spéciale dans le système d'exploitation de base, utilisez une barre oblique (/) comme caractère délimiteur pour spécifier les noms de sous–répertoires dans un chemin d'accès DOS.

Voir les exemple ci–après :

- Pour copier un fichier texte nommé **chap1** du système de fichiers du système d'exploitation de base sur une disquette DOS, entrez :

```
doswrite –a chap1 chap1.doc
```

Le fichier du système d'exploitation de base **chap1** du répertoire actuel est copié dans le fichier texte DOSe **\CHAP1.DOC** de **/dev/fd0**.

- Pour copier un fichier binaire nommé **/survey/test.dta** sur une disquette DOS depuis le système de fichiers du système d'exploitation de base, entrez :

```
doswrite -D/dev/fd0 /home/fran/testdata /survey/test.dta
```

  Le fichier de données du système d'exploitation de base **/home/fran/testdata** est copié dans le fichier DOS **\SURVEY\TEST.DTA** sur **/dev/fd0**.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **doswrite** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Suppression de fichiers DOS

Utilisez la commande **dosdel** pour supprimer le fichier DOS spécifié.

> **Remarque :** Les conventions d'appellation des fichiers s'appliquent, à une exception près. Comme la barre oblique inversée (\) peut avoir une signification particulière pour le système d'exploitation de base, utilisez la barre oblique (/) comme délimiteur pour spécifier des noms de sous–répertoire dans les noms de chemin DOS.

La commande **dosdel** convertit en majuscules les minuscules apparaissant dans le nom du fichier ou du répertoires avant de vérifier le disque. Tous les chemins d'accès étant supposés absolus (et non relatifs), la barre oblique (/) initiale est inutile.

Par exemple, pour supprimer un fichier DOS intitulé **file.ext** de l'unité par défaut (**/dev/fd0**), entrez :

```
dosdel file.ext
```

Reportez–vous à la commande **dosdel** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Affichage du contenu d'un répertoire DOS

Utilisez la commande **dosdir** pour afficher des informations sur les fichiers ou répertoires DOS spécifiés.

> **Remarque :** Les conventions d'appellation des fichiers s'appliquent, à une exception près. Comme la barre oblique inversée (\) peut avoir une signification particulière pour le système d'exploitation de base, utilisez la barre oblique (/) comme délimiteur pour spécifier des noms de sous–répertoire dans les noms de chemin DOS.

La commande **dosdir** convertit en majuscules les minuscules apparaissant dans le nom du fichier ou du répertoires avant de vérifier le disque. Tous les chemins d'accès étant supposés absolus (et non relatifs), la barre oblique initiale (/) est inutile.

Par exemple, pour lire un répertoire des fichiers DOS sur l'unité **/dev/fd0**, entrez :

```
dosdir
```

La commande affiche le nom des fichiers, et la taille de l'espace disponible, comme dans l'exemple suivant.

```
PG3-25.TXT
PG4-25.TXT
PG5-25.TXT
PG6-25.TXT
Espace libre : 312320 octets
```

Reportez–vous à la commande **dosdir** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Récapitulatif des commandes relatives aux fichiers

*****	Caractère générique utilisé pour remplacer un ou plusieurs caractères
?	Caractère générique utilisé pour remplacer un seul caractère
[ ]	Métacaractères utilisés pour remplacer des caractères inclus

Procédures de gestion de fichier

cat	Concatène ou affiche des fichiers
cmp	Compare deux fichiers
colrm	Extrait des colonnes d'un fichier
cp	Copie des fichiers
cut	Ecrit des octets, des caractères ou des champs sélectionnés de chaque ligne d'un fichier
diff	Compare des fichiers texte
file	Détermine le type de fichier
find	Recherche des fichiers avec une expression correspondant
grep	Recherche un fichier pour un modèle
head	Affiche les premières lignes ou les premiers octets d'un ou de plusieurs fichiers
more	Affiche du texte continu un écran à la fois
mv	Déplace des fichiers
nl	Numérote les lignes dans un fichier
pg	Formate les fichiers à l'affichage
rm	Retire (délie) des fichiers ou des répertoires
paste	Fusionne les lignes de plusieurs fichiers ou des lignes qui se suivent dans un fichier
page	Affiche du texte continu un écran à la fois
sort	Trie les fichiers, fusionne les fichiers déjà triés et vérifie si les fichiers ont été triés
tail	Envoie un fichier vers la sortie standard en commençant à un endroit spécifique
wc	Compte le nombre de lignes, de mots et d'octets dans un fichier

Liaison entre fichiers et répertoires

ln	Relie fichiers et répertoires

Fichiers DOS

dosdel	Supprime des fichiers DOS
dosdir	Répertorie le répertoire des fichiers DOS
dosread	Copie les fichiers DOS dans des fichiers de système d'exécution de base
doswrite	Copie les fichiers de système d'exécution de base dans des fichiers DOS

Chapitre 8. Imprimantes, travaux d'impression et files d'attente

Vous pouvez, en fonction de l'imprimante, contrôler la présentation et les caractéristiques des sorties papier. L'imprimante, d'une part, la console et l'unité centrale, d'autre part, peuvent se trouver dans des lieux distincts. L'imprimante peut être directement connectée à un système local, mais un travail d'impression peut également être envoyé via un réseau à un système distant.

Pour gérer les travaux d'impression avec un maximum d'efficacité, le système place chaque travail dans une file d'attente, jusqu'à ce que l'imprimante soit disponible. Un ou plusieurs travaux peuvent ainsi se trouver en file d'attente : Comme l'imprimante produit la sortie à partir d'un fichier, le système traite la prochaine tâche dans la file d'attente. Ce process continue jusqu'à l'impression de chaque tâche dans la file d'attente.

Pour des informations détaillées sur les imprimantes, les travaux d'impression ou les files d'attente, reportez–vous au manuel *AIX 5L Version 5.3 Imprimantes et impression – Guide de l'utilisateur*.

Ce chapitre traite des points suivants :

Terminologie relative à l'impression

Voici une définition des principaux termes relatifs à l'impression.

imprimante locale

Lorsque vous branchez une imprimante sur un nœud ou sur un hôte, elle est dite *imprimante locale*.

travail d'impression

Un *travail d'impression* est une unité de travail destinée à l'imprimante. Il peut être constitué d'un ou de plusieurs fichiers, selon la demande émise. Le système affecte un numéro unique à chaque travail qu'il exécute.

spouleur d'impression

Le *spouleur* utilisé pour l'impression offre une fonction de traitement différé générique, permettant la mise en file d'attente de différents types de travaux, y compris des travaux d'impression. Le spouleur ne "sait" normalement pas quel type de travail est mis en attente. Lorsque l'administrateur définit une file d'attente, son objet est déterminé par le programme expéditeur spécifié pour la file d'attente. Par exemple, si ce programme est la commande **piobe** (programme d'entrée E/S de l'imprimante), la file d'attente sera une file d'impression.

De même, si le programme expéditeur est un compilateur, il s'agit d'une file d'attente de travaux à compiler. Lorsque la commande **qdaemon** du spouleur sélectionne un travail de la file, il exécute le travail en appelant le programme expéditeur spécifié par l'administrateur lors de la définition de la file d'attente.

enq est la principale commande du spouleur. Pour placer un travail d'impression en file d'attente, vous pouvez directement invoquer cette commande, mais vous avez trois autres commandes frontales à disposition : Les commandes **lp**, **lpr** et **qprt**. Une demande émise par le biais d'une de ces commandes est d'abord transmise au programme **enq,** lequel place les informations sur le fichier dans la file d'attente, en vue de son traitement par le process **qdaemon**.

Programme expéditeur de l'imprimante

Le *programme expéditeur de l'imprimante* est un ensemble de programmes appelés par la commande **qdaemon** du spouleur pour traiter un travail en attente d'impression. Ce programme :

- reçoit de la commande **qdaemon** la liste des fichiers à imprimer ;
- applique les attributs imprimante et de formatage issus de la base de données, éventuellement remplacés par les indicateurs spécifiés sur la ligne de commande ;
- initialise l'imprimante avant d'imprimer un fichier ;
- lance au besoin les filtres requis pour convertir le flux des données dans un format pris en charge par l'imprimante ;
- fournit les filtres requis pour le formatage simple des documents ASCII ;
- assure le support des caractères nationaux ;
- transmet les données d'impression filtrées au pilote de l'unité d'impression ;
- génère les pages de début et de fin ;
- génère les exemplaires multiples ;
- signale les incidents (absence de papier, intervention requise, erreur imprimante, etc.) ;

- signale les incidents décelés par les filtres ;
- effectue un nettoyage après l'annulation d'un travail d'impression ;
- fournit un environnement d'impression, personnalisable par l'administrateur système.

unité d'imprimante/de traceur

Fichier spécifique figurant dans le répertoire **/dev** de l'unité. Ce fichier peut être utilisé par la fonction de réacheminement (par exemple, `cat FileName >/dev/lp0`). Pour modifier et afficher les paramètres du pilote d'unité, vous pouvez utiliser Web–based System Manager (entrez `wsm`, puis sélectionnez **Devices**) ou les commandes **lsdev** et **chdev**. Pour que les commandes de l'imprimante puissent accéder à une unité d'impression, une file d'attente d'impression doit être créée pour l'unité, ou l'imprimante doit être configurée dans le programme expéditeur de l'imprimante dans **/etc/qconfig**.

qdaemon

qdaemon est un process d'arrière–plan qui contrôle les files d'attente. Il est généralement initié par la commande **startsrc** lorsque le système est activé. **startsrc** est un process associé au démon **srcmstr** qui est initié par la commande **/etc/inittab**.

qdaemon assure le suivi des requêtes d'impression dans le répertoire **/var/spool/lpd/qdir** et veille à ce que les travaux soient dirigés sur l'imprimante adéquate et au moment opportun. Il assure aussi le suivi de l'état des imprimantes et enregistre les données concernant l'utilisation des imprimantes pour la comptabilité du système (pour les commandes **lpstat** et **enq –A**, par exemple). Ces données sont consignées dans le répertoire **/var/spool/lpd/stat**.

Quand **qdaemon** est arrêté, il est relancé par le processus **srcmstr**.

Remarque : N'interrompez pas **srcmstr** ; il contrôle d'autres démons actifs sur le système.

file d'attente

La *file d'attente* est l'endroit vers lequel sont dirigés les travaux d'impression. Il s'agit d'une strophe dans le fichier **/etc/qconfig** dont le nom correspond à celui de la file d'attente. Elle désigne l'unité de file d'attente associée. En voici un exemple :

```
Msa1:
    device = lp0
```

En général, les files d'attente sont créées via Web–based System Manager.

unité de file d'attente

La strophe relative à *l'unité de file d'attente* suit généralement celle relative à la file d'attente locale dans le fichier **/etc/qconfig**. Elle indique le fichier **/dev** (unité imprimante) destinataire de l'impression et le programme expéditeur à utiliser. En voici un exemple :

```
lp0:
file = /dev/lp0
header = never
trailer = never
access = both
    backend = /usr/lpd/piobe
```

Dans l'exemple précédent, `lp0` est le nom de l'unité, les autres lignes définissant son mode d'utilisation.

L'ajout d'une imprimante via Web–based System Manager (entrez `wsm`, puis sélectionnez **Devices**) crée automatiquement une entrée pour une unité de file d'attente standard dans une file d'attente existante.

Remarques :

1. Plusieurs unités de file d'attente peuvent être associées à une seule file d'attente.
2. Si vous utilisez une imprimante distante, il n'y a pas d'entrée particulière dans le fichier **/etc/qconfig**. La file d'attente dirige le fichier vers le serveur.

imprimante réelle

Une *imprimante réelle* est l'imprimante matérielle connectée à un port série ou parallèle à une adresse matérielle unique. Au niveau du noyau, le programme pilote de l'imprimante communique avec l'imprimante matérielle pour lui fournir une interface avec une imprimante virtuelle.

imprimante distante

Un *système d'impression à distance* donne aux noeuds non directement connectés accès à l'imprimante.

Pour accéder à distance aux fonctions d'impression, les nœuds indépendants doivent être connectés à un réseau via le protocole TCP/IP (Transmission Control Protocol/Internet Protocol) ; ils doivent dans ce cas prendre en charge les applications TCP/IP nécessaires.

imprimante virtuelle

Une *imprimante virtuelle*, généralement appelée *définition d'imprimante virtuelle*, est un fichier contenant un ensemble de valeurs d'attribut décrivant un flux de données spécifique pour une imprimante donnée. Avant qu'un travail d'impression puisse être placé dans une file d'attente, une définition d'imprimante virtuelle doit exister à la fois pour la file d'attente d'impression et pour l'unité de file d'attente. Reportez–vous à la commande **mkvirprt** pour plus d'informations.

Lancement d'un travail d'impression (commande qprt)

Utilisez la commande **qprt** ou **smit** pour lancer un travail d'impression et spécifier les éléments suivants :

- Nom du fichier à imprimer
- Nom de la file d'attente d'impression
- Nombre de copies à imprimer
- Copie du fichier sur l'hôte distant ?
- Fichier à effacer après l'impression ?
- Notifier l'état du travail d'impression ?
- Notifier l'état du travail d'impression via la messagerie ?
- Impression continue ou non
- Nom utilisateur pour l'étiquette 'Delivery To' (destinataire)
- Message d'accusé de réception de la console pour l'imprimante distante
- Message d'accusé de réception du fichier pour l'imprimante distante
- Niveau de priorité

Préalables

- Pour les travaux d'impression locaux, l'imprimante doit être connectée physiquement au système, ou, si vous utilisez une imprimante de réseau, configurée et connectée au réseau.
- Pour les travaux d'impression distants, le système doit être configuré pour communiquer avec le serveur d'impression distant.
- Pour imprimer un fichier, vous devez disposer des droits de *lecture* sur ce fichier. Pour supprimer un fichier imprimé, vous devez disposer des droits d'*écriture* sur le répertoire où il est stocké.

Utilisation de la commande qprt

La commande **qprt** crée et place en file d'attente un travail d'impression afin d'imprimer le fichier indiqué. Si vous indiquez plusieurs fichiers, l'ensemble de ces fichiers forme un seul travail d'impression. Les fichiers sont imprimés dans l'ordre indiqué sur la ligne de commande.

Le format de base de la commande **qprt** est :

```
qprt -Pnom_file nom_fichier
```

Vous pouvez ajouter différents indicateurs à la commande **qprt** :

–b *Nombre*	Définit la marge en bas de page. Elle est représentée par le nombre de lignes laissées vierges en bas des pages.
–B *Valeur*	Spécifie le mode d'impression des pages en continu (séparables au niveau des pointillés). La variable *Valeur* est une chaîne à deux caractères. Le premier caractère s'applique aux pages d'en–tête. Le second caractère s'applique aux pages d'en–queue. Vous leur donnez la valeur souhaitée : **a** Imprime toujours la page (d'en–tête ou d'en–queue) sur les travaux d'impression. **ur** N'imprime jamais la page (d'en–tête ou d'en–queue). **g** Imprime la page (d'en–tête ou d'en–queue) une seule fois par travail d'impression (groupe de fichiers). Par exemple, **–B ga** spécifie l'impression d'une page d'en–tête au début de chaque travail et d'une page d'en–queue à la fin de chaque fichier pour chaque travail d'impression. **Remarque :** En environnement d'impression distant, la valeur par défaut est déterminée à distance par la file d'attente du serveur.
–e *Option*	Indique si l'impression est demandée avec enrichissement de texte. **+** L'enrichissement de texte est demandé. **!** L'enrichissement de texte n'est pas demandé.
–E *Option*	Indique si l'impression est demandée en double hauteur. **+** L'impression est demandée en double hauteur. **!** L'impression en double hauteur n'est pas demandée.
–f *TypeFiltre*	Identificateur d'un caractère spécifiant un filtre par lequel les fichiers doivent passer avant d'être envoyés à l'impression. Les identificateurs disponibles sont **p**, qui sollicite le filtre **pr** et **n**, qui traite les sorties résultant de la commande **troff**.
–i *Nombre*	Décale chaque ligne selon le nombre d'espaces indiqué. La variable *Nombre* doit être intégrée à la largeur de page indiquée par l'indicateur **–w**.
–K *Option*	Indique si l'impression condensée est demandée. **+** L'impression condensée est demandée. **!** L'impression condensée n'est pas demandée.
–l *Nombre*	Définit la longueur de la page selon le nombre de lignes indiqué. Si la variable *Nombre* possède la valeur 0, la longueur de page n'étant pas prise en compte, la sortie est imprimée en continu. La longueur de page, tenant compte des marges haut et bas, indique la taille imprimable du papier.

–L *Option*	Indique si les lignes dépassant la largeur de page doivent être envoyées à la ligne suivante ou tronquées à la marge de droite. **+** La fin des lignes étendues passe à la ligne suivante. **!** Les lignes étendues sont tronquées au niveau de la marge de droite.
–N *Nombre*	Indique le nombre d'exemplaires à imprimer. Sans cet indicateur, un seul exemplaire est imprimé.
–p *Nombre*	Définit le pas (espacement) à *nombre* de caractères par pouce. 10 et 12 sont des valeurs standard pour *Nombre*. Le pas réel des caractères imprimés dépend aussi des valeurs des indicateurs **–K** (texte condensé) et **–W** (double largeur).
–P *FileAttente* [: *UnitéFileAttente*]	Désigne la file d'attente d'impression et (facultatif) l'unité de file d'attente. Sans cet indicateur, l'imprimante désignée est celle définie par défaut.
–Q *Valeur*	Spécifie le format du papier pour le travail d'impression La *valeur* de la taille du papier dépend de l'imprimante. 1 pour le format lettre, 2 pour le format 'légal', etc. Consultez votre manuel d'imprimante pour les valeurs attribuées aux formats de papier spécifiques.
–t *Nombre*	Définit la marge en haut de page. Définit la marge en haut de page. Elle est représentée par le nombre de lignes laissées vierges en haut des pages.
–w *Nombre*	Définit la largeur de page selon le nombre de caractères indiqués dans la variable *Nombre*. La largeur doit tenir compte de la valeur d'indentation spécifiée par l'indicateur **–i**.
–W *Option*	Indique si l'impression est demandée en double largeur. **+** L'impression est demandée en double largeur. **!** L'impression en double largeur n'est pas demandée.

–z *Valeur*	Fait pivoter la sortie papier de l'imprimante du nombre de quarts de tour (sens horaire) spécifié par la variable *Valeur*. La longueur (**–l**) et la largeur (**–w**) sont automatiquement ajustées en conséquence. **0** Portrait **1** Paysage à droite **2** Portrait à l'envers **3** Paysage à gauche.
–# *Valeur*	Indique une fonction spécifique. **j** Affiche le numéro du travail d'impression spécifié **h** Place le travail en attente, mais à l'état **HELD** (bloqué) jusqu'à ce qu'il soit libéré. **v** Valide les valeurs des indicateurs du programme expéditeur de l'imprimante. Cette validation sert notamment à détecter les indicateurs illégaux au moment de soumettre un travail d'impression. Si la validation n'est pas définie, un travail d'impression assorti d'un indicateur incorrect ne sera interrompu qu'ultérieurement, en cours de traitement.

Les exemples suivants décrivent l'utilisation des indicateurs de commande **qpri** :

- Pour imprimer le fichier **myfile** sur la première imprimante disponible configurée pour la file d'attente d'impression par défaut et avec les valeurs définies par défaut, entrez :

```
qprt myfile
```

- Pour imprimer le fichier **myfile** sur une file d'attente d'impression donnée, avec des indicateurs spécifiques et pour valider ces indicateurs au moment où le travail sera envoyé en impression, entrez :

```
qprt –f p –e + –Pfastest –# v somefile
```

  Le fichier **myfile** est traité par la commande du filtre **pr** (indicateur **–f p**) puis imprimé en mode enrichissement de texte (indicateur **–e +**) sur la première imprimante disponible configurée pour la file d'attente **fastest** (indicateur **–Pfastest**).

- Pour imprimer **myfile** sur une page au format légal, entrez :

```
qprt –Q2 myfile
```

- Pour imprimer trois exemplaires de chacun des fichiers **new.index.c**, **print.index.c** et **more.c** sur la file d'attente `Msp1`, entrez :

```
qprt –PMsp1 –N 3 new.index.c print.index.c more.c
```

- Pour imprimer trois exemplaires de la concaténation des trois fichiers **new.index.c**, **print.index.c** et **more.c**, entrez :

```
cat new.index.c print.index.c more.c | qprt -PMsp1 -N 3
```

Remarque : Le système d'exploitation de base prend aussi en charge la commande d'impression UNIX BSD (**lpr**) et celle d'UNIX System V (**lp**). Reportez–vous aux descriptions des commandes **lpr** et **lp** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour plus d'informations et pour obtenir des détails sur la syntaxe.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **qprt** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Commande smit

Pour lancer un travail d'impression via SMIT, entrez :

```
smit qprt
```

Annulation d'un travail d'impression (commande qcan)

Cette section explique comment annuler un travail d'impression dans la file d'attente d'impression. Lors de l'annulation, vous êtes invité à fournir le nom de la file d'attente d'impression où est contenu le travail et le numéro de travail à annuler.

La procédure décrite s'applique aux travaux d'impression traités localement et à distance.

Préalables

- Pour les travaux d'impression locaux, l'imprimante doit être connectée physiquement au système, ou, si vous utilisez une imprimante de réseau, configurée et connectée au réseau.
- Pour les travaux d'impression distants, le système doit être configuré pour communiquer avec le serveur d'impression distant.

Web–based System Manager

Pour annuler un travail d'impression à l'aide du raccourci Web–based System Manager, entrez :

```
wsm printers
```

Dans la fenêtre d'impressions en attente, sélectionnez le travail d'impression, puis utilisez les menus pour l'annuler d'une file d'attente d'impression.

Utilisation de la commande qcan

La commande **qcan** permet d'annuler un travail spécifique d'une file d'impression locale ou distante ou tous les travaux d'une file d'attente d'impression locale. Pour déterminer le travail, entrez la commande **qchk**.

Le format de base de la commande **qcan** est le suivant :

```
qcan –PQueueName –x JobNumber
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **qcan** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Les exemples suivants décrivent l'utilisation de la commande **qcan** :

- Pour annuler le travail `123` quelle que soit l'imprimante sur laquelle ce travail s'effectue, entrez :

```
qcan –x 123
```

- Pour annuler tous les travaux en attente sur l'imprimante `lp0`, entrez :

```
qcan –X –Plp0
```

Remarque : Le système d'exploitation de base prend également en charge la commande d'annulation d'impression UNIX BSD (**lprm**) et la commande d'annulation d'impression d'UNIX System V (**cancel**). Pour plus d'informations et pour obtenir des détails sur la syntaxe, reportez–vous aux descriptions des commandes **lprm** et **cancel** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Commande smit

Pour annuler un travail d'impression via SMIT, entrez :

```
smit qcan
```

Priorité d'un travail d'impression (commande qpri)

Cette section indique comment modifier la priorité affectée à un travail d'impression. Vous pouvez affecter les priorités voulues sur les files locales. Plus la valeur est élevée, plus la priorité est grande. La valeur par défaut est `15`. La priorité maximale est `20` pour la plupart des utilisateurs et `30` pour les utilisateurs avec des privilèges racine et les membres du groupe `printq` (groupe `9`).

Remarque : Vous ne pouvez pas attribuer de priorité aux travaux d'impression traités à distance.

Préalables

L'imprimante doit être connectée physiquement au système.

Web–based System Manager

Pour modifier la priorité d'un travail d'impression en attente à l'aide de Web–based System Manager, entrez `wsm`, puis sélectionnez **Printers**.

Sélectionnez le travail voulu dans la fenêtre, puis affectez–lui une priorité dans une file locale via les menus proposés.

Commande qpri

Utiliser la commande **qpri** pour réaffecter la priorité d'un travail d'impression que vous avez soumis. Si vous êtes utilisateur racine ou faites partie du groupe `printq`, vous pouvez affecter une priorité à n'importe quel travail d'impression en file d'attente.

Le format de base de la commande **qpri** est :

```
qpri -# JobNumber -a PriorityLevel
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **qpri** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Les exemples suivants décrivent comment utiliser la commande **qpri** :

- Pour attribuer au numéro de travail `123` la priorité `18`, entrez :

```
qpri -# 123 -a 18
```

- Pour donner la priorité à un travail d'impression local au moment de soumettre ce travail, entrez :

```
qprt -PQueueName -R PriorityLevel FileName
```

Commande smit

Pour changer la priorité d'un travail d'impression via SMIT, entrez :

```
smit qpri
```

Déplacement d'un travail d'impression dans une autre file d'attente d'impression (commande qmov)

Cette section décrit le déplacement d'un travail d'impression vers une autre file d'attente.

Remarque : vous ne pouvez pas déplacer vers une autre file d'attente un travail d'impression traité à distance.

Préalables

L'imprimante doit être connectée physiquement au système.

Web–based System Manager

Pour déplacer un travail d'impression vers une autre file d'attente d'impression à l'aide de Web–based System Manager, entrez `wsm`, puis sélectionnez **Printers**.

Dans la fenêtre Printer Queues, sélectionnez le travail d'impression, puis déplacez–le d'une file d'attente vers une autre via les menus proposés.

Utilisation de la commande qmov

Utilisez la commande **qmov** pour déplacer un travail d'impression d'une file d'attente à une autre. Vous pouvez déplacer un travail d'impression particulier ou déplacer tous les travaux d'impression d'une file d'attente d'impression spécifiée ou encore tous les travaux d'impression envoyés par un utilisateur spécifié. Pour déterminer le numéro du travail d'impression, entrez la commande **qchk**. Pour en savoir plus, reportez–vous à commande qchk, page 8-14.

Le format de base de **qmov** est :

```
qmov  –mNewQueue {[ –#JobNumber ] [ –PQueue ] [ –uUser ]}
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **qmov** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Les exemples suivants décrivent l'utilisation de la commande **qmov** :

- Pour déplacer le travail `280` vers la file d'attente d'impression `hp2`, entrez :

```
qmov –mhp2 –#280
```

- Pour déplacer tous les travaux d'impression de la file d'attente d'impression `hp4D` vers la file d'attente d'impression `hp2`, entrez :

```
qmov –mhp2 –Php4D
```

Commande smit

Pour déplacer un travail d'impression via SMIT, entrez :

```
smit qmov
```

Blocage et libération d'un travail d'impression (commande qhld)

Cette section décrit le blocage d'un travail d'impression envoyé dans la file d'attente d'impression. Par la suite, pour libérer les travaux bloqués, utilisez les mêmes commandes.

Remarque : Vous ne pouvez pas bloquer ou libérer les travaux d'impression traités à distance.

Préalables

L'imprimante doit être connectée physiquement au système.

Web–based System Manager

Pour bloquer ou libérer un travail d'impression à l'aide de Web–based System Manager, entrez `wsm`, puis sélectionnez **Printers**.

Dans la fenêtre Printer Queues, sélectionnez le travail d'impression puis utilisez les menus pour le bloquer ou le libérer.

Utilisation de la commande qhld

La commande **qhld** permet de bloquer un travail d'impression placé dans la file d'attente. Il est possible de mettre en suspens un travail d'impression particulier ou tous les travaux d'impression d'une file d'attente spécifiée. Pour déterminer le numéro du travail d'impression, entrez la commande **qchk**. Pour en savoir plus, reportez–vous à la description de la commande qchk, page 8-14.

Le format de base de la commande **qhld** est :

```
qhld [ -r ] {[ -#JobNumber ] [ -PQueue ] [ -uUser ]}
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **qhld** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Les exemples suivants décrivent l'utilisation de la commande **qhld** :

- Pour libérer le travail`452` quelle que soit la file d'attente sur laquelle il se trouve, entrez :

```
qhld -#452
```

- Pour libérer tous les travaux de la file d'attente `hp2`, entrez :

```
qhld -Php2
```

- Pour libérer le travail`452` quelle que soit la file d'attente sur laquelle il se trouve, entrez :

```
qhld -#452 -r
```

- Pour libérer tous les travaux de la file d'attente `hp2`, entrez :

```
qhld -Php2 -r
```

commande smit

Pour bloquer ou libérer un travail d'impression avec SMIT, entrez :

```
smit qhld
```

Contrôle de l'état d'un travail d'impression (commande qchk)

Cette section décrit l'affichage des informations d'état en cours pour des travaux, des files d'attente, des imprimantes ou des utilisateurs spécifiés.

Préalables

- Pour les travaux d'impression locaux, l'imprimante doit être connectée physiquement au système, ou, si vous utilisez une imprimante de réseau, configurée et connectée au réseau.
- Pour les travaux d'impression distants, le système doit être configuré pour communiquer avec le serveur d'impression distant.

Web–based System Manager

Pour vérifier l'état d'un travail d'impression avec Web–based System Manager, tapez `wsm`, puis sélectionnez **Printers**.

Dans la fenêtre Printer Queues, sélectionnez le travail d'impression, puis vérifiez son état à l'aide des menus.

Utilisation de la commande qchk

Utilisez la commande **qchk** pour afficher les informations sur l'état en cours des travaux d'impression, des files d'impression ou des utilisateurs spécifiés.

Le format de base de la commande **qchk** est le suivant :

```
qchk -P QueueName -# JobNumber -u OwnerName
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **qchk** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Les exemples suivants décrivent l'utilisation de la commande **qchk** :

- Pour afficher l'état de la file d'attente par défaut, entrez :

```
qchk -q
```

- Pour afficher l'état long de toutes les files jusqu'à ce qu'elles soient vides tout en mettant à jour l'écran toutes les `5` secondes, entrez :

```
qchk -A -L -w 5
```

- Pour afficher l'état de la file d'attente `lp0`, entrez :

```
qchk -P lp0
```

- Pour afficher l'état du travail `123`, entrez :

```
qchk -# 123
```

- Pour vérifier l'état de tous les travaux de toutes les files d'attente, saisissez :

```
qchk -A
```

Remarque : Le système d'exploitation de base prend également en charge la commande de contrôle de file d'attente UNIX BSD (**lpq**) et celle d'UNIX System V (**lpstat**). Pour obtenir des détails sur la syntaxe, reportez–vous à la description des commandes **lpq** et **lpstat** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Commande smit

Pour contrôler l'état d'un travail d'impression avec SMIT, entrez :

```
smit qchk
```

Conditions de l'état de file d'attente d'impression

Voici quelques–unes des conditions d'état d'une file d'impression :

DEV_BUSY	Signification : • Plusieurs files sont définies pour une unité d'impression (`lp0`) et une autre file utilise actuellement l'imprimante. • **qdaemon** a essayé d'utiliser l'unité de port d'imprimante (`lp0`), mais une autre application utilise actuellement cette imprimante. Pour pallier un état **DEV_BUSY**, patientez jusqu'à ce que la file ou l'application libère l'imprimante ou annulez le travail (ou le processus) qui utilise le port d'imprimante.
DEV_WAIT	La file attend l'imprimante, celle–ci étant hors ligne en raison d'un manque ou d'un bourrage, ou parce que son câble est desserré, défectueux ou n'est pas connecté correctement. Pour pallier un état **DEV_WAIT**, corrigez le problème à l'origine de l'attente. Pour exécuter les tests de diagnostic, il peut s'avérer plus simple d'utiliser la commande **enq** pour transférer tous les travaux en attente de la file **DEV_WAIT** à une autre file qui, soit est en cours d'impression, soit dans l'état **DOWN**. Une fois le problème résolu, vous pouvez replacer les travaux non imprimés dans leur file d'origine.
DOWN	Une file d'attente passe généralement à l'état **DOWN** après avoir été dans l'état **DEV_WAIT**. Ceci se produit quand le pilote de l'imprimante ne reconnaît plus l'imprimante en raison d'une mauvaise signalisation. Néanmoins, certaines imprimantes ne peuvent pas signaler au système de mise en file d'attente qu'elles sont hors ligne et, à la place, signalent qu'elles ne fonctionnent pas. Dans ce cas, ou si l'imprimante semble ne pas fonctionner, la file passera à l'état **DOWN**. Pour pallier un état **DOWN**, corrigez l'incident à l'origine du blocage de la file d'attente et demandez à l'administrateur système de remettre la file en marche. La file *doit* être remise en marche manuellement avant de pouvoir être réutilisée.
HELD	Indique qu'un travail d'impression est bloqué. Le spouleur ne pourra le traiter que lorsqu'il sera libéré.
QUEUED	Indique qu'un fichier figurant dans une file attend son tour pour être imprimé.
READY	Indique que tout ce qui concerne la file est prêt pour la mise en file d'attente et l'impression d'un travail.
RUNNING	Indique que l'impression d'un fichier est en cours.

Formatage des fichiers à imprimer (commande pr)

Utilisez la commande **pr** pour effectuer un formatage simple des fichiers à imprimer. Chaînez la sortie de la commande **pr** à la commande **qprt** pour formater votre texte.

Voici quelques indicateurs de la commande **pr** :

–d	Génère une sortie en double espacement.
–h *"Chaîne"*	Affiche la chaîne spécifiée, entre guillemets (" "), au lieu du nom du fichier, comme en–tête de page. Indicateur et chaîne doivent être séparés par un espace.
–l *Lignes*	Redéfinit la hauteur de page (66 lignes, par défaut) au nombre de *Lignes* indiqué. Si ce nombre est inférieur à la somme des lignes de l'en–tête et du bas de page (en nombre de lignes), ces derniers sont supprimés (même effet que l'indicateur **–t**).
–m	Fusionne des fichiers. La sortie standard est formatée de sorte que la commande **pr** écrive une ligne de chaque fichier spécifié par une variable *File*, côte à côte, en colonnes de texte de même taille (sur la base du nombre total de colonnes). Cet indicateur est incompatible avec l'indicateur – *Column*.
–n [*Largeur*] [*Caractère*]	Numérote les lignes selon le format spécifié par la variable *Largeur*. La valeur par défaut est de 5 chiffres. Si la variable *Caractère* (caractère non numérique quelconque) est spécifié, il est ajouté au numéro de ligne, pour marquer la séparation entre ce numéro et le reste de la ligne. Le caractère de séparation par défaut est le caractère ASCII Tab.
–o *Décalage*	Indente chaque ligne du nombre d'espaces indiqué par la variable *Décalage*. Le nombre total de caractères par ligne est la somme de la largeur de page et du décalage. La valeur par défaut de *Décalage* est 0.
–s *Caractère*	Sépare les colonnes par la variable *Caractère* au lieu d'insérer des espaces. Le caractère par défaut de *Caractère* est un caractère ASCII Tab.
–t	N'affiche ni les cinq lignes d'en-tête, ni les cinq lignes de bas de page. S'arrête à la dernière ligne de la page sans créer d'espace jusqu'à la fin de la page.
–w *Largeur*	Définit le nombre de colonnes par ligne à la valeur indiquée par la variable *Largeur*. La valeur par défaut est 72, générant une sortie multi-colonne de même taille. Aucune autre limite n'est imposée. Si l'indicateur **–w** n'est pas spécifié et que l'indicateur **–s** l'est, la largeur par défaut est de 512 colonnes.
– *Colonne*	Définit le nombre de colonnes à la valeur de la variable *Colonne*. La valeur par défaut est 1. Cette option est incompatible avec l'indicateur **–m**. Les indicateurs **–e** et **–i** sont supposés actifs pour les sorties multicolonnes. Une colonne de texte ne doit pas dépasser la largeur de la page (voir indicateur **–l**). Lorsque cet indicateur est combiné à l'indicateur **–t**, optez pour le nombre de lignes minimal pour la sortie.
+ *Page*	Commence par afficher le numéro de page spécifié par la variable *Page*. La valeur par défaut est 1.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **pr** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Voici une liste d'exemples d'utilisation des indicateurs de commande **pr** :

- Pour indiquer le titre du fichier **prog.c**, entrez :

```
pr –h "MAIN PROGRAM" prog.c | qprt
```

  prog.c est imprimé sous le titre `MAIN PROGRAM` (à la place du nom de fichier). Date de modification et numéro de page sont toujours imprimés.

- Pour imprimer le fichier **word.lst** sur plusieurs colonnes, entrez :

```
pr –3 word.lst | qprt
```

  Le fichier **word.lst** est imprimé sur trois colonnes verticales.

- Pour imprimer plusieurs fichiers côte à côte, entrez :

```
pr –m –h "Members and Visitors" member.lst visitor.lst | qprt
```

  Les fichiers **member.lst** et **visitor.lst** sont imprimés côte à côte sous le titre `Membres et visiteurs`.

- Pour modifier le fichier **prog.c** en vue d'un usage ultérieur, entrez :

```
pr –t –e prog.c > prog.notab.c
```

  Les tabulations du fichier **prog.c** sont remplacées par des espaces, et le résultat placé dans **prog.notab.c**. Une tabulation est placée aux colonnes 9, 17, 25, 33 et ainsi de suite. L'indicateur **–e** indique à la commande **pr** de remplacer les tabulations, l'indicateur **–t** supprime les en–têtes de page.

- Pour imprimer le fichier **myfile** sur deux colonnes, au format paysage et en texte de 7 points, entrez :

```
pr –l66 –w172 –2 myfile | qprt –z1 –p7
```

Impression de fichiers ASCII sur une imprimante PostScript

Le système de formatage de texte comprend le filtre enscript qui permet de convertir en PostScript des fichiers d'impression ASCII en vue de les imprimer sur une imprimante PostScript. Ce filtre est appelé par la commande **qprt –da** lors de l'envoi d'un travail d'impression vers une file d'attente PostScript.

Préalables

- L'imprimante doit être connectée physiquement au système.
- Elle doit être définie et configurée.
- La partie transcript des services de formatage de texte (Text Formatting Services) doit être installée.

Plusieurs indicateurs peuvent être spécifiés avec la commande **qprt** afin de personnaliser la sortie lors de l'envoi de fichiers ASCII vers une file d'attente d'impression PostScript :

–1+	Ajoute des en–têtes de page.
–2+	Formate la sortie en deux colonnes.
–3+	Imprime les en–têtes de pages, les dates et les numéros de page dans un style fantaisiste. On parle parfois de mode "gaudy".
–4+	Imprime le fichier, même s'il contient des caractères non imprimables.
–5+	Donne la liste des caractères qui ne font pas partie d'une police.
–h *chaîne*	Spécifie une chaîne à utiliser pour les en–têtes de page. Si ce paramètre n'est pas spécifié, l'en–tête se compose du nom de fichier, de la date de modification et du numéro de page.
–l *valeur*	Spécifie le nombre maximal de lignes imprimées par page. Selon la taille de caractères en points, le nombre de lignes par page peut être inférieur.
–L!	Tronque les lignes dépassant la largeur de la page.
–p	Spécifie la taille de caractères en points. Si ce paramètre n'est pas précisé, le système suppose que la taille de caractères est 10, sauf si le mode pivoté sur deux colonnes (**–2+ –z1**) est spécifié, auquel cas la valeur 7 est utilisée.
–s	Spécifie le style de police. Si ce paramètre n'est pas indiqué, c'est la police Courier qui est utilisée. L'imprimante PostScript doit avoir accès à la police spécifiée. Les polices possibles sont : • Courier–Oblique • Helvetica • Helvetica–Oblique • Helvetica–Narrow • Helvetica–Narrow–Oblique • NewCenturySchlbk–Italic • Optima • Optima–Oblique • Palatino–Roman • Palatino–Italic • Times–Roman • Times–Italic
–z1	Fait pivoter la sortie de 90 degrés (mode paysage).

Voici une liste d'exemples d'utilisation des indicateurs de commande **qrpt** :

- Pour envoyer le fichier ASCII **myfile.ascii** vers l'imprimante PostScript nommée `Msps1`, entrez :

```
qprt -da -PMsps1 myfile.ascii
```

- Pour envoyer le fichier ASCII **myfile.ascii** vers l'imprimante PostScript nommée `Msps1` et l'imprimer en police Helvetica, entrez :

```
qprt -da -PMsps1 -sHelvetica myfile.ascii
```

- Pour envoyer le fichier ASCII **myfile.ascii** vers l'imprimante PostScript nommée `Msps1` et l'imprimer avec la taille de caractère 9, entrez :

```
qprt -da -PMsps1 -p9 myfile.ascii
```

Automatisation de la conversion ASCII–PostScript

La plupart des applications qui génèrent des fichiers d'impression PostScript suivent la convention d'appellation qui veut qu'un fichier d'impression PostScript commence toujours par les caractères `%!`. Pour configurer le système de sorte qu'il repère les fichiers d'impression ASCII soumis à une file d'attente PostScript et qu'il les convertisse automatiquement en fichiers PostScript, procédez comme suit.

1. A l'invite, entrez ce qui suit :

```
smit chpq
```

2. Indiquez une file d'attente PostScript ou sélectionnez–en une à l'aide de la fonction List.
3. Sélectionnez l'option **Printer Setup**.
4. Basculez la valeur de l'option **AUTOMATIC detection of print file TYPE to be done?** sur **yes**.

Les commandes suivantes sont alors équivalentes : elles convertissent tout fichier ASCII en fichier PostScript et l'impriment sur une imprimante PostScript. Ainsi, pour convertir et imprimer le fichier **monfichier.ascii**, entrez l'une des commandes suivantes :

```
qprt -Pps monfichier.ps monfichier.ascii
lpr -Pps monfichier.ps monfichier.ascii
lp -dps monfichier.ps monfichier.ascii
```

où `ps` est une file d'attente d'impression PostScript.

Annulation de la détermination automatique des types de fichiers d'impression

Vous devrez peut–être annuler la détermination automatique des types de fichiers d'impression pour l'impression PostScript dans les cas suivants :

- Pour imprimer un fichier PostScript nommé **myfile.ps** qui ne commence pas par `%!`, entrez ce qui suit sur la ligne de commande :

```
qprt -ds -Pps myfile.ps
```

- Pour imprimer le listing source d'un fichier PostScript nommé **myfile.ps** qui commence par `%!`, entrez ce qui suit sur la ligne de commande :

```
qprt -da -Pps myfile.ps
```

Récapitulatif des commandes relatives à l'impression

annuler	Annule les demandes sur une imprimante ligne.
lp	Envoie des demandes à une imprimante ligne.
lpq	Examine la file d'attente d'impression.
lpr	Met en file d'attente les travaux d'impression.
lprm	Supprime des travaux d'impression de l'imprimante ligne.
lpstat	Affiche des informations sur l'état de l'imprimante ligne.
pr	Envoie un fichier vers l'unité de sortie standard.
qcan	Annule un travail d'impression.
qchk	Affiche l'état d'une file d'attente d'impression.
qhld	Bloque/libère un travail d'impression.
qmov	Déplace un travail vers une autre file d'attente.
qpri	Définit la priorité d'un travail dans la file d'attente.
qprt	Lance un travail d'impression.

Chapitre 9. Fichiers de sauvegarde et supports de stockage

Une fois que votre système fonctionne, il vous faut réfléchir aux moyens de sauvegarder systèmes de fichiers, répertoires et fichiers. Sachant qu'il est toujours possible d'effacer ou de modifier un fichier – intentionnellement ou accidentellement, adopter une stratégie de sauvegarde efficace s'impose. En cas de problème, vous aurez ainsi toujours la ressource de restaurer la version la plus récente de vos fichiers et systèmes de fichiers.

> **Remarque :** Si un disque "plante", toutes les données qu'il contient sont définitivement détruites. Le seul moyen de recouvrer ces données est de les restaurer à partir d'une copie de sauvegarde.

Il existe différentes méthodes pour sauvegarder vos systèmes fichiers. Le plus souvent, vous opterez pour une sauvegarde simple, c'est–à–dire une copie du fichier, du répertoire ou du système de fichiers, permettant un transfert ou une restauration en cas de destruction ou de modification accidentelle. Il est également possible d'archiver une copie d'un ou de plusieurs fichiers, ou d'une base de données, pour un usage ultérieur, à des fins d'historique ou pour recouvrer les données en cas de perte ou de détérioration.

Ce chapitre traite des points suivants :

Etablissement d'une stratégie de sauvegarde

Il n'existe pas une, mais plusieurs stratégies de sauvegarde – les utilisateurs ayant chacun leurs impératifs et leurs souhaits. Une approche adaptée à un système mono-utilisateur a peu de chances de convenir sur un système servant 5 ou 10 utilisateurs. De même, une stratégie adaptée à un système où les fichiers sont modifiés quotidiennement sera inefficace sur un système où la fréquence de modification des données est faible. Vous seul êtes à même de déterminer la stratégie qui convient le mieux.

Garantissez-vous contre les pertes majeures.

Votre système peut–il continuer à fonctionner si un seul disque tombe en panne ? Pouvez–vous recouvrer vos données si tous les disques tombent en panne ? Pouvez–vous reconstituer votre système si vous perdez vos disquettes ou vos bandes de sauvegarde, qu'elles sont brûlées ou volées ? Réfléchissez donc au moyen de recouvrer l'usage de votre système dans chacun de ces cas de figure.

Vérifiez régulièrement vos sauvegardes.

Supports et unités de sauvegarde, même les plus fiables, ne sont pas à l'abri d'une défaillance. Une bibliothèque conséquente de bandes ou de disquettes n'est d'aucune utilité si vous ne pouvez les relire et les transférer sur un disque. Pour vérifier que vos sauvegardes sont exploitables, affichez régulièrement leur table des matières (via **restore –T**, ou **tar –t** pour les bandes d'archives). Si vous effectuez vos sauvegardes sur disquettes et que vous disposez de plusieurs unités de disquettes, tentez de les lire sur une unité autre que celle qui a servi à créer la sauvegarde. Vous pouvez également créer un second lot de disquettes de sauvegarde de niveau 0. Si vous utilisez un dérouleur en continu, lancez régulièrement la commande **tapechk** pour effectuer un contrôle de cohérence rudimentaire sur la bande.

Conservez les anciens supports de sauvegarde.

Prévoyez de recycler régulièrement les supports de sauvegarde, mais pas *tous*. Il peut parfois s'écouler plusieurs mois avant que vous (ou un autre utilisateur du système) vous aperceviez qu'un fichier important a disparu ou qu'il est endommagé. Vous serez alors soulagé de savoir qu'il en existe une sauvegarde. Voici un exemple de planning de recyclage de disquettes et des bandes de sauvegarde :

- Une fois par semaine, recyclage de toutes les disquettes de sauvegarde quotidiennes, excepté celles du vendredi.
- Une fois par mois, recyclez toutes les disquettes du vendredi, excepté celle du dernier vendredi du mois. Ainsi, les quatre dernières sauvegardes du vendredi sont toujours disponibles.
- Une fois par trimestre, recyclage de toutes les disquettes mensuelles, excepté celles du dernier mois. Conservez indéfiniment la dernière disquette de chaque mois, éventuellement dans un lieu distinct.

Vérifiez les systèmes de fichiers avant de les sauvegarder.

Une sauvegarde effectuée sur un système de fichiers endommagé sera probablement inutilisable. Avant d'effectuer une sauvegarde, prenez l'habitude de vérifier l'intégrité du système de fichiers à l'aide de la commande **fsck**.

Vérifiez que les fichiers que vous sauvegardez ne sont pas en cours d'utilisation.

Le système ne doit pas être en cours d'exploitation lorsque vous lancez vos sauvegardes. Si c'est le cas, les fichiers risquent d'être modifiés, et votre sauvegarde de ne pas être à jour.

Sauvegardez votre système avant d'y effectuer des changements majeurs.

Il est toujours plus prudent de sauvegarder l'intégralité du système avant de tester un élément matériel, de procéder à une réparation ou d'installer des nouvelles unités, de nouveaux programmes ou autres.

Divers

Lorsque vous planifiez et implémentez une stratégie de sauvegarde, n'oubliez pas les points suivants :

- La fréquence de modification des données. Les données du système d'exploitation ne changeant pas très souvent, vous ne devez pas effectuer fréquemment les sauvegardes. Les données utilisateur, à l'inverse, sont en général souvent modifiées et vous devez les sauvegarder en conséquence.
- Le nombre d'utilisateurs sur le système. Le volume à sauvegarder et la fréquence des sauvegarde en dépendent directement.
- La difficulté de recréer les données. Vous devez savoir qu'il est impossible de recréer certaines données s'il n'y a pas de sauvegarde.

Ainsi, une stratégie de sauvegarde afin de conserver vos données revêt une importance capitale. En évaluant les besoins de votre site, vous pourrez identifier la méthode de sauvegarde la mieux adaptée à vos besoins. Prévoyez des sauvegardes fréquentes et régulières des données utilisateurs. Recouvrer des données perdues peut s'avérer difficile si vous n'avez pas défini de stratégie de sauvegarde.

Supports de stockage

Il existe différents types de supports de stockage. Les supports disponibles dépendent de la configuration matérielle et logicielle de votre système. Les supports les plus utilisés sont les disquettes 5 1/4 pouces, les bandes 8 mm, les bandes 9 pistes et les disquettes 3 1/2 pouces.

Avertissement : Exécuter la commande **backup** détruit toutes les données antérieurement stockées sur le support de sauvegarde sélectionné.

Disquettes

Les disquettes sont le support de sauvegarde standard. C'est–à–dire que, sauf spécification autre de votre part via la commande **backup –f**, la commande **backup** envoie automatiquement sa sortie vers l'unité **/dev/rfd0**, à savoir l'unité de disquette. Pour effectuer une sauvegarde sur l'unité de bande par défaut, entrez **/dev/rmt0** et appuyez sur Entrée.

Maniez les disquettes avec soin. Chaque élément d'information occupe un espace tellement infime que la moindre rayure, poussière, miette, particule de tabac... peut rendre l'information inutilisable. N'oubliez pas les points suivants :

- Ne touchez pas les surfaces d'enregistrement.
- Conservez les disquettes à l'abri de toute source de champ magnétique et électromagnétique (téléphone, dictaphone, calculatrice, etc.).
- Evitez les températures extrêmes (plage recommandée : entre 10°C et 60°C).
- Prenez soin des disquettes pour prévenir toute perte de données.
- Effectuez régulièrement des copies de sauvegarde.

Avertissement : Disquettes et unité de disquette doivent être du même type, faute de quoi vous risquez de perdre des données. Si vous introduisez une disquette incorrecte dans l'unité de disquette 3 1/2 pouces, les données de la disquette risquent d'être détruites.

Les disquettes 3 1/2 pouces compatibles sont les suivantes :

- 1 Mo de capacité (soit 720 Ko de données)
- 2 Mo de capacité (soit 1,44 Mo de données)

Bandes

En raison de leur capacité élevée et de leur durée de vie, les bandes sont souvent le support choisi pour stocker des fichiers de grande taille ou une grande quantité de fichiers, comme des copies d'archive de systèmes de fichiers. Elles servent également à transférer des fichiers d'un système à un autre. Elles sont en revanche peu adaptées au stockage de fichiers souvent utilisés, d'autres supports offrant des temps d'accès bien inférieurs.

Les fichiers bande sont créés via des commandes telles que **backup**, **cpio** et **tar**, qui ouvrent une unité de bande, y écrivent des données et la ferment.

Formatage de disquettes (commande format ou fdformat)

Avertissement : Le formatage d'une disquette détruit toutes les données qu'elle contient.

Vous pouvez formater des disquettes dans l'unité spécifiée par le paramètre *Unité* (**/dev/rfd0,** par défaut) via les commandes **format** et **fdformat**. La commande **format** détermine le type d'unité de disquette :

- Disquette 5,25 pouces à faible densité (360 Ko) comprenant 40 x 2 pistes, de 9 secteurs chacune
- Disquette 5,25 pouces à haute densité (1,2 Mo) comprenant 80 x 2 pistes, de 15 secteurs chacune
- Disquette 3,5 pouces à faible densité (720 Ko) comprenant 80 x 2 pistes, de 9 secteurs chacune
- Disquette 3,5 pouces à haute densité (2,88 Mo) comprenant 80 x 2 pistes, de 36 secteurs chacune

Tous les secteurs ont une taille de 512 octets (quel que soit le type de disquette).

Sauf spécification contraire via le paramètre *Unité*, la commande **format** formate une disquette en haute densité.

Utilisez la commande **fdformat** pour formater une disquette en faible densité, sauf si l'indicateur **–h** est spécifié. Le paramètre *Unité* indique l'unité où se trouve la disquette à formater (**/dev/rfd0** pour l'unité 0, par exemple).

Avant de formater une disquette, les commandes **format** et **fdformat** demandent confirmation. Vous pouvez ainsi annuler l'opération au besoin.

Voir les exemple ci–après :

- Pour formater une disquette dans l'unité **/dev/rfd0**, entrez :

```
format –d /dev/rfd0
```

- Pour formater une disquette sans vérifier les secteurs, entrez :

```
format –f
```

- Pour formater une disquette 360 ko dans une unité 5 1/4 pouces, 1,2 Mo de l'unité **/dev/rfd1**, entrez :

```
format –l –d /dev/rfd1
```

- Pour forcer le formatage en haute densité d'une disquette avec la commande **fdformat**, entrez :

```
fdformat –h
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **format** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Vérification de l'intégrité du système de fichiers (commande fsck)

Utilisez la commande **fsck** pour vérifier la cohérence du système de fichiers et remédier aux incohérences de façon interactive. Il est important d'exécuter cette commande sur chaque système de fichiers dans le cadre de l'initialisation du système. Vous devez pouvoir lire le fichier unité sur lequel réside le système de fichiers (unité **/dev/hd0,** par exemple). Généralement, le système de fichiers est cohérent et la commande **fsck** se contente d'indiquer le nombre de fichiers, de blocs utilisés et de blocs libres. Dans le cas contraire, la commande **fsck** signale les incohérences et vous demande l'autorisation d'y remédier. La commande **fsck** n'intervient que prudemment en évitant toute action qui provoquerait une perte de données valides. Toutefois, dans certains cas, **fsck** recommande la destruction d'un fichier endommagé.

Attention : Lancez toujours la commande **fsck** après un incident système. L'intervention peut détruire quelques données. L'action par défaut de chaque correction de cohérence consiste à attendre que l'opérateur entre `oui`ou `non`. Si vous ne disposez pas des droits d'*écriture* sur un fichier affecté, la commande **fsck** utilise par défaut la réponse `non`.

Voir les exemple ci–après :

- Pour vérifier tous les systèmes de fichiers par défaut, entrez :

```
fsck
```

Sous cette forme, la commande **fsck** vous demande confirmation avant toute modification sur un système de fichiers.

- Pour réparer automatiquement des problèmes mineurs avec les systèmes de fichiers par défaut, entrez :

```
fsck -p
```

- Pour vérifier le système de fichiers **/dev/hd1**, entrez :

```
fsck /dev/hd1
```

Le système de fichiers non monté se trouvant sur l'unité `/dev/hd1` est vérifié.

Remarque : La commande **fsck** n'opère aucune correction sur un système de fichiers monté.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **fsck** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Copie vers et à partir de disquettes (commande flcopy)

Utilisez la commande **flcopy** pour copier une disquette (ouverte comme **/dev/rfd0**) vers un fichier nommé **floppy** créé dans le répertoire en cours. Le message `Change floppy, hit return when done` s'affiche si nécessaire.
La commande **flcopy** copie alors le fichier **floppy** sur la disquette.

Voir les exemple ci–après :

- Pour copier **/dev/rfd1** dans le fichier **floppy** dans le répertoire en cours, entrez :

```
flcopy -f /dev/rfd1 -r
```

- Pour copier les 100 premières pistes de la disquette, entrez :

```
flcopy -f /dev/rfd1 -t 100
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **flcopy** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Copie de fichiers sur des bandes ou des disques (commande cpio –o)

Utilisez la commande **cpio –o** pour lire les chemins d'accès sur l'entrée standard et copier ces fichiers sur la sortie standard, avec les chemins d'accès et les informations d'état. Les chemins d'accès ne peuvent dépasser 128 caractères. Evitez d'utiliser des noms de chemins composés uniquement de fichiers liés avec la commande **cpio**. Si la mémoire est insuffisante, vous risquez de perdre des informations de liaison.

Reportez–vous aux exemple ci–après :

- Pour copier sur une disquette les fichiers du répertoire courant se terminant par **.c**, entrez :

```
ls *.c | cpio –ov >/dev/rfd0
```

  L'indicateur **–v** affiche le nom de chaque fichier.

- Pour copier sur une disquette le répertoire courant et ses sous–répertoires, entrez :

```
find . –print | cpio –ov >/dev/rfd0
```

  L'arborescence dont le sommet est le répertoire courant (.) est sauvegardée, avec tous ses sous–répertoires et ses fichiers.

- Pour utiliser une chaîne de commande plus brève, entrez :

```
find . –cpio /dev/rfd0 –print
```

  L'entrée –print affiche le nom de chaque fichier à mesure qu'il est copié.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **cpio** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Copie de fichiers à partir d'une bande ou d'un disque (commande cpio –i)

Utilisez la commande **cpio –i** pour lire à partir de l'entrée standard un fichier d'archive créé à l'aide de la commande **cpio –o** et copier à partir de celui–ci les fichiers dont les noms correspondent au paramètre *Modèle*. Ces fichiers sont copiés dans l'arborescence de répertoires en cours. Il est possible d'indiquer plusieurs paramètres *Modèle*, en respectant la notation de noms de fichiers décrite dans la commande **ksh**. La valeur par défaut du paramètre *Modèle* est un astérisque (`*`) qui permet de sélectionner tous les fichiers du répertoire en cours. Dans une expression telle que `[a-z]`, le tiret (`-`) signifie *jusqu'à,* conformément à la séquence de tri en cours.

Remarque : Les modèles `"*.c"` et `"*.o"` doivent être placés entre guillemets pour éviter que le shell ne traite l'astérisque (*) comme un caractère joker. Il s'agit d'un cas particulier dans lequel la commande **cpio** elle–même décode le caractère joker.

Voir les exemple ci–après :

- Pour afficher la liste des fichiers qui ont été enregistrés sur une disquette avec la commande **cpio**, entrez :

```
cpio -itv </dev/rfd0
```

  Cette commande affiche la table des matières des données préalablement enregistrées dans le fichier **/dev/rfd0** dans le format de la commande **cpio**. La liste est semblable à la liste de répertoire longue produite par la commande **ls –l**.

- Pour n'afficher que les noms de chemin des fichiers, n'employez que les paramètres **–it**.
- Pour copier, à partir d'une disquette, les fichiers préalablement enregistrés avec la commande **cpio**, entrez :

```
cpio -idmv </dev/rfd0
```

  Cette commande copie les fichiers préalablement enregistrés dans le fichier **/dev/rfd0** à l'aide de la commande **cpio** dans le système de fichiers (spécifiez le paramètre **–i**). Le paramètre **–d** permet à la commande **cpio** de créer les répertoires appropriés si une arborescence de répertoires est enregistrée. Le paramètre **–m** conserve la dernière heure de modification qui était en vigueur au moment de la sauvegarde des fichiers. Le paramètre **–v** entraîne l'affichage des noms des fichiers au fur et à mesure qu'ils sont copiés.

- Pour copier des fichiers sélectionnés à partir d'une disquette, entrez :

```
cpio -i "*.c" "*.o" </dev/rfd0
```

  Cette commande copie les fichiers se terminant par **.c** ou **.o** à partir d'une disquette.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **cpio** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Copie sur et à partir de bandes (commande tcopy)

Utilisez la commande **tcopy** pour effectuer des copies à partir de bandes magnétiques et sur des supports magnétiques.

Par exemple, pour copier à partir d'une bande continue sur une bande 9 pistes, entrez :

```
tcopy /dev/rmt0 /dev/rmt8
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **tcopy** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Vérification de l’intégrité d’une bande (commande tapechk)

Vous pouvez effectuer un contrôle de cohérence rudimentaire sur une unité de bande connectée à l’aide de la commande **tapechk**. Certains dysfonctionnements d’une unité de bande peuvent être détectés simplement en lisant une bande. La commande **tapechk** offre une solution pour effectuer des lectures de bande au niveau fichier.

Par exemple, pour vérifier les trois premiers fichiers sur une unité de bande, entrez :

```
tapechk 3
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **tapechk** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Compression des fichiers (commandes compress et pack)

Utilisez la commande **compress** et la commande **pack** pour compresser des fichiers en vue de leur stockage.

Utilisez la commande **uncompress** et la commande **unpack** pour décompresser des fichiers restaurés.

Le processus de compression et de décompression de fichiers prend du temps mais, une fois compressées, les données prennent moins de place sur le support de stockage.

Pour comprimer un système de fichiers, utilisez l'une des méthodes suivantes :

- L'option **–p** associée à la commande **backup**.
- Les commandes **compress** et **pack**.

Avantages de la compression de fichiers :

- Des coûts réduits et l'économie de temps : compression de données avant leur transfert via un réseau.
  - Compression des systèmes de fichiers avant d'effectuer des sauvegardes pour économiser l'espace sur bande.
  - Compression des fichiers journaux créés par les shell scripts qui fonctionnent la nuit ; il est simple de faire compresser le fichier par le script avant qu'il ne se termine.
  - Compression des fichiers ne faisant l'objet d'aucun accès. Par exemple, les fichiers appartenant à utilisateur qui est absent pour une période prolongée peuvent être compressés et placés dans une archive **tar** sur disque ou sur bande afin d'être restaurés ultérieurement.
- Economie financière et gain de temps, du fait de la compression des fichiers avant leur envoi sur un réseau.

Remarques :

1. La commande **compress** peut se trouver à court d'espace de travail dans le système de fichiers pendant la compression. La commande compress crée les fichiers compressés avant de supprimer tout fichier décompressé. Elle a donc besoin d'un espace de travail représentant 50 % de place supplémentaire par rapport à la taille totale des fichiers.
2. La compression d'un fichier peut échouer parce qu'il est déjà compressé. Si la commande **compress** ne peut réduire la taille des fichiers, elle échoue.

Compression de fichiers avec la commande compress

Utilisez la commande **compress** pour réduire la taille des fichiers à l'aide du codage adaptatif Lempel–Zev. Chaque fichier original spécifié par le paramètre *Fichier* est remplacé par un fichier compressé au nom duquel est ajouté le suffixe **.Z**. Le fichier compressé conserve les caractéristiques suivantes du fichier original : propriétaire, modes, heures d'accès et de modification. Si aucun fichier n'est spécifié, l'entrée standard est compressée vers la sortie standard. Si la compression ne permet pas de réduire la taille d'un fichier, un message est envoyé vers la sortie d'erreur standard et le fichier d'origine n'est pas remplacé.

Utilisez la commande **uncompress** pour rétablir les fichiers compressés dans leur forme d'origine.

Le degré de compression dépend de la taille du fichier d'entrée, du nombre de bits par code spécifié par la variable *Bits*, et de la fréquence de chaînes communes. En général le code source ou le texte anglais sont réduits de 50 à 60 %. La compression produite par la commande **compress** est généralement plus compacte et nécessite moins de temps de calcul que celle de la commande **pack**, qui utilise le codage adptatif Huffman.

Par exemple, pour compresser le fichier **foo** et écrire le pourcentage de compression sur la sortie d'erreur standard, entrez :

```
compress –v foo
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **compress** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Compression de fichiers avec la commande pack

Utilisez la commande **pack** pour enregistrer le ou les fichiers spécifiés par le paramètre *Fichier* sous une forme compressée, à l'aide du codage Huffman. Le fichier d'entrée est remplacé par un fichier compressé dont le nom est dérivé du nom de fichier d'origine (*Fichier***.z**), et qui possède les mêmes modes d'accès, dates d'accès et de modification, et propriétaire que le fichier d'origine. Le nom du fichier d'entrée ne peut comporter plus de 253 octets, afin de permettre l'ajout du suffixe **.z**. Si la commande **pack** réussit, le fichier d'origine est supprimé.

Utilisez la commande **unpack** pour rétablir les fichiers compressés dans leur forme d'origine.

Si la commande **pack** ne peut pas créer un fichier plus petit, elle cesse son traitement et signale qu'elle ne peut économiser de l'espace. (L'impossibilité de gagner de la place se produit généralement avec les fichiers de petite taille, ou ceux ayant une distribution de caractères uniforme.) Le gain d'espace réel dépend de la taille du fichier d'entrée et de la fréquence de distribution des caractères. Du fait qu'un arbre de décodage constitue la première partie de chaque fichier **.z**, vous ne gagnez pas de place avec les fichiers inférieurs à trois blocs. En général, les fichiers texte sont réduits de 25 à 40 %.

La valeur de sortie de la commande **pack** est le nombre de fichiers qu'elle n'a pas pu compresser. La compression n'est pas effectuée dans les cas suivants :

- Le fichier est déjà compressé.
- Le nom du fichier d'entrée possède plus de 253 octets.
- Le fichier contient des liens.
- Le fichier est un répertoire.
- Le fichier ne peut être ouvert.
- La compression n'économise pas de blocs de stockage.
- Un fichier nommé *Fichier***.z** existe déjà.
- Le fichier **.z** ne peut pas être créé.
- Une erreur d'E/S s'est produite pendant le traitement.

Par exemple, pour compresser les fichiers **chap1** et **chap2**, entrez :

```
pack chap1 chap2
```

Cette commande compresse **chap1** et **chap2**, et les remplace par des fichiers nommés **chap1.z** et **chap2.z**. La commande **pack** affiche le pourcentage de diminution de taille de chaque fichier.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **pack** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Décompression de fichiers (commandes uncompress et unpack)

Utilisez les commandes **uncompress** et **unpack** pour décompresser des fichiers.

Utilisation de la commande uncompress

Utilisez la commande **uncompress** pour décompresser les fichiers précédemment compressés à l'aide de la commande **compress**. Tout fichier compressé spécifié par la variable *Fichier* est remplacé par sa version développée. Cette version porte le même nom que la version compressée, sans l'extension **.Z**. Ce fichier conserve les attributs du fichier d'origine (propriétaire, droits d'accès, date et heure de dernière modification, etc.). A défaut de fichier(s) spécifié(s), l'entrée standard est décompressée sur la sortie standard.

Semblable à la commande **uncompress**, la commande **zcat** écrit toujours la sortie développée sur l'unité de sortie standard.

- Par exemple, pour décompresser le fichier **foo**, entrez :

```
uncompress foo
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **uncompress** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Utilisation de la commande unpack

Utilisez la commande **unpack** pour décompresser les fichiers créés à l'aide de la commande **pack**. Pour chaque fichier spécifié, la commande **unpack** recherche le fichier *Fichier***.z** compressé. Si ce fichier est comprimé, **unpack** le remplace par sa version décompressée. La commande **unpack** renomme le nouveau fichier en supprimant le suffixe **.z** de *Fichier*. Le nouveau fichier conserve les attributs du fichier d'origine (propriétaire, modes, heures d'accès et de modification, etc.).

La commande **unpack** n'opère que sur les fichiers suffixés **.z**. Aussi, si vous spécifiez un nom de fichier ne se terminant pas par **.z**, la commande **unpack** lui ajoute le suffixe et recherche dans le répertoire un fichier avec ce suffixe.

La valeur en sortie de la commande **unpack** est le nombre de fichiers qu'elle n'a pu décompresser. Un fichier ne peut pas être décompressé dans les situations suivantes :

- Le nom du fichier (hors extension **.z**) dépasse 253 octets.
- Le fichier ne peut être ouvert.
- Le fichier n'est pas compressé.
- Un fichier décompressé portant le même nom existe déjà.
- Le fichier décompressé ne peut être créé.

  Remarque : La commande **unpack** inscrit un avertissement sur l'unité de sortie standard si le fichier qu'elle décompresse est doté de liens. Le fichier nouvellement décompressé comporte un numéro d'i-node (numéro de référence d'i-node) différent du fichier compressé d'origine. Toutefois, les autres fichiers liés à l'i-node du fichier compressé d'origine existent toujours et sont toujours comprimés.

Par exemple, pour décompresser les fichiers **chap1.z** et **chap2.z**, entrez :

```
unpack chap1.z chap2
```

Les fichiers **chap1.z** et **chap2.z** sont décompressés et remplacés par les fichiers **chap1** et **chap2**.

Remarque : Vous pouvez utiliser la commande **unpack** avec des noms de fichiers comportant le suffixe **.z** ou un autre suffixe.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **unpack** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Sauvegarde de fichiers (commande backup)

Attention : Toute tentative de sauvegarde d'un système de fichiers monté génère un message. La commande **backup** poursuit son traitement mais il peut en résulter des incohérences dans le système de fichiers. Cet avertissement ne concerne pas le système de fichiers racine (**/**).

Utilisez la commande **backup** ou la commande **smit** pour créer des copies de vos fichiers sur des supports de sauvegarde tels qu'une bande magnétique ou une disquette. Ces copies adoptent l'un des deux formats de sauvegarde suivants :

- Fichiers spécifiques sauvegardés par noms, via l'indicateur **–i**.
- Intégralité de systèmes de fichiers sauvegardés par numéros d'i-node, via les paramètres *–Level* et *FileSystem*.

Remarques :

1. Il y a toujours un risque d'endommagement des données lorsqu'un fichier est modifié pendant la sauvegarde du système. Réduisez donc au minimum les activités sur le système pendant la procédure de sauvegarde.
2. Si vous effectuez une sauvegarde sur une bande 8 mm alors que la taille de bloc de l'unité est défini à 0 (zéro), il sera impossible de procéder à une restauration directement à partir de la bande. Dans ce cas, reportez-vous aux procédures particulières décrites dans la section relative à la commande **restore.**

Attention : Vérifiez que les indicateurs que vous spécifiez sont compatibles avec le support de sauvegarde.

Sauvegarde de fichiers (commande backup)

Utilisez la commande **backup** pour créer des copies de fichiers sur des supports de sauvegarde.

Par exemple, pour sauvegarder par noms des fichiers de votre répertoire **$HOME**, entrez :

```
find $HOME –print | backup –i –v
```

L'indicateur **–i** indique au système de lire sur l'entrée standard les noms des fichiers à sauvegarder. La commande **find** génère une liste des fichiers du répertoire de l'utilisateur. Cette liste devient, par chaînage, l'entrée standard de la commande **backup**. L'indicateur **–v** affiche un état de la progression de la sauvegarde. Les fichiers sont sauvegardés sur l'unité de sauvegarde par défaut du système local.

Voir les exemple ci–après :

- Pour sauvegarder le système de fichiers racine, entrez :

```
backup –0 –u /
```

  Le niveau (0) et la barre oblique **(/)** instruisent le système qu'il s'agit de sauvegarder le système de fichiers racine **(/)**. La sauvegarde a lieu dans le fichier **/dev/rfd0**. L'indicateur **–u** signifie au système qu'il doit mettre à jour l'article relatif au niveau de sauvegarde courant, dans le fichier **/etc/dumpdates.**

- Pour sauvegarder tous les fichiers du système de fichiers racine **(/),** modifiés depuis la dernière sauvegarde de niveau 0, entrez :

```
backup –1 –u /
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **backup** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Sauvegarde de fichiers (commande smit)

Utilisez la commande **smit** pour exécuter la commande **backup** qui permet de créer des copies de vos fichiers sur des supports de sauvegarde.

1. A l'invite, entrez ce qui suit :

```
smit backup
```

2. Entrez le chemin d'accès au répertoire sur lequel est normalement monté le système de fichiers, dans le champ **chemin d'accès complet au REPERTOIRE** :

```
/home/bill
```

3. Dans les champs unité ou **FICHIER** DE **SAUVEGARDE**, entrez le nom de l'unité de sortie (ci–dessous le nom d'une unité de bande magnétique) :

```
/dev/rmt0
```

4. Appuyez sur Tab pour basculer la valeur du champ facultatif **SIGNALEMENT de chaque étape de la sauvegarde**, si vous souhaitez afficher des messages d'erreur.
5. Dans un environnement de gestion de système, observez la valeur par défaut du champ **Nombre MAX. de blocs à écrire sur la sauvegarde** : ce champ ne concerne pas les sauvegardes sur bande.
6. Appuyez sur Entrée pour sauvegarder le répertoire ou le système de fichiers nommé.
7. Exécutez la commande **restore –t**. Si elle génère un message d'erreur, la sauvegarde est à recommencer.

Restauration de fichiers (commande restore)

Utilisez la commande **restore** ou la commande **smit** pour lire des fichiers écrits à l'aide de la commande **backup** à partir du support de sauvegarde et pour les restaurer sur votre système local.

Remarques :

1. La restauration doit être effectuée via la même méthode que celle utilisée pour la sauvegarde. Par exemple, un système de fichiers sauvegardé par noms doit être restauré par noms.
2. Lorsque la sauvegarde occupe plusieurs disquettes, la commande **restore** lit celle qui est en place, puis vous invite à insérer la suivante. Appuyez sur Entrée une fois la disquette insérée, pour poursuivre la restauration.

Utilisation de la commande restore

Utilisez la commande **restore** pour lire des fichiers écrits à l'aide de la commande **backup** et pour les restaurer sur votre système local.

Voir les exemple ci–après :

- Pour afficher la liste des fichiers précédemment sauvegardés, entrez :

```
restore –T
```

  Les informations sont lues à partir de l'unité de sauvegarde par défaut **/dev/rfd0**. S'il s'agit de fichiers isolés, seul leur nom est affiché. S'il s'agit de systèmes de fichiers entiers, leur numéro d'i-node apparaît également.

- Pour restaurer des fichiers sur le système de fichiers principal, entrez :

```
restore –x –v
```

  L'indicateur **–x** extrait tous les fichiers du support de sauvegarde et les restaure à leur emplacement dans le système de fichiers. L'indicateur **–v** affiche un état de la progression des opérations à mesure que les fichiers sont restaurés. Dans le cas de la restauration d'un système de fichiers tout entier, les fichiers sont affichés avec leur numéro d'i-node. Sinon, seul leur nom est affiché.

- Pour copier le fichier **/home/mike/manual/chap1**, entrez :

```
restore –xv /home/mike/manual/chap1
```

  Cette commande extrait le fichier **/home/mike/manual/chap1** du support de sauvegarde et le restaure. Le nom de fichier **/home/mike/manual/chap1** doit pouvoir être affiché à l'aide de la commande **restore –T**.

- Pour copier tous les fichiers dans le répertoire **manual**, entrez :

```
restore –xdv manual
```

  Le répertoire **manual** et les fichiers qu'il contient sont alors restaurés. Si le répertoire n'existe pas, un répertoire **manual** est créé dans le répertoire en cours afin de recevoir les fichiers restaurés.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **restore** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Utilisation de la commande smit

Utilisez la commande **smit** pour exécuter la commande **restore**, qui lit des fichiers écrits par la commande **backup** et les restaure sur votre système local.

1. A l'invite, entrez ce qui suit :

```
smit restore
```

2. Renseignez la zone **REPERTOIRE cible**. Il s'agit du répertoire dans lequel vous souhaitez que les fichiers soient restaurés.
3. Passez au champ **UNITE** ou **FICHIER DE SAUVEGARDE**, entrez le nom de l'unité de sortie (ci–dessous le nom d'une unité de bande magnétique) :

```
/dev/rmt0
```

   Si l'unité n'est pas disponible, un message semblable au suivant s'affiche :

```
Cannot open /dev/rmtX, no such file or directory.
```

   Ce message indique que le système ne peut pas contacter le pilote de l'unité car il n'y a pas de fichier correspondant à **rmtX** dans le répertoire **/dev**. Seuls les éléments dont l'état est `disponible` sont dans le répertoire **/dev**.

4. Pour le champ **NBRE de blocs à lire en une entrée**, nous vous conseillons de conserver la valeur par défaut.
5. Appuyez sur Entrée pour restaurer le système de fichiers ou le répertoire spécifié.

Archivage de fichiers (commande tar)

Il est également possible d'archiver une copie d'un ou de plusieurs fichiers, ou d'une base de données, pour un usage ultérieur, à des fins d'historique ou pour recouvrer les données en cas de perte ou de détérioration. Une archive est généralement utilisée lorsque ces données spécifiques sont supprimées du système.

Utilisez la commande **tar** pour écrire des fichiers ou les extraire d'un fichier archive. Sauf spécification contraire, la commande **tar** recherche les archives sur l'unité par défaut (bande, généralement).

Lorsque vous écrivez dans un fichier archive, la commande **tar** passe par un fichier temporaire (fichier **/tmp/tar***) et garde en mémoire une table des fichiers dotés de plusieurs liens. Un message d'erreur est émis si la commande **tar** ne peut créer ce fichier temporaire ou si la mémoire disponible est insuffisante pour contenir les tables de liens.

Voir les exemple ci–après :

- Pour écrire les fichiers **file1** et **file2** dans un nouveau fichier archive, sur l'unité de bande par défaut, entrez :

```
tar –c file1 file2
```

- Pour extraire tous les fichiers du répertoire **/tmp** du fichier archive de l'unité de bande **/dev/rmt2** et imputer la durée de l'opération au temps de modification, entrez :

```
tar –xm –f/dev/rmt2 /tmp
```

- Pour afficher le nom des fichiers du fichier archive sur disque **out.tar** à partir du répertoire courant, entrez :

```
tar –vtf out.tar
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **tar** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Récapitulatif des commandes relatives à la sauvegarde

sauvegarde	Sauvegarde des fichiers et des systèmes de fichiers.
compress	Compresse et décompresse des données.
cpio	Copie des fichiers vers et à partir de supports de stockage et de répertoires.
fdformat	Formate des disquettes.
flcopy	Copie de et sur disquettes.
format	Formate des disquettes.
fsck	Vérifie la cohérence du système de fichiers et remédie aux problèmes en mode interactif.
pack	Compresse des fichiers.
restore	Copie des fichiers ou des systèmes de fichiers, préalablement sauvegardés via la commande **backup**, à partir d'une unité locale.
tapechk	Contrôle le dérouleur de bande en continu.
tar	Manipule les archives.
tcopy	Copie une bande magnétique.
uncompress	Compresse et décompresse des données.
unpack	Décompresse des fichiers.

Chapitre 10. Sécurité du système et des fichiers

L'objectif de la sécurité informatique est d'assurer la protection des informations stockées sur le système informatique. La sécurité informatique est axée sur différents points :

Intégrité	La valeur d'une information dépend de sa fiabilité. Des données modifiées à mauvais escient perdent tout ou partie de leur intérêt.
Confidentialité	La valeur de beaucoup d'informations tient au fait qu'elles sont secrètes.
Disponibilité	Les informations doivent être disponibles.

Il est bon de planifier et de mettre en œuvre votre stratégie en matière de sécurité avant de commencer à exploiter votre système. Il est en effet très long de la modifier après coup : mieux vaut donc prévenir que guérir.

Ce chapitre traite des points suivants :

Menaces sur la sécurité

Les types de comportement suivants sont susceptibles de porter atteinte à la sécurité d'un système:

Négligence	La sécurité des informations est souvent mise à mal par simple négligence des utilisateurs autorisés. Par exemple, si vous divulguez votre mot de passe, aucun dispositif de sécurité, aussi sophistiqué soit–il, ne pourra empêcher un accès non autorisé à votre compte et à vos données.
Indiscrétion	Nombre de problèmes de sécurité sont le fait d'utilisateurs autorisés mais indiscrets, explorant le système à la recherche de données non sécurisées.
Pénétration	Il s'agit là d'une "attaque" délibérée contre le système. Il s'agit souvent d'une pénétration effectuée par quelqu'un ayant étudié de près les protections en place et ayant réussi à en déceler les failles.

Bien que la pénétration dans un système représente généralement la plus grande menace à la sécurité des informations, ne sous–estimez pas les problèmes engendrés par la négligence ou l'indiscrétion.

Sécurité de base

Voici quelques règles de base, applicables à tous les systèmes, pour assurer une sécurité minimale :

Sauvegardes

Se doter de sauvegardes de système physiquement sûres, fiables et mises à jours constitue la stratégie de sécurité la plus importante. Avec une sauvegarde de système fiable, vous pouvez faire face à tous types de problèmes sur le systèmes et subir des pertes de données réduites. Votre stratégie de sauvegarde doit être documentée, avec notamment des informations concernant :

- la fréquence des sauvegardes,
- le type de sauvegarde (système, données, par incréments),
- le mode de contrôle des bandes de sauvegarde,
- le mode de stockage des bandes de sauvegarde.

Pour en savoir plus, reportez–vous à "Fichiers de sauvegarde et supports de stockage, page 9-1".

Identification et authentification

Identification et authentification établissent votre identité. Vous êtes invité à vous connecter à votre système. Pour cela, vous devez fournir votre nom utilisateur et, éventuellement, un mot de passe (sur un système sûr, tous les comptes non affectés d'un mot de passe doivent être invalidés). Si le mot de passe est correct, vous accédez au compte correspondant et disposez des droits et des privilèges afférents.

Le mot de passe étant l'unique protection de votre compte, choisissez–le bien et protégez–le des indiscrétions. Tenter de deviner un mot de passe est souvent la base d'une tentative d'accès illégal. Pour une protection efficace des mots de passe, il n'y aucune confusion entre les données de sécurité et les données utilisateur. Les mots de passe, codés, et les autres données de sécurité se trouvent dans le fichier **/etc/security/passwd**. Ce fichier ne doit être accessible que par l'utilisateur root : en restreignant ainsi cet accès, un utilisateur malveillant ne peut décrypter les mots de passe via un programme qui boucle sur un ensemble de mots de passe possibles.

Il reste possible de deviner un mot de passe en tentant la connexion à un compte : si le mot de passe est banal ou qu'il n'est pas modifié assez souvent, l'opération risque d'aboutir.

ID de connexion

Le système d'exploitation identifie les utilisateurs grâce à leur *ID de connexion*. Cet ID permet au système de suivre toutes les actions d'un utilisateur. Après la connexion d'un utilisateur, mais avant l'exécution de son programme initial, le système affecte l'ID de connexion du processus à l'ID utilisateur trouvé dans la base de données. Tous les process suivants de la session sont affectés de cet ID. Il est ainsi possible de garder trace de toutes les activités exécutées sous cet ID.

Au cours de la session, vous pouvez réinitialiser l'*ID utilisateur effectif*, l'*ID utilisateur réel*, l'*ID groupe effectif*, l'*ID groupe réel* et l'*ID groupe supplémentaire*, mais vous ne pouvez pas changer l'ID de connexion utilisateur.

Terminaux sans surveillance

Tous les systèmes sont vulnérables si des terminaux sont laissés connectés sans surveillance. Le cas le plus grave étant celui d'un administrateur laissant sans surveillance son terminal connecté sous l'identité de l'utilisateur root. En règle générale, les utilisateurs doivent se déconnecter avant d'abandonner leur terminal.

Vous pouvez forcer un terminal à se déconnecter après une période d'inactivité en réglant les paramètres **TMOUT** et **TIMEOUT** dans le fichier de profile **profile /etc/**.
Le paramètre **TMOUT** fonctionne dans le shell (Korn) **ksh** et le paramètre **TIMEOUT** fonctionne dans le shell (Bourne) **bsh**. Pour de plus amples informations sur la commande **TMOUT**, reportez–vous à Substitution de paramètres (shell Korn ou POSIX), page 12-22. Pour de plus amples informations sur la commande **TIMEOUT**, reportez–vous à Substitution de variables (shell Bourne), page 12-83.

L'exemple suivant, extrait d'un fichier **.profile** force le terminal à se désactiver après une heure d'inactivité :

```
TO=3600
echo "Setting Autologout to $TO"
TIMEOUT=$TO
TMOUT=$TO
export TIMEOUT TMOUT
```

Remarque : Vous pouvez annuler les valeurs **TMOUT** et **TIMEOUT** dans le fichier **/etc/profile** en spécifiant d'autres valeurs dans le fichier **.profile** de votre répertoire personnel.

Propriété des fichiers et groupes d'utilisateurs

A l'origine, le propriétaire d'un fichier est identifié par l'ID de l'utilisateur ayant créé le fichier. C'est lui qui détermine qui peut lire, écrire (modifier) ou exécuter le fichier. La propriété peut être modifiée avec la commande **chown**.

Chaque ID utilisateur est rattaché à un ID groupe unique. Les groupes sont créés par l'administrateur au moment de la configuration du système. Lorsqu'un fichier est créé, le système attribue des droits à l'ID de l'utilisateur créateur, à l'ID du groupe auquel il appartient et à un groupe dit `autres`, consisté de tous les autres utilisateurs. La commande **id** affiche votre ID utilisateur (UID), votre ID groupe (IP), et les noms de tous les groupes auxquels vous appartenez.

Dans les listes de fichiers (comme les listes affichées par la commande **ls**), les groupes d'utilisateurs sont toujours représentés dans l'ordre suivant : utilisateur, groupe et autres. Si vous avez besoin de trouver votre nom de groupe, la commande **groupes** affiche tous les groupe d'un ID utilisateur.

Modification de la propriété d'un fichier ou d'un répertoire (commande chown)

Pour changer le propriétaire de vos fichiers, lancez la commande **chown**.

Lorsque l'option **–R** est spécifiée, la commande **chown** redescend de manière récursive dans la structure du répertoire à partir du répertoire spécifié. Chaque fois qu'elle rencontre un lien symbolique, le propriétaire du fichier ou du répertoire sur lequel pointe le lien est changé (la propriété du lien lui-même reste inchangée).

> **Remarque :** Seul l'utilisateur root est habilité à changer le propriétaire d'un fichier dont il n'est pas propriétaire. Les erreurs ne sont pas affichées lorsque l'option **–f** est spécifiée.

Par exemple, pour changer le propriétaire du fichier **program.c**, entrez :

```
chown jim program.c
```

Appuyez sur Entrée.

Les droits d'accès utilisateur au fichier **program.c** s'appliquent à présent à **jim**. Comme le propriétaire, **jim** peut utiliser la commande **chmod** pour autoriser ou refuser l'accès à d'autres utilisateurs au fichier **program.c**.

Reportez–vous à la commande **chown** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Modes d'accès aux fichiers et aux répertoires

Chaque fichier a un propriétaire. Il s'agit du créateur du fichier, pour les nouveaux fichiers. C'est lui qui définit le mode d'accès au fichier. Les modes d'accès déterminent les droits accordés aux autres utilisateurs sur ce fichier. Seuls le propriétaire d'un fichier ou l'utilisateur root peuvent modifier les droits d'accès à ce fichier.

Il existe trois classes d'utilisateurs : utilisateur/propriétaire, groupe et tous les autres. Les droits accordés à ces classes d'utilisateurs sont une combinaison des trois modes: lecture, écriture et exécution. Lorsqu'un nouveau fichier est créé, les droits d'accès par défaut les droits d'accès en lecture, en écriture, et en exécution pour l'utilisateur qui a créé le fichier. Les deux autres groupes disposent des droits d'accès en lecture et en exécution. Le tableau ci–dessous illustre les droits accordés par défaut à chaque classe de groupes d'utilisateurs :

Classes	Lecture	Ecriture	Exécution
Propriétaire	Oui	Oui	Oui
Groupe	Oui	Non	Oui
Autres	Oui	Non	Oui

Le système détermine qui détient des droits et le niveau d'autorisation accordé pour chaque activité. Les modes d'accès sont représentés de manière symbolique et de manière numérique dans le système d'exploitation.

Représentation symbolique des droits d'accès

La représentation symbolique des droits d'accès est la suivante :

r Droit de lecture : les utilisateurs sont autorisés à consulter le fichier.

w Droit d'écriture : les utilisateurs sont autorisés à modifier le contenu du fichier.

x Droit d'exécution : les utilisateurs sont autorisés à exécuter le fichier (il doit s'agir bien entendu d'un fichier exécutable, c'est-à-dire, le plus souvent d'un fichier contenant un programme). Pour les répertoires, ce droit donne la possibilité d'explorer leur contenu.

Les modes d'accès des fichiers et des répertoires sont représentés par neuf caractères. Les trois premiers caractères représentent les droits d'accès en cours de **Propriétaire**, les trois caractères suivants représente les droits d'accès en cours de **Groupe** et les trois derniers caractères représentent les paramètres par défaut des droits d'accès des **Autres**. Un tiret (**–**) dans l'ensemble de neuf caractères indique qu'aucun droit d'accès n'est octroyé. Par exemple, un fichier avec un mode d'accès, `rwxr-xr-x` autorise la lecture et l'exécution aux trois groupes, mais les droits d'accès en écriture sont seulement octroyés au propriétaire du fichier. Il s'agit de la représentation symbolique des droits par défaut.

La commande **ls** utilisée avec l'indicateur **–l** (L minuscule), fournit une liste détaillée du répertoire courant. Les 10 premiers caractères de la liste **ls –l** affiche le type du fichier et les droits d'accès pour chacun des trois groupes. La commande **ls –l** indique également le propriétaire et le groupe associés à chaque fichier et à chaque répertoire.

Le premier caractère indique le type de fichier. Les neuf autres caractères comportent des informations de droits d'accès au fichier pour chacune des trois classes d'utilisateurs. Voici les types de fichiers possibles :

– fichiers standard

d répertoire

b fichiers blocs spéciaux

c fichiers caractères spéciaux

p fichiers tubes spéciaux

l liens symboliques

s sockets

Dans cet exemple, il s'agit d'un extrait de liste **ls –l** :

```
-rwxrwxr-x   2   janet   acct   512 Mar 01 13:33 january
```

Le premier caractère le tiret (–) indique un fichier standard. Les neuf caractères suivants `(rwxrwxr-x)` représentent les modes d'accès de Utilisateur, Groupe et Autres, comme décrit auparavant. `janet` est le propriétaire du fichier et `acct` est le nom du groupe de Janet. `512` est la taille du fichier, en octets, `Mar 01 13:33` est la dernière date de modification, et `january` est le nom du fichier. Le `2` indique le nombre de liens associés au fichier.

Représentation numérique des droits d'accès

Numériquement, les droits d'accès sont représentés par 4 pour la lecture, 2 pour l'écriture et 1 pour l'exécution. La somme des droits (comprise entre 1 et 7) représente les droits d'accès de chaque catégorie (utilisateur, groupe et autres). Le tableau suivant présente les valeurs numériques de chaque niveau d'accès :

Total	Lecture	Ecriture	Exécution
0	–	–	–
1	–	–	1
2	–	2	–
3	–	2	1
4	4	–	–
5	4	–	1
6	4	2	–
7	4	2	1

A la création d'un fichier, les droits par défaut sont représentés par 755, soit : accès en lecture, écriture et exécution (4+2+1=7) pour l'utilisateur, accès en lecture et exécution (4+1=5) pour le groupe, et lecture et exécution (4+1=5) pour les autres. Pour modifier les modes de droits d'accès des fichiers que vous possédez, exécutez la commande **chmod** (changer de mode).

Affichage d'informations sur le groupe (commande lsgroup)

Utilisez la commande **lsgroup** pour afficher les attributs de tous les groupes du système (ou de groupes spécifiques). Si un ou plusieurs attributs ne peuvent pas être lus, la commande **lsgroup** répertorie autant d'informations que possible. Les informations d'attribut s'affichent sous la forme de définitions *Attribut* **=** *Valeur*, chacune étant séparée par un espace.

Liste de tous les groupes du système

Pour répertorier tous les groupes sur le système, entrez :

```
lsgroup ALL
```

Le système affiche les groupes, leur ID et la liste des utilisateurs du groupe, comme suit :

```
system  0        arne,pubs,ctw,geo,root,chucka,noer,su,dea,
backup,build,janice,denise
staff   1        john,ryan,flynn,daveb,jzitt,glover,maple,ken
gordon,mbrady
bin     2        root,bin
sys     3        root,su,bin,sys
```

Liste d'attributs spécifiques de tous les groupes

Pour afficher les attributs spécifiques de tous les groupes, effectuez l'une des étapes suivantes :

- Série d'expressions `Attribut=Valeur`, séparées par un espace (style par défaut). C'est le style par défaut. Par exemple, pour afficher la liste des ID et des utilisateurs de tous les groupes du système, entrez :

```
lsgroup -a id users ALL | pg
```

  Appuyez sur Entrée. Cette commande affiche les attributs.

  Une liste semblable à la suivante s'affiche :

```
system id=0 users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build
staff id=1 users=john,ryan,flynn,daveb,jzitt,glover,maple,ken
```

- Format strophes. Par exemple, pour afficher la liste des ID et des utilisateurs de tous les groupes du système au format strophe, entrez :

```
lsgroup -a -f id users ALL | pg
```

  Une liste semblable à la suivante s'affiche :

```
system:
   id=0
   users=pubs,ctw,geo,root,chucka,noer,su,dea,backup,build

staff:
   id=1
   users=john,ryan,flynn,daveb,jzitt,glover,maple,ken

bin:
   id=2
   users=root,bin

sys:
   id=3
   users=root,su,bin,sys
```

Affichage des attributs d'un groupe spécifique

Pour afficher tous les attributs d'un groupe spécifique, vous avez également le choix entre deux styles :

- Série d'expressions `Attribut=Valeur`, séparées par un espace. C'est le style par défaut. Par exemple, pour afficher la liste des attributs du groupe système, entrez :

```
lsgroup system
```

  Une liste semblable à la suivante s'affiche :

```
system id=0
users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,backup,build,janice,denise
```

- Format strophes. Par exemple, pour afficher la liste de tous les attributs du groupe `bin` au format strophes, entrez :

```
lsgroup -f system
```

  Une liste semblable à la suivante s'affiche :

```
system:
    id=0   users=arne,pubs,ctw,geo,root,chucka,noer,su,dea,
 backup,build,janice,denise
```

Liste d'attributs spécifiques d'un groupe

Pour afficher des attributs spécifiques d'un groupe donné, entrez :

```
lsgroup –a Attributs Groupe
```

Par exemple, pour afficher la liste des ID et des utilisateurs du groupe `bin`, entrez :

```
lsgroup –a id users bin
```

Appuyez sur Entrée.

Une liste semblable à la suivante s'affiche :

```
bin id=2 users=root,bin
```

Reportez–vous à la commande **lsgroup** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Modification des droits d'accès aux fichiers et aux répertoires (commande chmod)

Pour modifier les droits d'accès en lecture, en écriture et en exécution des fichiers spécifiés, et les modes des répertoires spécifiés, utilisez la commande **chmod**.

Voir les exemple ci–après :

- Pour ajouter un type de permission aux fichiers **chap1** et **chap2**, entrez :

```
chmod g+w chap1 chap2
```

  Le droit d'accès en écriture aux fichiers chap1 et chap2 est octroyé aux membres du groupe.

- Pour apporter simultanément plusieurs changements de permission au répertoire `monrep`, entrez :

```
chmod go–w+x monrep
```

  Cette commande refuse (**–**) aux membres du groupe (**g**) et aux autres utilisateurs (**o**) les droits de créer ou de supprimer les fichiers (**w**) dans le répertoire **monrep** et permet (**+**) aux membres du groupe et aux autres utilisateurs d'effectuer une recherche dans le répertoire **monrep** ou de l'utiliser (**x**) dans un nom de chemin d'accès. Cette commande équivaut à la séquence:

```
chmod g–w monrep
chmodo–wmonrep
chmodg+xmonrep
chmodo+xmonrep
```

- Pour réserver à son propriétaire l'usage de la procédure shell **cmd** comme commande, entrez :

```
chmod u=rwx,go= cmd
```

  Les droits d'accès en lecture, en écriture et en exécution sont alors octroyés à l'utilisateur propriétaire du fichier (**u=rwx**). Le groupe et les autres utilisateurs se voient également refuser les droits d'accès à `cmd` (**go=**).

- Pour utiliser le mode numérique de la commande **chmod** pour changer les droits afférents au fichier `text`, entrez :

```
chmod 644 text
```

  Ce qui accorde au propriétaire le droit d'accès en lecture et en écriture, les autres se contentant de la lecture seule.

Reportez–vous à la commande **chmod** dans le manuel *AIX 5L Version 5.3 Commands Reference* pour avoir la syntaxe complète.

Listes de contrôle d'accès

Le contrôle d'accès est constitué de ressources d'informations protégées indiquant qui peut avoir accès à ces ressources. Le système d'exploitation permet un accès limité aux initiés ainsi qu'une sécurité discrétionnaire. Le propriétaire d'une ressource d'informations peut octroyer des droits d'accès en lecture ou en écriture à d'autres utilisateur pour cette ressource. Un utilisateur possédant des droits d'accès à une ressource peut transférer ses droits à d'autres utilisateurs. Cette sécurité permet un flux d'informations contrôlé par l'utilisateur dans le système. Le propriétaire d'une ressource d'informations définit les droits d'accès à l'objet.

Les utilisateurs possèdent uniquement un accès basé sur le nombre d'utilisateurs vers l'objet qu'ils possèdent. En général, les utilisateurs reçoivent soit des droits de groupe soit des droits par défaut pour une ressource. La tâche principale pour l'administration du contrôle d'accès consiste à définir l'appartenance à un groupe des utilisateurs, en effet, cette appartenance détermine les droits d'accès des utilisateurs aux fichiers qu'ils ne possèdent pas.

Liste ACL pour objets de systèmes de fichiers

Les objets de systèmes de fichiers sont généralement associés à une Liste de contrôle des accès (Access Control List) qui se compose normalement d'une série d'entrées de contrôle des accès (Access Control Entries). Chaque entrée ACE définit l'identité et les droits d'accès afférents.

Pour conserver les listes de contrôle des accès, utilisez les commandes **aclget**, **acledit**, **aclput** et **aclconvert**.

Notez que la liste ACL est généralement stockée et gérée sur le support pas le système de fichier physique (PFS). AIX 5.3 fournit une infrastructure pour les systèmes de fichiers physiques afin de prendre en charge et gérer différents types de listes ACL. Le système de fichier JFS2 fourni avec AIX prend en charge deux types de listes ACL :

- AIXC
- NFS4

Les systèmes de fichier précédents ne prennent en charge que le type de liste ACL AIXC comme dans les versions précédentes du logiciel AIX. Ces types de listes ACL sont décrits en détail dans le manuel *AIX 5L Version 5.3 Security Guide*. Les sections suivantes comportent de brèves explications de ces types de listes ACL.

Type de liste ACL AIXC

Le type de liste ACL AIXC (AIX Classic) tient compte du comportement de la liste ACL, comme défini dans les versions précédentes de AIX. Ce type de liste ACL comporte les bits en mode de base réguliers et les droits étendus (ACE). Par le biais de ces droits étendus, vous pouvez autoriser ou interdire l'accès à un fichier sans modifier les droits de base.

Remarque : La liste ACL AIXC d'un fichier ne peut pas excéder une page mémoire (environ 4096 octets).

La commande **chmod** en mode numérique (avec notations octales) peut définir des droits d'accès et des attributs de base. La sous–routine **chmod**, appelée par la commande, désactive les droits d'accès étendus. Si vous utilisez le mode numérique de la commande **chmod** sur un fichier doté d'une liste ACL, les droits d'accès étendus sont désactivés. Le mode symbolique de la commande **chmod** ne désactive pas les droits d'accès étendus lorsque la liste ACL associée est du type AIXC. Pour en savoir plus sur ces modes, reportez–vous à la commande **chmod**.

Droits d'accès de base

Les droits d'accès de base de la liste ACL AIXC sont les modes d'accès fichier attribués normalement au propriétaire du fichier, au groupe associé et aux autres utilisateurs. Les modes d'accès sont en lecture (r), en écriture (w) et en exécution/recherche (x).

Remarque : Les droits d'accès de base du type de liste ACL AIXC sont identiques aux bits en mode fichier stockés dans les en–têtes i–node de l'objet du système de fichiers. Ceci signifie que les informations en bits en mode de base sont identiques à la valeur retournée par le système fichier lorsque la commande **stat** est exécutée sur l'objet de système fichier.

Dans une liste de contrôle d'accès, les droits d'accès de base sont au format suivant, avec le paramètre *Mode* exprimé en rwx (un tiret (–) remplaçant chaque droit non spécifié) :

```
droits d'accès de base :
    propriétaire(nom) : Mode
    groupe (groupe) :  Mode
    autres : Mode
```

Attributs

Trois attributs peuvent être ajoutés à une liste de contrôle d'accès :

setuid (SUID) Bit ID utilisateur Cet attribut définit les ID utilisateurs effectifs et sauvegardés
du process sur l'ID propriétaire du fichier en cours d'exécution.

setgid (SGID) Bit ID groupe Cet attribut définit les ID groupes effectifs et sauvegardés du process sur l'ID groupe du fichier en cours d'exécution.

savetext (SVTX) Sauvegarde le texte sous format fichier texte.

Ces attributs sont ajoutés au format suivant :

```
attributs : SUID, SGID, SVTX
```

Droits d'accès étendus

Les droits étendus de la liste ACL AIXC sont un moyen pour le propriétaire d'un fichier de définir plus précisément les droits d'accès à ce fichier. Ils permettent de modifier les droits de base (utilisateur, groupe et autres) en accordant, en supprimant ou en spécifiant des droits spécifiques pour des individus, des groupes ou des combinaisons de groupes. Les droits d'accès sont modifiés à l'aide de mots clés.

Les mots clés autoriser, refuser et spécifier sont définis comme suit :

autoriser	Accorde à l'utilisateur ou au groupe l'accès spécifié au fichier
refuser	Retire à l'utilisateur ou au groupe le droit d'accès spécifié au fichier
spécifier	Définir précisément l'accès au fichier par l'utilisateur ou le groupe.

Si un utilisateur se voir refuser un accès spécifique par une touche **accès refusé** ou une touche **spécification**, aucune autre entrée ne peut remplacer ce refus d'accès.

La touche **activé** doit être spécifiée dans la liste de contrôle d'accès pour que les droits étendus prennent effet. La valeur par défaut est la touche **désactivé**.

Dans une liste ACL AIXC, les droits d'accès étendus apparaissent sous la forme :

```
droits d'accès étendus :
   activé | désactivé
     accès accordé   Mode  InfoUtil...:
     accès refusé    Mode  InfoUtil...:
     spécification   Mode  InfoUtil...:
```

Utilisez une ligne de séparation pour chaque entrée **accès accordé**, **accès refusé** ou **spécification**. Le paramètre **Mode** est exprimé en **rwx** (un tiret (–) remplaçant chaque droit non spécifié). Le paramètre **InfoUtil** est exprimé de la façon suivante : **u : NomUtilisateur** ou **g : NomGroupe** ou une combinaison, séparée par une virgule, de **u : NomUtilisateur** et **g : NomGroupe**.

Remarque : Si vous spécifiez plusieurs noms d'utilisateur dans une entrée, celle–ci ne peut pas être utilisée dans une décision de contrôle d'accès car un process n'a qu'un seul ID utilisateur.

Type de liste ACL NFS4

Le système de fichiers JFS2 dans AIX prend aussi en charge le type de liste ACL NFS 4. Cette implémentation de liste ACL suit la définition de liste ACL comme indiqué dans le RFC associé au protocole de la version 4 du NFS. Cette liste ACL fournit un contrôle granulaire beaucoup plus affiné sur les droits d'accès et fournit des fonctions spécifiques comme l'héritage. La liste ACL NFS4 est une série d'entrées ACE. Chaque entrée ACE définit les droits d'accès pour une identité. Comme défini dans le RFC, les principaux éléments de l'entrée ACE NFS4 sont :

```
struct nfsace4 {

                    acetype4          type;
                    aceflag4           flag;
                    acemask4        access_mask;
                    utf8str_mixed   who;
 };
```

Où :

type Masque de bit définissant le type d'entrée ACE. Certaines informations, comme par exemple si cette entrée ACE permet l'accès ou le refuse, sont définies ici.

flag Masque de bit qui décrit les aspects d'héritage de l'entrée ACE. Définit si cette entrée ACE s'applique à l'objet de systèmes de fichiers, à ses enfants ou aux deux.

access_mask Masque de bit définissant différents droits d'accès possibles. Parmi ces droits figurent : lecture, écriture, exécuter, créer, supprimer, créer enfant, supprimer enfant, etc.

who Cette chaîne se terminant par zéro définit l'identité de la personne à qui cette entrée ACe s'applique. Notez que par RFC, la taille de cette chaîne est illimitée et une définition peu précise permet de définir des domaines à l'intérieur des réseaux de NFS version 4 pour gérer le contrôle des accès. A l'origine (la plupart du temps) AIX n'interprète pas cette chaîne et chaque entrée ACE est associée avec une identité comprise par AIX (comme **uid** ou **gid**). Le système de fichier NFS (Network File System) version 4 est censé interpréter ces chaînes lorsqu'il s'avère nécessaire de les convertir en ID utilisateur ou groupe prises en charge par le système d'exploitation. AIX comprend seulement certaines des chaînes spéciales **who** définies dans le RFC.

Dans AIX, utilisez les commandes **aclget**, **acledit**, **aclput** et **aclconvert** pour gérer les listes ACL NFS4.

Exemple de liste ACL pour AIXC

Voici un exemple de liste ACL pour AIXC :

```
attributs : SUID
droit d'accès de base
       propriétaire(frank):  rw-
       groupe(system):       r-x
       autres: ---
droits d'accès étendus :
       activé
         accès accordé  rw-  u:dhs
         accès refusé   r--  u:chas, g:system
         spécification  r--  u:john, g:gateway, g:mail
         accès accordé  rw-  g:account, g:finance
```

Voici la signification des différents éléments de cette liste :

- La première ligne indique que l'octet setuid est activé.
- La deuxième ligne, qui introduit les droits de base, est facultative.
- Les trois lignes suivantes précisent ces droits. Les noms du propriétaire et du groupe (entre parenthèses) sont donnés à titre d'information : les modifier n'a pas d'incidence sur le propriétaire réel du fichier, pas plus que sur le groupe auquel appartient le fichier. Seule la commande **chown** et la commande **chgrp** peuvent modifier ces attributs de fichier.
- La ligne suivante, qui introduit les droits étendus, est facultative.
- La ligne suivante indique que les droits étendus qui suivent sont activés.
- Les quatre dernières lignes correspondent aux entrées étendues :
- La première entrée étendue accorde à l'utilisateur dhs l'accès en lecture (r) et en écriture (w) au fichier.
- La deuxième entrée étendue refuse les droits d'accès en lecture (r) à l'utilisateur `chas` uniquement s'il est membre du groupe `system`.
- La troisième entrée étendue indique qu'aussi longtemps que l'utilisateur `john` est membre des groupes `gateway` et `mail` il détiendra les droits d'accès en lecture (r). Si l'utilisateur `john` n'appartient plus à ces deux groupes, l'accès lui est refusé.
- La dernière entrée étendu octroie à tout utilisateur appartenant aux *deux* groupes, `account` et `finance`, les droits d'accès en lecture (r) et en écriture (w).

Remarque : Plusieurs entrées étendues peuvent être appliquées à un process, les interdits primant sur les autorisations.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **acledit** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Autorisations d'accès

Le propriétaire de la ressource d'informations est responsable de la gestion des droits d'accès. Les ressources sont protégées par des bits d'accès, intégrés au mode de l'objet. Pour la liste ACL AIXC, ces bits définissent les droits du propriétaire de l'objet, ceux du groupe correspondant et ceux de la classe par défaut `autres`. Le type de liste ACL AIXC gère trois droits d'accès (lecture, écriture et exécution), qui peuvent être accordés séparément.

Lorsqu'un utilisateur se connecte à un compte (via une commande **login** ou **su**), les ID utilisateur et groupe attribués au compte sont associés aux processus de cet utilisateur et en déterminent les droits d'accès. Ces ID déterminent les droits d'accès du processus.

Pour les fichiers, les répertoires, les tubes nommés, et les unités (fichiers spéciaux) associés à une liste ACL AIX, les accès sont autorisés comme suit :

- Pour chaque entrée (ACE) de la liste de contrôle des accès (ACL), la liste des identificateurs est comparée aux identificateurs du process. Si elles sont identiques, le process se voit attribuer les droits et les interdits définis pour cette entrée. Les correspondances logiques pour les droits comme pour les interdits sont calculées pour chaque entrée concernée de l'ACL. Si le process demandeur ne correspond à aucune entrée de l'ACL, il se voit attribuer les droits associés à l'entrée par défaut.
- Si le droit d'accès demandé est autorisé (compris dans la somme des droits) et non interdit (compris dans la somme des interdits), le droit demandé est accordé. Sinon, il est refusé.

De plus, pour un type de liste ACL AIXC, la liste des identificateurs d'une liste ACL correspond à un process si tous ses identificateurs correspondent à l'identificateur effectif – de même type – du process demandeur. Un identificateur de type USER correspond à un process s'il est identique à l'ID utilisateur effectif du process. Un identificateur de type GROUP correspond s'il est identique à l'ID groupe effectif du process ou à l'un des ID des groupes complémentaires. Ainsi, une liste ACE avec la liste d'identificateurs suivante :

```
USER:fred, GROUP:philosophers, GROUP:software_programmer
```

correspondrait à un process avec un ID utilisateur effectif, `fred` et à un ensemble de groupes,

```
philosophers, philanthropists, software_programmer, doc_design
```

mais ne correspondrait pas à un process avec un ID utilisateur effectif, `fred` et à un ensemble de groupes,

```
philosophers, iconoclasts, hardware_developer, graphic_design
```

Notez qu'une ACE avec la liste suivante correspond aux deux process :

```
USER:fred, GROUP:philosophers
```

En d'autres termes, la liste d'identificateurs de l'ACE fonctionne comme un ensemble de conditions, à respecter pour que l'accès spécifié soit accordé.

Le mécanisme de contrôle d'accès discrétionnaire assure un contrôle effectif de l'accès aux ressources et assure une protection distincte de la confidentialité et de l'intégrité des informations. Les mécanismes de contrôle gérés par le propriétaire ne sont effectifs que s'ils sont définis par les utilisateurs. Tous les utilisateurs doivent maîtriser le mécanisme d'octroi et de refus de droits d'accès.

Notez qu'avec les objets de systèmes de fichiers avec un type de liste ACL NFS4, les contrôles d'accès sont basés sur différentes entrées ACE qui composent la liste ACL conformément aux règles définies dans le RFC associé au protocole de la version 4 du NFS. La correspondance d'identité est effectuée sur la base de l'ID utilisateur ou de l'ID groupe ou des chaînes spéciales `who` définies dans la liste ACE par rapport aux droits du process. En cas de correspondance, les droits d'accès demandés sont comparés aux droits d'accès définis dans la liste ACE. Tout droit d'accès autorisé est mis de côté et le processus de comparaison se poursuit sur l'entrée ACE suivante. Ce process se poursuit jusqu'à ce que la liste ACL soit intégralement parcourue, jusqu'à ce que les conditions relatives aux droits d'accès soient remplies ou bien jusqu'à ce que les droits d'accès demandés soient refusés. Les étapes suivantes montrent le processus de vérification d'accès en présence d'un objet de système de fichier avec une liste ACL NFS4 :

1. Pour chaque entrée (ACE) de la liste de contrôle des accès (ACL), la liste des identificateurs est comparée aux identificateurs du process. Les contrôles d'identité comprennent l'ID utilisateur ou l'ID groupe définie dans la liste ACE. De plus, si l'identité est définie comme étant **spéciale** avec des chaînes comme OWNER@, une correspondance sera obtenue si le process d'appel est engagé par le propriétaire du fichier. En cas de correspondance, le process se voit attribuer les droits d'accès définis pour cette entrée. Sinon, le process se poursuit avec l'entrée ACE suivante.
2. Les droits d'accès demandés sont comparés avec les droits d'accès récupérés à partir de l'entrée ACE. Si tout droit d'accès est explicitement refusé par l'entrée ACE, le

process de vérification de l'accès est stoppé et l'accès sera refusé au process demandeur.

3. Si certains des droits d'accès demandés sont conformes avec l'entrée ACE, ces droits d'accès seront séparés de la liste des droits d'accès demandés. Le process de comparaison se poursuit alors avec l'entrée ACE suivante.
4. Si tous les droits d'accès demandés sont conformes avec l'entrée ACE, l'accès demandé est alors autorisé.
5. Si l'ensemble de la liste ACL est traité avant que tous les droits d'accès demandés n'aient été résolus, l'accès est refusé.

Notez que, mis à part les contrôles d'accès basés sur le type de liste ACL, les systèmes de fichiers physiques individuels peuvent aussi fournir un accès basé sur les privilèges aux objets de systèmes de fichiers. Par exemple, un propriétaire pourra toujours avoir au moins les droits pour modifier la liste ACL, indépendamment des droits d'accès ACL existants. Un process doté de l'ID utilisateur 0 est dit process utilisateur racine. Ces process ont généralement tous les droits d'accès. Toutefois, si un processus root demande le droit d'exécution sur un programme, celui–ci ne lui est accordé que s'il est détenu par au moins un utilisateur.

Tous les contrôles d'accès pour ces objets sont effectués au niveau de l'appel système, au moment du premier accès aux objets. Dans la mesure où l'accès aux objets SVIPC n'est pas nominatif, les contrôles sont effectués à chaque accès. Cependant, des contrôles peuvent être effectués par les systèmes de fichiers physiques lorsque l'objet de système de fichiers est ouvert et non lors du process de lecture ou d'écriture. Pour les objets avec noms de systèmes de fichiers, il est nécessaire de pouvoir résoudre le nom de l'objet réel. Les noms sont résolus de façon relative (au répertoire de travail du process) ou absolue (par rapport au répertoire racine du process). Toute résolution de nom commence par la recherche de l'un de ces deux éléments.

Affichage des informations de contrôle des accès (commande aclget)

La commande **aclget** permet d'afficher les informations de contrôle des accès à un fichier. Les informations affichées dépendront du type de liste ACL.

Par exemple, pour afficher les informations de contrôle des accès au fichier d'état, entrez :

```
aclget status
```

Une liste indiquant les attributs, les droits d'accès de base et les droits d'accès étendus s'affiche. Un exemple est donné à la section Exemple de liste ACL, page10-15.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **aclget** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Paramétrage des informations de contrôle des accès (commande aclput)

La commande **aclget** permet de paramétrer les informations de contrôle des accès à un fichier.

Par exemple, pour paramétrer les informations de contrôle des accès au fichier d'état, avec des informations de contrôle des accès stockées dans le fichier **acldefs**, entrez :

```
aclput –i acldefs status
```

Pour paramétrer les informations de contrôle des accès au fichier d'état avec les mêmes informations que celles utilisées pour le fichier **plans**, entrez :

```
aclget plans | aclput status
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **aclput** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Modification des informations de contrôle des accès (commande acledit)

Utilisez la commande **acledit** pour modifier les informations de contrôle des accès à un fichier. Elle affiche les informations courantes et permet au propriétaire du fichier de les modifier. Une confirmation est demandée avant toute modification permanente.

Remarque : La variable d'environnement EDITOR doit être spécifiée avec un nom de chemin complet ; dans le cas contraire, la commande **acledit** échoue.

Les information de contrôle des accès qui s'affichent dépendent du type de liste ACL. Un exemple est donné à la section Exemple de liste ACL, page10-15.

Pour modifier les informations de contrôle des accès du fichier **plans**, entrez :

```
acledit plans
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **acledit** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Exemple de liste ACL

Voici un exemple de liste ACL :

```
attributes : SUID
base permissions:
       owner(frank):  rw-
       group(system): r-x
       others: ---
extended permissions:
       enabled
         permit  rw-  u:dhs
         deny    r--  u:chas, g:system
         specify r--  u:john, g:gateway, g:mail
         permit  rw-  g:account, g:finance
```

Voici la signification des différents éléments de cette liste :

- La première ligne indique que le bit **setuid** est activé.
- La deuxième ligne, qui présente les droits d'accès de base, est facultative.
- Les trois lignes suivantes précisent ces droits. Les noms du propriétaire et du groupe (entre parenthèses) sont donnés à titre d'information. La modification de ces noms n'a pas d'incidence sur le propriétaire réel du fichier, pas plus que sur le groupe auquel appartient le fichier. Seules les commandes **chown** et **chgrp** permettent de modifier ces attributs.
- La ligne suivante, qui présente les droits d'accès étendus, est facultative.
- La ligne suivante indique que les droits d'accès étendus qui suivent sont activés.
- Les quatre dernières lignes correspondent aux entrées étendues : La première entrée étendue accorde à l'utilisateur `dhs` l'accès en lecture (r) et en écriture (w) au fichier.
- La deuxième interdit l'accès en lecture (r) à l'utilisateur `chas` lorsqu'il est membre du groupe `system`.
- La troisième accorde à l'utilisateur `john` l'accès en lecture (r) tant qu'il est membre des deux groupes `gateway` et `mail`. Si l'utilisateur `john` n'appartient pas à ces deux groupes, l'accès lui est refusé.
- La dernière ligne, enfin, accorde à tout utilisateur membre des deux groupes *account* et `finance` l'accès en lecture `(r) et en écriture` (w).

  Remarque : Plusieurs entrées étendues peuvent être appliquées à un process, les interdits primant sur les autorisations.

  Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **acledit** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Affichage des informations de contrôle des accès (commande aclget)

Utilisez la commande **aclget** pour afficher les informations de contrôle des accès à un fichier. Ces informations sont les attributs, les droits d'accès de base et les droits d'accès étendus.

Par exemple, pour afficher les informations de contrôle des accès au fichier **d'état**, entrez :

```
aclget status
```

Une liste indiquant les attributs, les droits d'accès de base et les droits d'accès étendus s'affiche. Un exemple est donné à la section Exemple de liste ACL, page10-15.

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **aclget** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Paramétrage des informations de contrôle des accès (commande aclput)

Utilisez la commande **aclput** pour paramétrer les informations de contrôle des accès à un fichier.

> **Remarque :** Une liste de contrôle des accès ne peut pas excéder une page mémoire (environ 4096 octets).

Voir les exemple ci–après :

- Pour paramétrer les informations de contrôle des accès pour le fichier **d'état** avec celles stockées dans le fichier `acldefs`, entrez :

```
aclput –i acldefs status
```

- Pour paramétrer les informations de contrôle des accès au fichier **d'état** avec les mêmes informations que celles utilisées pour le fichier **plans**, entrez :

```
aclget plans | aclput status
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **aclput** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Modification des informations de contrôle des accès (commande acledit)

Utilisez la commande **acledit** pour modifier les informations de contrôle des accès à un fichier. Elle affiche les informations courantes et permet au propriétaire du fichier de les modifier. Une confirmation est demandée avant toute modification permanente.

> **Remarque :**La variable d'environnement **EDITOR** doit être spécifiée avec son chemin d'accès complet, faute de quoi la commande **acledit** échoue.

Une liste indiquant les attributs, les droits d'accès de base et les droits d'accès étendus s'affiche. Un exemple est donné à la section Exemple de liste ACL, page 10-15.

Par exemple, pour modifier les informations de contrôle des accès au fichier `plans`, entrez :

```
acledit plans
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **acledit** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Verrouillage du terminal (commande lock ou xlock)

Utilisez la commande **lock** pour verrouiller votre terminal. La commande **lock** vous invite à entrer votre mot de passe, le lit, puis vous invite à entrer de nouveau votre mot de passe afin de le vérifier. Dans l'intervalle, la commande verrouille l'écran et ne le libère que lorsque vous indiquez le mot de passe la seconde fois. Le délai par défaut est de 15 minutes, mais vous pouvez le modifier via l'indicateur **–** *Nombre*.

> **Remarque :** Si vous travaillez sous AIXwindows, utilisez la commande **xlock** de la même manière.

Par exemple, pour verrouiller votre écran avec contrôle par le mot de passe, entrez :

```
lock
```

Vous êtes invité à indiquer votre mot de passe deux fois pour que le système le vérifie. Si ce mot de passe n'est pas répété dans les 15 minutes qui suivent, la commande expire.

Par exemple, pour attribuer un délai de `10`minutes à un terminal sous contrôle par mot de passe, entrez :

```
lock –10
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **lock** ou **xlock** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Récapitulatif des commandes relatives à la sécurité

acledit	Modifie les données de contrôle d'accès à un fichier
aclget	Affiche les données de contrôle d'accès à un fichier
aclput	Définit les données de contrôle d'accès à un fichier
chmod	Modifie les droits d'accès
chown	Change le propriétaire d'un fichier
lock	Réserve un terminal
lsgroup	Affiche les attributs de groupes
xlock	Verrouille l'écran X local (jusqu'à l'entrée d'un mot de passe)

Chapitre 11. Personnalisation de l'environnement utilisateur

Le système d'exploitation met à votre disposition plusieurs commandes et fichiers d'initialisation vous permettant de personnaliser à votre gré votre environnement utilisateur.

Il est également possible de personnaliser certaines ressources par défaut des applications utilisées sur votre système. Les valeurs par défaut sont lancées par le programme au démarrage. Si vous les modifiez, vous devez quitter le programme et le relancer pour que les nouvelles valeurs soient prises en compte.

Pour obtenir des informations sur la personnalisation de votre environnement Common Desktop, reportez–vous à *Common Desktop Environment 1.0 : Advanced User's and System Administrator's Guide*.

Cette section décrit les points suivants :

Fichiers de lancement du système : généralités

Lorsque vous vous connectez, le shell définit votre environnement utilisateur en fonction des fichiers d'initialisation que vous avez configurés. Les caractéristiques de votre environnement utilisateur sont définies à l'aide des valeurs attribuées à vos variables d'environnement. L'environnement reste actif jusqu'à ce que vous vous déconnectiez.

Au moment de la connexion au système d'exploitation, le shell se sert de deux types de fichiers de profil. Il évalue les commandes de ces fichiers, puis exécute les commandes de configuration de l'environnement. Ces fichiers possèdent des fonctions similaires, si ce n'est que le fichier **/etc/profile** contrôle les variables de profil de tous les utilisateurs d'un système, tandis que le fichier **.profile** permet de personnaliser votre propre environnement.

Le shell exécute d'abord les commandes pour définir votre environnement système dans le fichier **/etc/environment** puis évalue les commandes contenues dans le fichier **/etc/profile.** Le shell vérifie ensuite si vous disposez d'un fichier **.profile** dans votre répertoire personnel. Si un fichier **.profile** existe, il est lancé. Le fichier **.profile** indique également s'il existe un fichier d'environnement. Dans l'affirmative (le fichier s'appelle généralement **.env**), le système l'exécute et configure les variables d'environnement.

Les fichiers **/etc/profile**, **/etc/environment** et **.profile** sont exécutés une fois au moment de la connexion. Le fichier **.env**, lui, est exécuté chaque fois que vous ouvrez un nouveau shell ou une fenêtre.

Cette section traite des points suivants :

Fichier /etc/environment

Le fichier **/etc/environment** est le premier fichier utilisé par le système au moment de la connexion. Il contient des variables spécifiant l'environnement de base de tous les process. Lorsqu'un nouveau process est lancé, la sous–routine **exec** crée un tableau des chaînes disponibles, sous la forme *Nom=Valeur*. Ce tableau est appelé *environnement*. Chaque nom défini par une chaîne étant appelé *variable d'environnement* ou *variable shell*. La sous–routine **exec** permet de définir en une seule fois l'ensemble de l'environnement.

Lorsque vous vous connectez, le système définit les variables d'environnement à partir du fichier **/etc/environment** avant de lire votre profil de connexion, **.profile**. Voici les variables constitutives de l'environnement de base.

HOME	Chemin d'accès complet au répertoire de connexion ou **HOME** de l'utilisateur. Le programme **login** lui affecte le nom défini dans le fichier **/etc/passwd**.
LANG	Nom courant actuel. La variable **LANG** est initialement définie dans le fichier **/etc/profile** au moment de l'installation.
NLSPATH	Chemin d'accès complet aux catalogues de messages.
LOCPATH	Chemin d'accès complet à l'emplacement des tables NLS (National Language Support).
CHEMIN	Séquence des répertoires à explorer par des commandes, telles que **sh**, **time**, **nice** et **nohup,** à la recherche d'une commande dont le chemin d'accès est incomplet.
TZ	Zone temps. La variable d'environnement **TZ** est initialement définie dans le fichier **/etc/profile**, le fichier de connexion système.

Reportez–vous au manuel *AIX 5L Version 5.3 Files Reference* pour obtenir des informations détaillées sur le fichier **/etc/environment**.

fichier /etc/profile

Le fichier **/etc/profile** est le second fichier utilisé par le système au moment de la connexion. Ce fichier contrôle des variables par défaut applicables à l'intégralité du système, telles que :

- Variables d'exportation
- Masque de création de fichier (umask)
- Types de terminaux
- Messages indiquant l'arrivée d'un nouveau message électronique

L'administrateur système configure le fichier **/etc/profile** pour tous les utilisateurs du système. Lui seul est habilité à le modifier.

Voici un exemple de fichier **/etc/profile** :

```
#Set file creation mask
 unmask 022
 #Tell me when new mail arrives
 MAIL=/usr/mail/$LOGNAME
 #Add my /bin directory to the shell search sequence
 PATH=/usr/bin:/usr/sbin:/etc::
 #Set terminal type
TERM=lft
 #Make some environment variables global
 export MAIL PATH TERM
```

Reportez–vous au manuel *AIX 5L Version 5.3 Files Reference* pour obtenir des informations détaillées sur le fichier **/etc/profile**.

Fichier .profile

Le troisième fichier utilisé par le système au moment de la connexion est le fichier **.profile.** Le fichier**.profile** se trouve dans votre répertoire personnel (**$HOME**) et permet de personnaliser votre environnement de travail individuel. Etant donné que le fichier **.profile** est caché, utilisez la commande **ls–a** pour l'afficher.

Une fois que le programme **login** a intégré les variables **LOGNAME** (nom de connexion) et **HOME** (répertoire de connexion) à l'environnement, les commandes du fichier **$HOME/.profile** (si présent) sont exécutées. Le fichier **.profile** contient votre propre profil, qui prime sur les variables définies dans le fichier **/etc/profile**. Le fichier **.profile** est souvent utilisé pour définir les variables d'environnement exportées et les modes du terminal. Modifiez le fichier **.profile** pour adapter précisément votre environnement à vos besoins. Utilisez le fichier **.profile** pour contrôler les paramètres par défaut suivants :

- Shells à ouvrir
- Style de l'invite
- Son du clavier

Voici un exemple de fichier **.profile** :

```
PATH=/usr/bin:/etc:/home/bin1:/usr/lpp/tps4.0/user::
epath=/home/gsc/e3:
export PATH epath
csh
```

Dans cet exemple, deux variables de chemin (`PATH` et `epath`), sont définies, puis exportées, et un shell C (`csh`) est ouvert.

Vous pouvez également utiliser le fichier **.profile** (ou, à défaut, le fichier **/etc/profile**) pour déterminer les variables du shell de connexion. Vous pouvez également personnaliser d'autres environnements shell. Par exemple, utilisez le fichier **.cshrc** et le fichier **.kshrc** pour personnaliser un shell C et un shell Korn (respectivement).

Fichier .env

Un quatrième fichier utilisé par le système au moment de la connexion est le fichier **.env**, sous réserve que votre fichier **.profile** contienne la ligne : `export ENV=$HOME/.env`

Le fichier **.env** permet de personnaliser vos propres variables d'environnement de travail. Etant donné que le fichier **.env** est caché, utilisez la commande **ls–a** pour l'afficher. Le fichier **.env** contient le profil personnel de l'utilisateur, qui prime sur les variables définies dans le fichier **/etc/environment**. Vous pouvez personnaliser à volonté vos variables d'environnement en modifiant votre fichier **.env**.

Voici un exemple de fichier**.env** :

```
export  myid=‘id | sed –n –e ’s/).*$//’ –e ’s/^.*(//p’‘
 #set prompt: login & system name & path
 if [ $myid = root ]
         then    typeset –x PSCH=’#:\${PWD}> ’
                 PS1=”#:\${PWD}> ”
         else    typeset –x PSCH=’>’
                 PS1=”$LOGNAME@$UNAME:\${PWD}> ”
                 PS2=”>”
                 PS3=”#?”
 fi
 export PS1 PS2 PS3
 #setup my command aliases
 alias   ls=”/bin/ls –CF” \
         d=”/bin/ls –Fal | pg” \
         rm=”/bin/rm –i” \
         up=”cd ..”
```

Remarque : Lorsque vous modifiez le fichier **.env**, veillez à ce que les variables nouvellement définies n'entrent pas en conflit avec les variables standard telles que **MAIL**, **PS1**, **PS2** et **IFS**.

Fichiers de lancement du système AIXwindows : généralités

Etant donné que les procédures de lancement de X Server et AIXwindows varient selon le système informatique, informez–vous auprès de votre administrateur système pour connaître la procédure de lancement. En règle générale, X Server et AIXwindows sont lancés via un script shell, exécuté automatiquement au moment de la connexion. Mais il vous faudra peut–être lancer l'un et/ou l'autre explicitement.

Si, une fois connecté, vous vous trouvez devant un écran simple, sans fenêtres, vous pouvez lancer X Server en entrant la commande suivante :

```
xinit
```

Si cette commande reste sans effet, vérifiez, avec l'aide de votre administrateur, que votre chemin de recherche contient bien le répertoire X11, dans lequel se trouvent les programmes exécutables. Ce chemin varie selon les systèmes.

> **Remarque :** Avant de lancer cette commande, vérifiez que le pointeur se trouve dans une fenêtre affichant une invite système.

Si, une fois connecté, vous vous trouvez avec une ou plusieurs fenêtres sans cadre, vous pouvez lancer AIXwindows Window Manager en entrant :

```
mwm &
```

Etant donné que AIXwindows autorise la personnalisation à la fois aux programmeurs (via des applications AIXwindows) et aux utilisateurs, vous constaterez peut–être que les boutons de la souris ou d'autres fonctions ne fonctionnent pas exactement comme indiqué dans ce manuel. Il vous suffit dans ce cas de réinitialiser votre environnement AIXwindows en appuyant simultanément sur les quatre touches suivantes :

Alt–Ctrl–Maj–!

Vous pouvez rétablir le comportement personnalisé en appuyant de nouveau sur cette combinaison de touches. Si votre système ne permet pas cette combinaison de touches, vous pouvez également restaurer le comportement par défaut à partir du menu racine par défaut.

.xinitrc file

La commande **xinit** se sert d'un fichier script shell personnalisable qui liste les programmes client X à démarrer. Le fichier **.xinitrc** de votre répertoire personnel contrôle les fenêtres et les applications qui sont lancées en même temps que AIXwindows.

La commande **xinit** fonctionne avec les scripts shell dans l'ordre suivant :

1. Pour lancer AIXwindows, la commande **xinit** recherche d'abord la variable d'environnement **$XINITRC**.
2. Si la variable d'environnement **$XINITRC** ne peut être retrouvée, la commande **xinit** cherche le script shell **$HOME/.xinitrc**.
3. Si elle ne le trouve pas non plus, elle lance le script shell **/usr/lib/X11/$LANG/xinitrc**.
4. Si **/usr/lib/X11/$LANG/xinitrc** est introuvable, la commande recherche le script shell **/usr/lpp/X11/defaults /$LANG/xinitrc**. En cas d'échec, elle recherche le script shell **/usr/lpp/X11/defaults/xinitrc**.
5. Le script shell **xinitrc** lance des commandes telles que **mwm** (AIXwindows Window Manager), **aixterm** et **xclock.**

La commande **xinit** :

- lance un serveur X sur l'écran courant ;
- définit la variable d'environnement **$DISPLAY** ;
- exécute le fichier **xinitrc** pour lancer les programmes client X ;

Voici la partie personnalisable du fichier **xinitrc** :

```
# Ce script est appelé par /usr/lpp/X11/bin/xinit

.
.
.
#************************************************************
#  Start the X clients. Change the following lines to       *
#  whatever command(s) you desire!                          *
#  The default clients are an analog clock (xclock), an lft *
#  terminal emulator (aixterm), and the Motif Window  Manager*
#  (mwm).                                                   *
#************************************************************
exec mwm
```

Fichier .Xdefaults

Si vous travaillez dans une interface AIXwindows, vous pouvez personnaliser cette interface avec le fichier **.Xdefaults**. AIXwindows vous permet d'indiquer vos préférences en matière de caractéristiques visuelles, comme les couleurs et les polices.

L'aspect et le comportement des applications utilisant des fenêtres sont en grande partie régis par des ensembles de variables appelées *ressources*. L'aspect, visuel ou fonctionnel, d'une ressource dépend de la valeur qui lui est attribuée. Il existe plusieurs types de valeurs : par exemple, une ressource contrôlant les couleurs peut se voir affecter la valeur prédéfinie *DarkSlateBlue* ou *Black*. Des valeurs numériques sont attribuées aux ressources qui indiquent des dimensions. Certaines ressources prennent des valeurs booléennes (*True* ou *False*).

Si votre répertoire personnel ne contient pas de fichier **.Xdefaults**, vous pouvez en créer un à l'aide d'un éditeur texte quelconque. Vous avez ensuite toute latitude pour y définir les ressources de votre choix. Un fichier modèle, **Xdefaults.tmpl**, se trouve dans le répertoire **/usr/lpp/X11/defaults**.

Voici un extrait d'un fichier **.Xdefaults** :

```
*AutoRaise: activé
*DeIconifyWarp: activé
*warp:on
*TitleFont:andysans12
*scrollBar: true
*font: Rom10.500
Mwm*menu*foreground: black
Mwm*menu*background: CornflowerBlue
Mwm*menu*RootMenu*foreground: black
Mwm*menu*RootMenu*background: CornflowerBlue
Mwm*icon*foreground: grey25
Mwm*icon*background: LightGray
Mwm*foreground: black
Mwm*background: LightSkyBlue
Mwm*bottomShadowColor: Blue1
Mwm*topShadowColor: CornflowerBlue
Mwm*activeForeground: white
Mwm*activeBackground: Blue1
Mwm*activeBottomShadowColor: black
Mwm*activeTopShadowColor: LightSkyBlue
Mwm*border: black
Mwm*highlight:white
```

```
aixterm.foreground: green
aixterm.background: black
aixterm.fullcursor: true
aixterm.ScrollKey: on
aixterm.autoRaise: true
aixterm.autoRaiseDelay: 2
aixterm.boldFont:Rom10.500
aixterm.geometry: 80x25
aixterm.iconFont: Rom8.500
aixterm.iconStartup: false
aixterm.jumpScroll: true
aixterm.reverseWrap: true
aixterm.saveLines: 500
aixterm.scrollInput: true
aixterm.scrollKey: false
aixterm.title: AIX
```

Fichier .mwmrc

Les caractéristiques le plus souvent personnalisées correspondant à des ressources du fichier **.Xdefaults**. Toutefois, les affectations des touches et des boutons de la souris, de même que les définitions de menus du gestionnaire de fenêtres sont spécifiées dans un fichier complémentaire, **.mwmrc**, référencé par des ressources du fichier **.Xdefaults.**

Si votre répertoire personnel ne contient pas de fichier **.mwmrc,** vous pouvez le copier via la commande :

```
cp /usr/lib/X11/system.mwmrc .mwmrc
```

Les spécifications du fichier **.mwmrc** priment sur celles, globales, définies dans le fichier **system.mwmrc**. Ainsi, vous ne risquez pas d'interférer avec les spécifications des autres.

Voici un extrait d'un fichier **system.mwmrc** type :

```
#  DEFAULT mwm RESOURCE DESCRIPTION FILE (system.mwmrc)
#
# menu pane descriptions
#
# Root Menu Description

Menu RootMenu
 { "Root Menu"        f.title
   no-label           f.separator
   "New Window"       f.exec "aixterm &"
   "Shuffle Up"       f.circle_up
   "Shuffle Down"     f.circle_down
   "Refresh"          f.refresh
   no-label           f.separator
   "Restart"          f.restart
   "Quit"             f.quit_mwm
 }

# Default Window Menu Description

Menu DefaultWindowMenu MwmWindowMenu
{ "Restore"   _R   Alt<Key>F5              f.normalize
  "Move"      _M   Alt<Key>F7              f.move
  "Size"      _S   Alt<Key>F8              f.resize
  "Minimize"  _n   Alt<Key>F9              f.minimize
  "Maximize"  _x   Alt<Key>F10             f.maximize
  "Lower"     _L   Alt<Key>F3              f.lower
   no-label           f.separator
  "Close"     _C   Alt<Key>F4              f.kill
 }
```

```
# no accelerator window menu
Menu NoAccWindowMenu
 {
  "Restore"    _R     f.normalize
  "Move"       _M     f.move
  "Size"       _S     f.resize
  "Minimize"   _n     f.minimize
  "Maximize"   _x     f.maximize
  "Lower"      _L     f.lower
   no-label           f.separator
  "Close"      _C     f.kill
 }

Keys DefaultKeyBindings
 {
  Shift<Key>Escape             icon|window        f.post_wmenu
  Meta<Key>space               icon|window        f.post_wmenu
  Meta<Key>Tab                 root|icon|window   f.next_key
  Meta Shift<Key>Tab           root|icon|window   f.prev_key
  Meta<Key>Escape              root|icon|window   f.next_key
  Meta Shift<Key>Escape        root|icon|window   f.prev_key
  Meta Ctrl Shift<Key>exclam   root|icon|window   f.set_behavior
 }

#
# button binding descriptions
#

Buttons DefaultButtonBindings
 {
   <Btn1Down>          frame|icon             f.raise
   <Btn3Down>          frame|icon             f.post_wmenu
   <Btn1Down>          root                   f.menu  RootMenu
   <Btn3Down>          root                   f.menu  RootMenu
   Meta<Btn1Down>      icon|window            f.lower
  Meta<Btn2Down>      window|icon            f.resize
   Meta<Btn3Down>      window                 f.move
 }

Buttons PointerButtonBindings
 {
   <Btn1Down>          frame|icon             f.raise
   <Btn2Down>         frame|icon             f.post_wmenu
   <Btn3Down>         frame|icon             f.lower
   <Btn1Down>          root                   f.menu  RootMenu
  Meta<Btn2Down>      window|icon            f.resize
  Meta<Btn3Down>     window|icon            f.move
 }

#
#  END OF mwm RESOURCE DESCRIPTION FILE
#
```

Personnalisation de votre environnement système

Cette section décrit les procédures suivantes qui permettent de personnaliser votre environnement système :

Exportation de variables shell (commande shell export)

Une variable shell *locale* est une variable reconnue uniquement par le shell qui l'a créée. Si vous lancez un nouveau shell, toutes les variables définies pour l'ancien shell lui sont inconnues. Si vous voulez que les nouveaux shell que vous ouvrez utilisent les variables d'un shell existant, exportez les variables pour les rendre *globales*.

Vous pouvez utiliser la commande **export** pour transformer les variables locales en variables globales. Pour automatiser le processus, exportez-les dans le fichier **.profile**.

Remarque : Les variables peuvent être exportées vers des shells enfant, mais pas vers des shells parent.

Voir les exemple ci–après :

- Pour transformer la variable shell locale **PATH** en variable globale, entrez :

```
export path
```

- Pour afficher toutes vos variables exportées, entrez :

```
export
```

Le système affiche des informations similaires à l'exemple suivant :

```
DISPLAY=unix:0
EDITOR=vi
ENV=$HOME/.env
HISTFILE=/u/denise/.history
HISTSIZE=500
HOME=/u/denise
LANG=En_US
LOGNAME=denise
MAIL=/usr/mail/denise
MAILCHECK=0
MAILMSG=**YOU HAVE NEW MAIL.
USE THE mail COMMAND TO SEE YOUR MAILPATH=/usr/mail/denise?denise has mail
!!!
MAILRECORD=/u/denise/.Outmail

PATH=/usr/ucb:/usr/lpp/X11/bin:/bin:/usr/bin:/etc:/u/denise:/u/denise/bin:/
u/bin1
PWD=/u/denise
SHELL=/bin/ksh
```

Changement de la police par défaut (commande chfont)

Vous pouvez modifier la police par défaut au lancement du système à l'aide de la commande **chfont** ou **smit**. Une *palette de polices* est un fichier que le système utilise pour définir et identifier les polices disponibles.

Remarque : Pour exécuter la commande **chfont**, vous devez être un utilisateur racine.

Utilisation de la commande chfont

Les exemples ci–après illustrent les différentes utilisations de la commande **chfont** :

- Pour passer à la cinquième police de la palette, entrez :

```
chfont –a5
```

- Pour passer à la police Roman, en italique et en gras, en conservant la même taille, entrez :

```
chfont –n /usr/lpp/fonts/It114.snf /usr/lpp/fonts/Bld14.snf
/usr/lpp/fonts/Rom14.snf
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **chfont** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Commande smit

Vous pouvez également exécuter la commande **chfont** via **smit**.

Pour sélectionner la police active, entrez :

```
smit chfont
```

Pour sélectionner la palette de polices, entrez :

```
smit chfontpl
```

Changement de l'affectation des touches de contrôle (commande stty)

Vous pouvez modifier l'affectation des touches de contrôle à l'aide de la commande **stty**. Les changements sont appliqués jusqu'à la déconnexion. Pour les rendre permanents, placez-les dans le fichier **.profile**.

Voir les exemple ci–après :

- Par exemple, pour définir Ctrl–Z comme touche d'interruption, entrez :

```
stty intr ^Z
```

  N'oubliez pas d'insérer un espace entre intr et ^Z.

- Pour restaurer les touches de contrôle à leurs valeurs par défaut, entrez :

```
stty sane
```

- Pour afficher vos paramètres actuels, entrez :

```
stty –a
```

Pour obtenir des détails sur la syntaxe, reportez–vous à la description de la commande **stty** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Changement de l’invite système

Le shell utilise trois variables d’invite :

PS1	Invite normale.
PS2	Invite destinée à entrer des données supplémentaires
PS3	Invite indiquant que vous détenez des droits utilisateur racine

Pour changer le contenu d’une invite, il suffit de modifier la variable correspondante. Les changements apportés aux invites s’appliquent jusqu’à la déconnexion. Pour les rendre permanents, placez-les dans le fichier **.env**.

Voir les exemple ci–après :

- Pour afficher la valeur actuelle de la variable PS1, entrez :

```
echo ”prompt is $PS1”
```

Le système affiche des informations similaires à l’exemple suivant :

```
prompt is $
```

- Pour changer votre invite en `Ready>`, entrez :

```
PS1=”Ready> ”
```

- Pour changer votre invite de continuation en `Enter more->`, entrez :

```
PS2=”Enter more->”
```

- Pour changer votre invite racine en `Root->`, entrez :

```
PS3=”Root-> ”
```

Récapitulatif des commandes de personnalisation

Fichiers de lancement du système

/etc/profile	Fichier système contenant les commandes exécutées par le système lorsque vous vous connectez
/etc/environment	Fichier système contenant des variables spécifiant l'environnement de base de tous les process
$HOME/.profile	Fichier dans votre répertoire qui contient des commandes qui remplacent le système **/etc/profile** à la connexion. Pour plus d'informations, reportez–vous à la section Fichier .profile, page 11-3.
$HOME/.env	Fichier de votre répertoire personnel contenant des commandes primant sur celles de l'environnement système **/etc/environment** et contenant des variables spécifiant l'environnement de base de tous les process. Pour en savoir plus, reportez–vous à la section Fichier .env, page 11-4.

Fichiers de lancement AIXwindows

$HOME/.xinitrc	Fichier de votre répertoire personnel contrôlant les fenêtres et les applications lancées en même temps qu'AIXwindows. Pour en savoir plus, reportez–vous à la section Fichier .xinitrc, page 11-5.
$HOME/.Xdefaults	Fichier de votre répertoire personnel contrôlant l'aspect visuel et fonctionnel des ressources AIXwindows Pour en savoir plus, reportez–vous à la section Fichier .Xdefaults, page 11-6.
$HOME/.mwmrc	Fichier de votre répertoire personnel définissant les affectations des touches et des boutons de la souris, et la définition des menus pour le gestionnaire de fenêtres. Pour en savoir plus, reportez–vous à la section Fichier .xinitrc, page 11-7.

Procédures de personnalisation

PS1	Invite système normale
PS2	Invite d'entrées complémentaires
PS3	Invite système root
chfont	Change la police d'affichage sur un écran (après relance du système).
stty	Définit, redéfinit et fait état des paramètres d'exploitation d'une station de travail.

Chapitre 12. Shells : généralités

Les *shells* assurent l'interface avec le système d'exploitation. Ils constituent la couche externe du système d'exploitation. Intégrant un langage de programmation, ils sont à même de contrôler process et fichiers, et de gérer et de contrôler d'autres programmes. Ils vous invitent à entrer des données, les interprètent et traitent les résultats générés par le système.

Ils assurent la communication avec le système d'exploitation, communication qui s'effectue soit en mode interactif (les entrées au clavier sont immédiatement interprétées), soit via des scripts shell. Un *script shell* est une séquence de commandes (shell et système), enregistrée dans un fichier.

Lors de votre connexion au système, le nom du shell à exécuter est détecté. Une fois le shell exécuté, une invite s'affiche, généralement un signe dollar ($). Lorsque vous tapez une commande et appuyez sur Entrée, le shell évalue la commande et tente de l'exécuter. Le résultat est ensuite dirigé soit vers l'écran, soit réacheminé, selon vos instructions. L'invite est ensuite réaffichée, dans l'attente de la commande suivante.

La ligne sur laquelle vous tapez les commandes est une *ligne de commande*. Elle contient l'invite shell. Son format de base est le suivant :

```
$ Commande Argument(s)
```

Le shell interprète le premier mot (jusqu'au premier espace) comme étant le nom de la commande, et les mots suivants comme des arguments.

Remarque : Lorsque **libc.a** est déplacé ou renommé, le message d'erreur **Killed** s'affiche à partir du shell car il n'y a pas de fichier **libc.a** disponible pour que le système puisse charger et exécuter les programmes utilitaires. La commande **recsh** appelle le shell de reprise qui permet de renommer un fichier **libc.a** déplacé malencontreusement.

Ce chapitre traite des points suivants :

Caractéristiques des shells

Communiquer avec le système via une interface shell présente les avantages suivants :

- **Substitution de caractères génériques dans les noms de fichiers (correspondance)**

  Exécute des commandes sur un groupe de fichiers, identifiés par une trame (et non sur un fichier explicitement nommé).

  Pour en savoir plus, reportez–vous aux sections suivantes :

  – Substitution de noms de fichiers (shell Korn ou POSIX), page 12-33

  – Substitution de noms de fichiers (shell Bourne), page 12-84

  – Substitution de noms de fichiers (shell C), page 12-110

- **Traitement en arrière–plan**

  Exécute les tâches lourdes en arrière-plan, libérant le terminal pour des traitements interactifs concurrents.

  Pour en savoir plus, reportez–vous à la commande **bg** dans les sections suivantes :

  – Contrôle des travaux (shell Korn ou POSIX), page 12-56

  – Commandes intégrées (shell C), page 12-94

  Remarque : Le shell Bourne ne prend pas en charge le contrôle des travaux.

- **Alias de commandes**

  Attribue un alias à une commande ou à une expression. Lorsqu'un alias figure dans une ligne de commande ou dans un script shell, le système remplace cet alias par le texte qu'il représente.

  Pour en savoir plus, reportez–vous aux sections suivantes :

  – Alias de commandes (shell Korn ou POSIX), page 12-20

  – Substitution d'alias (shell C), page 12-107

  Remarque : Le shell Bourne ne prend pas en charge les alias de commandes.

- **Historique des commandes**

  Consigne dans un fichier d'historique les commandes lancées. Vous pouvez utiliser ce fichier pour appeler, modifier ou relancer une commande enregistrée.

  Pour en savoir plus, reportez–vous à la commande **history** dans les sections suivantes :

  – Historique des commandes (shell Korn ou POSIX), page 12-14

  – Commandes intégrées (shell C), page 12-94

  – Substitution d'historique (shell C), page 12-104

  Remarque : Le shell Bourne ne prend pas en charge l'historique des commandes.

- **Substitution de noms de fichiers**

  Génère automatiquement une liste de noms de fichiers sur une ligne de commande sur la base de caractères génériques.

  Pour en savoir plus, reportez–vous à :

  – Substitution de noms de fichiers (shell Korn ou POSIX), page 12-33

  – Substitution de noms de fichiers (shell Bourne), page 12-84

  – Substitution de noms de fichiers (shell C), page 12-110

- **Réacheminement des entrées/sorties**

  Lit des données à partir d'une source autre que le clavier et réachemine les sorties vers un fichier ou une unité autre que le terminal. Par exemple, les entrées destinées à un programme peuvent être extraites d'un fichier et réacheminées vers l'imprimante ou vers un autre fichier.

  Pour en savoir plus, reportez–vous à :

  – Réacheminement des entrées/sorties (shell Korn ou POSIX), page 12-35
  – Réacheminement des entrées/sorties (shell Bourne), page 12-90
  – Réacheminement des entrées/sorties (shell C), page 12-116

- **Traitement "pipeline"**

  Relie des commandes entre elles pour former un programme complexe : la sortie standard de chaque programme constitue l'entrée standard du suivant.

  Pour en savoir plus, reportez–vous à la définition de **traitement "pipeline"**, page 12-5 dans la section relative à la terminologie des shells, page 12-5.

- **Substitution de variables shell**

  Enregistre des données dans des variables définies par l'utilisateur et dans des variables shell prédéfinies.

  Pour en savoir plus, reportez–vous aux sections suivantes :

  – Substitution de paramètres (shell Korn ou POSIX), page 12-22
  – Substitution de variables (shell Bourne), page 12-83
  – Substitution de variables (shell C), page 12-108.

Shells disponibles

Les shells fournis avec le système d'exploitation sont les suivants :

- shell Korn (lancé par la commande **ksh**) ;
- shell Bourne (lancé par la commande **bsh**) ;
- Shell restreint (version limitée du shell Bourne, lancé par la commande **Rsh**)
- shell POSIX (également appelé shell Korn, lancé par la commande **psh**) ;
- shell par défaut (lancé par la commande **sh**) ;
- shell C (lancé par la commande **csh**) ;
- Shell sécurisé (version limitée du shell Korn, lancé par la commande **tsh**)
- shell distant (lancé par la commande **rsh**).

Le *shell de connexion* est lancé lors de la connexion. Votre shell de connexion est défini dans le fichier **/etc/passwd**. Le shell Korn (voir page 12-9) est le shell de connexion standard. Il présente une compatibilité descendante avec le shell Bourne (voir Shell Bourne, page 12-70).

Le *shell par défaut* ou *shell standard* est le shell lié à et lancé par la commande **usr/bin/sh**. Le shell Bourne est configuré comme shell par défaut et est un sous–ensemble du shell Korn.

La commande **/usr/bin/sh** réside comme copie du shell Korn, qui est **/usr/bin/ksh**.
Par conséquent, le shell Korn peut être remplacé comme shell par défaut. Le shell POSIX appelé par la commande **/usr/bin/psh** demeure un lien vers la commande **/usr/bin/sh**.

Terminologie des shells

Voici quelques définitions utiles :

blanc (espace)	Un des caractères de la classe de blancs définie dans la catégorie LC_CTYPE. Dans le shell POSIX, un blanc est soit une tabulation, soit un espace.
commande intégrée	Commande que le shell exécute sans la rechercher ni créer de process distinct.
commande	Séquence de caractères construite suivant la syntaxe du langage shell. Le shell lit chaque commande, puis exécute l'action demandée directement ou par l'intermédiaire d'utilitaires.
commentaire	Mot préfixé par le signe dièse (#) : tout caractère ou chaîne de caractères situé entre ce signe et le caractère nouvelle ligne est ignoré.
identificateur	Séquence de lettres, chiffres ou symboles de soulignement introduite par une lettre ou un symbole de soulignement. Le premier caractère d'un identificateur ne peut pas être un chiffre. Les identificateurs servent de noms pour les alias, les fonctions et les paramètres nommés.
liste	Séquence d'un ou de plusieurs pipelines séparés par l'un des symboles suivants : point–virgule (;), perluète (&), double perluète (&&) ou double barre verticale (||). La fin de la liste est indiquée par l'un des symboles suivants : point– virgule (;), perluète (&) ou barre verticale, perluète (|&). **;** Le shell traite le pipeline précédent en mode séquentiel. Il exécute successivement les commandes jusqu'à la dernière, dont il attend la fin de l'exécution. **&** Le shell traite le pipeline précédent en mode asynchrone. Il exécute successivement les commandes, traitant le pipeline en arrière–plan sans attendre la fin de son exécution. **|&** Le shell traite le pipeline précédent en mode asynchrone et établit un canal bilatéral vers le shell parent. Il exécute successivement les commandes, traitant le pipeline en arrière–plan sans attendre la fin de son exécution. Les commandes **read –p** et **print –p** permettent au shell parent de lire depuis l'entrée standard et d'écrire vers la sortie standard de la commande générée dynamiquement. Une seule commande peut être activée à la fois. **&&** Le shell traite la liste située après ce symbole si le résultat du pipeline précédent est la valeur zéro (0). **||** Le shell traite la liste située après ce symbole si le résultat du pipeline précédent est une valeur différente de 0 (zéro). Les symboles point–virgule (`;`), perluète (`&`) et barre verticale+perluète (`|&`) ont une priorité moindre que les symboles double perluète (`&&`) et double barre verticale (`||`). Les symboles `;`, `&`, et `|&` ont la même priorité. De même, les symboles `&&` et `||` ont la même priorité. Un ou plusieurs caractères ligne suivante peuvent remplacer le symbole point virgule pour délimiter deux commandes dans une liste. **Remarque :** Le symbole `|&` est valable uniquement dans le shell Korn.

métacaractère	Un métacaractère est un caractère qui a une signification spéciale pour le shell et marque toujours la fin d'un mot, sauf s'il est placé entre apostrophes. Les symboles considérés comme des métacaractères sont : barre verticale (|), perluète (&), point–virgule (;), inférieur à (<), supérieur à (>), parenthèse gauche ((), parenthèse droite ()), dollar ($), apostrophe gauche (`), barre oblique inverse (\), apostrophe droit ('), guillemets ("), caractère de ligne suivante, espace et tabulation. Les caractères encadrés d'apostrophes, dits déclarés, sont interprétés littéralement. Sinon, c'est leur signification spéciale qui est prise en compte. (Dans le shell C, les métacaractères sont aussi appelés *métacaractères d'analyse sémantique*.)
liste d'affectation des paramètres	Liste de mots de la forme *Identificateur* = *Valeur*. Il doit y avoir le même nombre d'espaces de part et d'autre du signe égal (=) ou aucun espace. **Remarque :** Dans le shell C, cette liste est de la forme **set** *Identificateur* = *Valeur*. Les espaces encadrant le signe = (égal) sont obligatoires.
pipeline	Séquence de commandes séparées par une barre verticale (|). Chaque commande (sauf éventuellement la dernière) est exécutée séparément, mais la sortie standard d'une commande constitue l'entrée standard de la suivante. Une liste de commandes entre parenthèses est exécutée comme simple commande d'un sous–shell distinct. Si le pipeline n'est pas précédé du mot réservé !, son état de sortie est celui de la dernière commande. Sinon, c'est l'opposé (NON logique) de l'état de cette dernière commande. En d'autres termes, si la dernière commande renvoie zéro, l'état de sortie est 1. Si elle renvoie une valeur supérieure à zéro, l'état de sortie est zéro. Le format d'un pipeline est le suivant : `[!] commande1 [	commande2 ...]` **Remarque :** Dans les premières versions du shell Bourne, un canal était symbolisé par un (^).
variable shell	Nom ou paramètre auquel est affectée une valeur. Pour affecter une valeur, entrez le nom de la variable suivi d'un signe égal (=) et de la valeur souhaitée. Vous pouvez remplacer le nom de la variable par la valeur affectée en préfixant le nom de la variable d'un signe dollar ($). Les variables sont particulièrement utiles pour abréger un nom de chemin trop long : **$HOME**, par exemple, donne un accès direct au répertoire personnel. Une variable prédéfinie est une variable dont la valeur est attribuée par le shell. Une variable définie par l'utilisateur est une variable dont la valeur est attribuée par l'utilisateur.
commande simple	Séquence de listes d'affectations et de réacheminements de paramètres, dans un ordre quelconque. Elle est éventuellement suivie de commandes, de mots et de réacheminements. Elle est terminée par ; , |, &,	|, &&, |&, ou un caractère nouvelle ligne. Le nom de la commande est passé comme paramètre 0 (tel que défini par la sous–routine **exec**). La valeur d'une commande simple est son état en sortie (zéro si elle a abouti, différent de zéro, sinon). Reportez–vous à la sous–routine **sigaction, sigvec**, or **signal** dans le manuel *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 2* pour la liste des valeurs d'état en sortie.
sous–shell	Shell exécuté comme enfant du shell de connexion ou du shell courant.

caractère générique	Aussi appelé *métacaractère*. Symbole spécial affecté d'une valeur par le shell. Les caractères génériques standard sont les suivants : **?**, *****, [ensemble], et [!ensemble]. Ils sont particulièrement utiles pour effectuer des substitutions de noms de fichiers.
mot	Séquence de caractères exempte de blancs. Les mots sont séparés par un ou plusieurs métacaractères.

Création et exécution d'un script shell

Un *shell script* est un fichier qui contient une ou plusieurs commandes. Les shell scripts permettent d'exécuter facilement des commandes fastidieuses, des séquences de commandes importantes ou complexes et des tâches de routine. Lorsque vous saisissez le nom d'un fichier script shell, le système exécute la séquence de commandes figurant dans ce fichier.

Pour créer un shell script, utilisez un éditeur de texte. Le script peut contenir des commandes du système d'exploitation et des commandes intégrées du shell.

Pour écrire un script shell, procédez comme suit.

1. Via un éditeur de texte, créez et sauvegardez un fichier. Vous pouvez y inscrire n'importe quelle combinaison de commandes système et de commandes shell. Par convention, les scripts shell non destinés à être exploités par plusieurs utilisateurs sont enregistrés dans le répertoire **$HOME/bin.**

   Remarque : Le système d'exploitation ne prend pas en charge les sous–routines **setuid** et **setgid** dans un script shell.

2. Lancez la commande **chmod** pour limiter au seul propriétaire le droit d'exécuter le fichier. Par exemple, si le fichier s'appelle `script1`, entrez :

```
chmod u=rwx script1
```

3. Saisissez le nom du script shell sur la ligne de commande pour l'exécuter. Pour exécuter le script **script1**, entrez :

```
script1
```

   Remarque : Vous pouvez exécuter un script shell sans l'avoir rendu exécutable, sous réserve que son nom soit précédé d'une commande shell (**ksh**, **bsh** ou **csh**) sur la ligne de commande. Par exemple, pour exécuter le fichier non–exécutable **script1** sous le shell Korn, entrez :

   ```
   ksh script1
   ```

Spécification d’un shell pour un fichier script

Lorsque vous exécutez un script shell sous le shell Korn (shell POSIX) ou le shell Bourne, les commandes sont exécutées sous contrôle du shell courant – sauf spécification contraire explicite. Lorsque vous exécutez un script shell sous le shell C, les commandes sont exécutées sous contrôle du shell Bourne (**/usr/bin/bsh**) sauf si vous spécifiez un autre shell.

Il est possible d’exécuter un script shell dans un shell spécifique en incluant ce shell dans le script shell.

Pour lancer un script shell exécutable sous un shell spécifique, tapez `#! Chemin` sur la première ligne de commande du script et appuyez sur Entrée. Les caractères `#!` identifient le type de fichier. La variable *Chemin* indique le chemin du shell sous lequel le script est exécuté.

Par exemple, pour exécuter le script **bsh** dans le shell Bourne, entrez :

```
#!/usr/bin/bsh
```

Si vous précédez d’une commande shell le nom d’un fichier script (sur la ligne de commande), le shell ainsi spécifié prime sur celui éventuellement indiqué dans le script lui–même. Ainsi, si vous tapez la commande `ksh myfile` et appuyez sur Entrée, cela exécute le fichier **myfile** sous le shell Korn, même si la première ligne de **myfile** est `#!/usr/bin/csh`.

Commandes du shell Korn ou POSIX

Le shell Korn est un interpréteur de commandes interactif et un langage de programmation. Il est conforme à l'environnement POSIX (Portable Operating System Interface for Computer Environments), standard international pour les systèmes d'exploitation. POSIX n'est pas un système d'exploitation, mais un *standard* ayant pour objet la portabilité des applications, au niveau source, entre plusieurs systèmes. Les caractéristiques POSIX sont construites au niveau supérieur du shell Korn. Le shell Korn (également appelé shell POSIX) offre, outre nombre des fonctions des shells Bourne et C (réacheminement des E/S, substitution de variables et de noms de fichiers, etc.), un certain nombre de fonctions spécifiques. En outre, il inclut un certain nombre de commandes supplémentaires et de caractéristiques de langages de programmation :

Remarque : Il existe une version restreinte du shell Korn, appelée **rksh**. Pour plus de détails, reportez–vous à la description de la commande **rksh**.

Evaluation arithmétique	Le shell Korn ou le shell POSIX peut effectuer des opérations arithmétiques sur des nombres entiers avec la commande intégrée **let** à l'aide de n'importe quelle base comprise entre 2 et 36. Pour plus d'informations, reportez–vous à la section Evaluation arithmétique (shell Korn ou POSIX), page 12-30.
Historique des commandes	Le shell Korn, ou POSIX, contient un fichier qui enregistre toutes les commandes entrées. Vous pouvez modifier n'importe quelle commande de ce fichier à l'aide d'un éditeur de texte, puis la réutiliser. Pour en savoir plus, reportez–vous à la section Historique des commandes, page 12-14.
Fonction coprocess	La fonction coprocess permet d'exécuter des process en arrière–plan, de leur envoyer des informations et d'en recevoir. Pour en savoir plus, reportez–vous à la description de la fonction Fonction Coprocess, page 12-37.
Modification	Les options d'édition permettent de modifier la ligne de commande. Des éditeurs semblables à emacs, gmacs et vi sont disponibles. Pour en savoir plus, reportez–vous à la section Modification en ligne (shell Korn ou POSIX), page 12-58.

Une commande du shell Korn est l'un des éléments suivants :

- Commande simple, page 12-6
- Pipeline, page 12-5
- Liste, page 12-5
- Commande composée, page 12-10
- Fonction, page 12-13

Lorsque vous lancez une commande dans le shell Korn ou POSIX, il l'analyse et effectue les actions suivantes :

- il effectue toutes les substitutions indiquées ;
- Détermine si la commande contient une barre oblique (/). Dans l'affirmative, le shell exécute le programme nommé par le chemin spécifié.

Dans la négative, le shell effectue les opérations suivantes :

- Il détermine s'il s'agit d'une commande intégrée spéciale. Dans l'affirmative, il exécute la commande dans le process shell courant.

  Pour plus d'informations sur les commandes intégrées spéciales, reportez–vous à la section Commandes intégrées (shell ou POSIX), page 12-39.

- Il compare la commande aux fonctions définies par l'utilisateur. Si elle correspond à une de ces fonctions, les paramètres positionnels sont sauvegardés et réinitialisés par les arguments de l'appel de **fonction**. Lorsque la fonction s'achève ou émet un retour, la liste des paramètres positionnels est restaurée, et l'éventuel traitement d'interruption défini pour **EXIT** dans la fonction est exécuté. La valeur d'une fonction est celle de la dernière commande exécutée. Une fonction est exécutée dans le process shell courant.

- Si le nom de la commande correspond à celui d'une commande intégrée standard, cette commande est appelée.

  Pour plus d'informations sur les commandes intégrées régulières, reportez–vous à la section Commandes intégrées (shell ou POSIX), page 12-39.

- Il crée un process et tente d'exécuter la commande par le biais de la commande **exec** (s'il ne s'agit ni d'une commande intégrée, ni d'une fonction définie par l'utilisateur).

Le shell Korn, ou POSIX, explore les répertoires du chemin d'accès à la recherche d'un fichier exécutable. La variable shell **PATH** définit le chemin d'accès au répertoire contenant la commande. Les autres répertoires sont séparés par un signe (:). Le chemin par défaut est `/usr/bin:` (indiquant le répertoire **/usr/bin**, et le répertoire courant, dans cet ordre). Le répertoire courant est spécifié par des deux points contigus, ou par un caractère deux points au début ou à la fin de la liste des chemins.

Si le fichier est doté des droits d'exécution, mais n'est pas un répertoire ni un fichier **a.out**, le shell suppose qu'il contient les commandes shell. Le process shell en cours génère dynamiquement un sous–shell qui lit le fichier. Tous les alias non exportés, toutes les fonctions et tous les paramètres indiqués sont supprimés du fichier. Si le fichier de commande shell est accessible en lecture, ou que les bits **setuid** ou **setgid** sont définis dans le fichier, le shell lance un agent qui définit les droits d'accès et exécute le shell, le fichier de commande shell étant passé comme un fichier ouvert. Toute commande entre parenthèses est exécutée dans un sous-shell et les quantités non exportées ne sont pas supprimées.

Cette section traite des points suivants :

Commandes composées du shell Korn

Une commande composée peut être une liste de commandes simples, un pipeline ou une commande commençant par un mot réservé. Vous serez souvent amené à vous servir de commandes composées telles que **if**, **while** et **for** lorsque vous écrirez des scripts shell.

Liste des commandes composées du shell Korn ou POSIX

Syntaxe des commandes	Description
for *Identificateur*, page 12-5 [**in** *Mot*, page 12-7...]**;** **do** *Liste*, page 12-7 **;done**	A chaque itération de la commande **for**, *Identificateur* prend la valeur du mot suivant de la liste **in** *Mot*... A défaut de cette liste, c'est la commande **do** *Liste* qui est exécutée pour chaque paramètre positionnel défini. L'exécution se termine lorsqu'il n'y a plus de mots dans la liste. Pour plus d'informations sur les paramètres positionnels, reportez–vous à la section Substitution de paramètres (shell Korn ou POSIX), page 12-22.
select *Identificateur*, page 12-5 [**in** *Mot*, page 12-7...]**;do** *Liste*, page 12-7 **;done**	La commande **select** dirige vers la sortie d'erreur standard (descripteur de fichier 2) l'ensemble de mots spécifié, chaque mot étant précédé d'un numéro. Si la liste **in** *Mot*... est omise, ce sont les paramètres positionnels qui sont utilisés. L'invite **PS3** s'affiche et une ligne est lue à partir de l'entrée standard. Si cette ligne correspond à l'un des numéros des mots de la liste, *Identificateur* prend la valeur du mot correspondant à ce numéro. Si la ligne lue est vide, la liste de sélection s'affiche à nouveau. Sinon, *Identificateur* prend la valeur nulle. Le contenu de la ligne lue est sauvegardé dans le paramètre **REPLY**. Le paramètre *Liste* est exécuté pour chaque sélection jusqu'à ce qu'un caractère d'interruption ou de fin –de–fichier soit rencontré. Pour plus d'informations sur les paramètres positionnels, reportez–vous à la section Substitution de paramètres (shell Korn ou POSIX), page 12-22.
case *Mot*, page 12-7 **in** [[**(**] *Trame* [| *Trame*] ... **)** *Liste*, page 12-7 **;;**] ... **esac**	Une commande **case** exécute la *Liste* associée à la première *Trame* correspondant au *Mot*. Le format des trames est le même que celui utilisé pour les substitutions de noms de fichiers.
if *Liste*, page 12-7 **;then** *Liste* [**elif** *Liste***;then** *Liste*] ... [**;else** *Liste*]**;if**	*Liste* spécifie une liste de commandes à exécuter. Le shell exécute d'abord la commande **if** *Liste*. Si cette commande renvoie un état de sortie nul, il exécute la commande **then** *Liste*. Sinon, il exécute les commandes spécifiées par le paramètre *Liste* qui suit **elif**. Si la dernière commande de **elif** *Liste* renvoie la valeur zéro, la commande **then** *Liste* est exécutée. Si la dernière commande de **then** *Liste* renvoie la valeur zéro, la commande **else** *Liste* est exécutée. Si aucune des commandes spécifiées par les paramètres *Liste* pour **else** ou **then** n'est exécutée, **if** renvoie un état de sortie nul.
while *Liste*, page 12-7 **;** **do** *Liste***;done until** *Liste*, page 12-7 **;do** *Liste***;done**	

until *Liste* **;do** *Liste***;done**	*Liste* spécifie une liste de commandes à exécuter. *Liste* spécifie une liste de commandes, exécutées itérativement par la commande **while**. Si l'état de sortie de la dernière commande renvoie une valeur nulle, le shell exécute la commande **do** *Liste*. Si l'état de sortie de la dernière commande renvoie une valeur non nulle, la boucle se termine. Si aucune des commandes de **do** *Liste* n'est exécutée, la commande **while** renvoie une valeur nulle. La commande **until** a le même effet que la commande **while**, à ceci près que le test de fin de boucle n'est pas exécuté.
(*Liste*, page 12-7 **)**	*Liste* spécifie une liste de commandes à exécuter. Le shell exécute le paramètre *Liste* dans un environnement distinct. **Remarque :** Si l'imbrication exige la présence de deux parenthèses contiguës, vous devez insérer un espace entre elles afin de différencier la commande de l'évaluation arithmétique.
(*Liste*, page 12-7 **)**	*Liste* spécifie une liste de commandes à exécuter. Le paramètre *Liste* est simplement exécuté. **Remarque :** Contrairement aux métacaractères `()`, les signes `{}` indiquent les mots réservés (utilisés à des fins particulières, et non des ID déclarés par l'utilisateur). Pour être reconnus, ces mots réservés doivent se trouver au début d'une ligne ou après un point–virgule (`;`).
[[*Expression*]]	Evalue le paramètre *Expression*. Si elle est vraie, la commande renvoie un état de sortie nul.
function *Identificateur*, page 12-5 **{** *Liste*, page 12-7 **;}** ou **function** *Identificateur* **()** **{** *Liste*; **}**	Définit une fonction référencée par *Identificateur*. Le corps de la fonction est constitué par la liste des commandes entre {}. Les parenthèses () constituent deux opérateurs, aussi le panachage de caractères blancs avec *Identificateur*, (et) est-il autorisé, quoique non obligatoire.
time *Pipeline*, page 12-5	Exécute le paramètre *Pipeline*. Le délai écoulé, le temps utilisateur et le temps système sont envoyés sur la sortie standard.

Lancement du shell

Vous pouvez lancer le shell Korn via la commande **ksh**, la commande **psh** (shell POSIX) ou la commande **exec**.

Si vous le lancez par la commande **exec**, et que le premier caractère de l'argument zéro (**$0**) est le tiret (–), le shell est supposé être un shell de connexion. Le shell commence par lire les commandes du fichier **/etc/profile**, puis celles du fichier **.profile** du répertoire courant ou du fichier **$HOME/.profile** (si les fichiers existent). Ensuite, il lit les commandes du fichier (s'il existe) nommé lors de la substitution de paramètre sur la valeur de la variable d'environnement **ENV**.

Si vous spécifiez le paramètre *Fichier* [*Paramètre*] en appelant le shell Korn ou POSIX, celui-ci exécute le fichier script identifié par *Fichier*, avec tous ses paramètres. Le fichier script doit être accessible en lecture. Les éventuels paramètres **setuid** et **setgid** sont ignorés. Le shell lit ensuite les commandes.

Remarque : Ne spécifiez pas de fichier script avec les indicateurs **–c** ou **–s** lors de l'appel du shell Korn ou POSIX.

Pour en savoir plus sur les paramètres positionnels, reportez–vous à "Substitution de paramètres (shell Korn ou POSIX)", page 12-22.

Environnement shell Korn

Toutes les variables (avec leurs valeurs) connues d'une commande lors de son lancement constituent son *environnement* : variables héritées du process parent et variables spécifiées comme paramètres-clés sur la ligne de commande. Les interactions entre le shell et l'environnement sont multiples. Lorsqu'il est lancé, le shell balaye l'environnement et crée un paramètre chaque fois qu'il trouve un nom en attribuant à ce paramètre la valeur correspondante et en le marquant pour exportation. Les commandes exécutées héritent de l'environnement.

Si vous modifiez les valeurs des paramètres shell ou si vous en créez de nouvelles à l'aide des commandes **export** ou **typeset –x**, ces paramètres sont intégrés à l'environnement. Pour une commande exécutée, l'environnement se présente donc comme suit : paires nom-valeur héritées par le shell (valeurs éventuellement modifiées par le shell courant), plus les paires résultant de la commande **export** ou **typeset –x**. La commande exécutée (sous–shell) "voit" les modifications apportées aux variables d'environnement héritées, mais doit exporter ces variables pour que ses shells et process enfants les voient.

Pour modifier l'environnement d'une fonction ou d'une commande simple, il suffit de la préfixer par une ou plusieurs affectations de paramètres, de la forme *Identificateur* = *Valeur*. Les deux expressions suivantes sont donc équivalentes (en ce qui concerne l'exécution de la commande) :

```
TERM=450 arguments commande
(export TERM; TERM=450; arguments commande)
```

Fonctions du shell Korn

Le mot réservé **function** définit les fonctions shell. Le shell lit et enregistre les fonctions en interne, les noms d'alias étant résolus à la lecture de la fonction. Le shell exécute les fonctions de la même manière que les commandes, les arguments étant passés comme paramètres positionnels. Pour en savoir plus sur les paramètres positionnels, reportez–vous à Substitution de paramètres (shell Korn ou POSIX), page 12-22.

Pour plus d'informations sur les paramètres positionnels, reportez–vous à la section Substitution de paramètres (shell Korn ou POSIX), page 12-22.

- valeurs et attributs des variables (sauf utilisation de la commande **typeset** dans la fonction pour déclarer une variable locale) ;
- répertoire de travail ;
- alias, définitions de fonctions et attributs ;
- paramètre spécial $;
- fichiers ouverts.

Les éléments non communs à la fonction et au script appelant (aucun effet secondaire) sont les suivants :

- paramètres positionnels ;
- paramètre spécial # ;
- variables de la liste d'affectation de la fonction appelée ;
- variables déclarées via la commande **typeset** dans la fonction ;
- options ;
- traitements d'interruption (toutefois, les signaux ignorés du script appelant sont également ignorés par la fonction).

Remarque : Dans les versions antérieures du shell Korn, les traitements d'interruption (autres que **EXIT** et **ERR**) étaient communs à la fonction et au script appelant.

Si le traitement d'interruption sur **0** ou **EXIT** est exécuté *dans* le corps d'une fonction, l'action est exécutée une fois la fonction achevée, dans l'environnement à partir duquel a été appelée la fonction. Si le traitement d'interruption est exécuté *à l'extérieur* du corps d'une fonction, l'action est exécutée après sortie du shell Korn. Dans les versions antérieures du shell Korn, aucun traitement d'interruption sur **0** ou **EXIT** défini à l'extérieur du corps de la fonction n'était exécuté après sortie de la fonction.

Erreurs de syntaxe et affectations de variables suivent les règles définies à la section Commandes intégrées du shell Korn ou POSIX, page 12-39.

Pour exécuter une commande composée, il suffit de spécifier son nom comme celui d'une commande simple, les opérandes de la commande faisant temporairement office de paramètres positionnels, pendant la durée de l'exécution. Le paramètre spécial # est modifié pour refléter le nombre d'opérandes. Le paramètre spécial 0 reste inchangé.

La commande spéciale **return** permet de reprendre la main à partir des appels de la fonction. Si des erreurs surviennent à l'intérieur des fonctions, le contrôle est repris par le programme appelant.

Les ID fonctions sont listés via l'option **–f** ou **+f** de la commande spéciale **typeset**. L'option **–f** liste également le texte des fonctions. Les fonctions sont désactivées par l'option **–f** de la commande spéciale **unset**.

Généralement, les fonctions sont désactivées lorsque le shell exécute un script shell. La commande **typeset** assortie de l'option **–xf** permet d'exporter une fonction vers des scripts exécutés, sans appel séparé du shell. Les fonctions qui doivent être définies par appel séparé du shell doivent être spécifiées dans le fichier **ENV** via la commande **typeset** assortie de l'option **–xf.**

Une fonction renvoie un état de sortie nul si sa déclaration a échoué, un état positif sinon. L'état de sortie d'un appel de fonction est celui de la dernière commande exécutée par la fonction.

Historique des commandes (shell Korn ou POSIX)

Le shell Korn ou POSIX enregistre les commandes entrées dans un fichier d'historique, dont le nom est défini par la variable **HISTFILE**. Si cette variable n'est ou ne peut être définie, le fichier **$HOME/.sh_history** est utilisé par défaut. Si le fichier d'historique n'existe pas et que le shell Korn ne dispose pas des droits nécessaires pour le joindre, le shell utilise un fichier temporaire comme fichier d'historique. Le shell accède aux commandes de tous les shells interactifs via le même fichier d'historique, doté des droits d'accès adhoc.

Par défaut, le shell Korn ou POSIX sauvegarde le texte des 128 dernières commandes. La taille du fichier d'historique (variable **HISTSIZE**) n'est pas limitée, mais un fichier trop volumineux risque de ralentir le démarrage du shell Korn.

Substitution de l'historique des commandes

La commande intégrée **fc** permet d'afficher et de modifier tout ou partie du fichier d'historique. Le cas échéant, précisez le numéro ou l'intervalle de numéros (ou encore les premiers caractères) des commandes qui vous intéressent. Vous pouvez spécifier une ou plusieurs commandes.

Si vous ne précisez pas de programme d'édition en argument de la commande **fc**, c'est l'éditeur spécifié par la variable **FCEDIT** qui est utilisé. En cas de non définition de la variable **FCEDIT**, le fichier **/usr/bin/ed** est utilisé. Les commandes modifiées sont affichées et exécutées dès que vous quittez l'éditeur.

Le nom d'éditeur tiret (**–**) permet de sauter la phase d'édition et de réexécuter directement la commande. Vous pouvez, dans ce cas, passer par un paramètre de substitution de la forme *Ancien* **=** *Nouveau* pour modifier la commande avant de l'exécuter. Par exemple, si `r` a pour alias `fc -e -`, tapez `r bad=good c` : la commande la plus récente commençant par la lettre *c* est exécutée et remplace la première occurrence de la chaîne `bad` par la chaîne `good`.

Pour plus d'informations sur l'utilisation de la commande d'historique du shell, reportez–vous à la section Liste des commandes précédemment entrées (commande history), page 4-7 et la commande **fc** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Déclaration de caractères (shell Korn ou POSIX)

Si vous voulez que le shell Korn ou POSIX interprète un caractère comme étant un caractère normal (c'est-à-dire dépourvu de la signification qui lui est normalement associée), vous devez le déclarer. S'il s'agit d'un métacaractère, utilisez un des caractères de déclaration de la liste ci–après.

Un métacaractère est un caractère qui a une signification spéciale pour le shell : si vous omettez de le déclarer, le shell l'interprétera et le mot sera tronqué. Voici les métacaractères du shell Korn ou POSIX (à déclarer, le cas échéant) :

- | (barre verticale)
- & (perluète)
- ; (point–virgule)
- < (inférieur à) et > (supérieur à)
- ((parenthèse gauche) et) (parenthèse droite)
- $ (dollar)
- ` (apostrophe inverse) et ' (apostrophe)
- \ (barre oblique inverse)
- " (guillemets)
- nouvelle ligne
- espace
- tabulation

S'il s'agit d'un métacaractère, utilisez un des caractères de déclaration de la liste ci–après.

\ (barre oblique inverse)	Une barre oblique inverse (\) non déclarée préserve la valeur littérale du caractère suivant (sauf s'il s'agit du caractère nouvelle ligne). S'il s'agit du caractère nouvelle ligne, le shell l'interprète comme une continuation de ligne.
Apostrophes	Mettre des caractères entre apostrophes (' ') préserve leur valeur littérale. Ne placez pas d'apostrophe dans la chaîne encadrée. Une barre oblique inverse ne permet pas d'insérer une apostrophe dans une chaîne de ce type. Pour imbriquer des apostrophes, écrivez, par exemple : 'a'\''b', pour obtenir a'b.
Guillemets	Mettre des caractères entre guillemets (" ") préserve leur valeur littérale – sauf pour les caractères dollar, apostrophe inverse et barre oblique inverse. **$** Le signe dollar conserve sa fonction : développement de paramètres (forme de substitution de commandes) et développement arithmétique. Les caractères entre guillemets se trouvant également entre $(et le symbole) correspondant ne sont pas affectés par les guillemets, mais définissent la commande dont la sortie remplace $ (...) lorsque le mot est développé. A l'intérieur d'une chaîne délimitée par $ { et }, le nombre d'apostrophes ou de guillemets (le cas échéant) doit être pair. Pour déclarer un littéral { ou }, utilisez une barre oblique inverse. ` L'apostrophe inverse conserve sa fonction : introduire l'autre forme de substitution de commande. La partie de la chaîne comprise entre la première apostrophe et la suivante (non précédée d'une barre oblique inverse) définit la commande dont la sortie remplace ` ... ` lorsque le mot est développé. \ La barre oblique inverse conserve sa fonction d'échappement si elle est suivie de l'un des caractères suivants : $, `, ", \ ou par un caractère nouvelle ligne.

Pour insérer un guillemet dans une chaîne entre guillemets, faites–le précéder d'une barre oblique inverse. Avec des guillemets, si une barre oblique inverse est immédiatement suivie d'un caractère spécial, la barre est supprimée et le caractère suivant, interprété littéralement. Si la barre oblique inverse précède un caractère normal, elle demeure à sa place, et le caractère suivant est inchangé. Par exemple :

```
"\$"     ->       $
 "\a"     ->       \a
```

Voici les règles applicables aux métacaractères et aux caractères de déclaration dans le shell Korn ou POSIX :

- La signification du signe dollar, astérisque (`$*`) et signe dollar, arobase (`$@`) est la même lorsqu'ils ne sont pas déclarés, qu'ils soient utilisés comme valeur d'affectation de paramètre ou comme nom de fichier.
- Utilisé comme argument de commande, le signe guillemets doubles, signe dollar, astérisque, guillemets doubles (`"$*"`) équivaut à `"$1 d $2 d..."`, où *d* est le premier caractère du paramètre IFS.
- Le signe guillemets doubles, dollar, arobase, guillemets double (`"$@"`) équivaut à `"$1" "$2" ....`
- Entre apostrophes inverses (`` ` ` ``), la barre oblique inverse déclare les caractères barre oblique inverse (`\`), apostrophe (`'`) et signe dollar (`$`). Si les apostrophes inverses se trouvent entre guillemets (`" "`), la double barre inverse déclare également le caractère guillemets.
- Les substitutions de commandes et de paramètres ne sont pas affectées par les guillemets (`""`).
- Pour annuler la signification spéciale des alias et des mots réservés, déclarez–en chaque caractère. Les noms des fonctions et des commandes intégrées ne peuvent être déclarés.

Mots réservés (shell Korn ou POSIX)

Voici la liste des mots réservés du shell :

```
!          case     do
 done       elif     else
 esac       fi       for
 function   if       in
 select     then     time
 until      while    {
 }          [[       ]]
```

Un mot réservé n'est identifié que lorsqu'il n'est pas déclaré et qu'il constitue :

- le premier mot d'une commande ;
- le premier mot suivant un mot réservé autre que **case**, **for** ou **in** ;
- le troisième mot d'une commande **case** ou **for** (**in** est le seul mot possible).

Alias de commandes (shell Korn ou POSIX)

Le shell Korn, ou POSIX, permet de personnaliser les commandes en créant des alias. La commande **alias** définit comme alias un mot de la forme *Nom=Chaîne*. Lorsque vous utilisez un alias comme premier mot d'une ligne de commande, le shell Korn vérifie s'il traite déjà un alias de même nom. Dans l'affirmative, il ne remplace pas le nom de l'alias. Dans la négative, il remplace le nom de l'alias par sa valeur.

Le premier caractère d'un nom d'alias peut être n'importe quel caractère imprimable à l'exclusion des caractères spéciaux. Les autres caractères sont les mêmes que ceux utilisés pour un identificateur. La chaîne de remplacement peut contenir n'importe quel texte shell correct, métacaractères compris.

Si le dernier caractère d'une valeur alias est un blanc, le shell vérifie aussi le mot qui suit l'alias. Les alias peuvent servir à redéfinir les commandes intégrées spéciales, mais pas les mots réservés. Les définitions d'alias ne sont pas conservées d'un appel de **ksh** à l'autre. Toutefois, si vous spécifiez **alias –x**, l'alias reste en vigueur pour les scripts appelés par leur nom, qui n'appellent pas de scripts distincts. Pour exporter une définition d'alias et en offrir l'accès aux process enfants, spécifiez **alias –x** et la définition de l'alias dans le fichier d'environnement.

Utilisez la commande **alias** pour créer, répertorier et exporter des alias.

Pour supprimer les alias, utilisez la commande **unalias**.

Format de création d'un alias :

```
alias Nom=Chaîne
```

où le paramètre *Name* est le nom de l'alias et *String* sa valeur.

Les alias exportés suivants sont prédéfinis par le shell Korn, mais vous pouvez les inhiber ou les redéfinir. Nous vous conseillons toutefois de ne pas les modifier : vous risqueriez de perturber un tiers, habitué au fonctionnement des alias prédéfinis.

```
autoload='typeset -fu'
false='let 0'
functions='typeset -f'
hash='alias -t'
history='fc -l'
integer='typeset -i'
nohup='nohup '
 r='fc -e -'
true=':'
type='whence -v'
```

Les alias ne sont pas pris en charge par les appels non interactifs du shell Korn (ksh) (dans un script shell ou, comme dans l'exemple suivant avec l'option **–c** associée à **ksh**.

```
ksh -c alias
```

Pour plus d'informations sur la création d'alias, reportez–vous à Création d'un alias (commande shell alias), page 4-10 et à la commande **alias** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

Alias de trace

Les alias sont souvent utilisés pour abréger les noms de chemins absolus. La fonction de création d'alias permet d'associer automatiquement la valeur d'un alias au chemin absolu de la commande correspondante. Ce type d'alias s'appelle alias de *trace*. Ce type d'alias accélère l'exécution des commandes, le shell n'ayant plus à rechercher dans la variable **PATH** le chemin absolu.

La commande **set –h** active le processus de *traçage*, de sorte que chaque fois qu'une commande est référencée, le shell définit un alias de trace. La valeur de cet alias devient indéfinie dès que vous réinitialisez la variable **PATH**.

Ces alias demeurent tracés, jusqu'à ce qu'une référence ultérieure les redéfinisse. Plusieurs alias de trace sont compilés dans le shell.

Substitution de tilde

Après exécution des substitutions d'alias, le shell examine les mots à la recherche de ceux qui commencent par un tilde (~) non déclaré. S'il en trouve, il vérifie le mot, jusqu'à la première barre oblique (/), pour voir s'il correspond à un nom utilisateur du fichier **/etc/passwd**. Si oui, il remplace le caractère ~ et le nom par le répertoire de connexion de l'utilisateur correspondant. Ce processus est appelé *substitution de tilde*.

S'il ne trouve pas de correspondance, le shell ne modifie pas le texte original. Le shell Korn effectue aussi des remplacements dans le cas particulier où le caractère ~ est le seul du mot ou est suivi du signe + (plus) ou du signe – (moins).

~	Remplacé par la variable **HOME**.
~+	Remplacé par la variable **$PWD** (chemin absolu du répertoire précédent).
~–	Remplacé par la variable **$OLDPWD** (chemin absolu du répertoire précédent).

En outre, le shell essaie d'effectuer une substitution de tilde lorsque la valeur d'un paramètre d'affectation de variable commence par un caractère **~**.

Substitution de paramètres (shell Korn ou POSIX)

Le shell Korn, ou shell POSIX, vous permet d'effectuer des substitutions de paramètres.

Cette section traite des points suivants :

Paramètres (shell Korn)

Il existe plusieurs catégories de paramètres :

- Identificateur constitué d'une combinaison des caractères * (astérisque), @ (arobase), #(dièse), ? (point d'interrogation), – (tiret), $ (dollar) et ! (point d'exclamation). Il s'agit des paramètres spéciaux.
- Argument désigné par un chiffre (paramètre positionnel).
- Paramètre désigné par un identificateur, doté d'une valeur et, éventuellement, d'attributs (paramètres/variables nommés).

La commande intégrée spéciale **typeset** affecte valeurs et attributs aux paramètres nommés. Vous trouverez la description des attributs admis par le shell Korn dans la section qui traite de la commande **typeset**. Les paramètres exportés transmettent valeurs et attributs à l'environnement.

Le format d'affectation d'une valeur est le suivant:

```
Nom=Valeur [ Nom=Valeur ] ...
```

Si l'attribut entier **–i** est associé au paramètre *Name*le paramètre *Value* fait l'objet d'une évaluation arithmétique. Pour en savoir plus, reportez–vous à la section Evaluation arithmétique (shell Korn ou POSIX), page 12-30.

Le shell offre une fonction de tableau à une dimension. Chaque élément du tableau est référencé par un indice, désigné par une expression arithmétique entre crochets [ ]. Pour affecter des valeurs à un tableau, utilisez la commande `set –A Nom Valeur...`. Les indices doivent être compris entre 0 et 511. Les tableaux n'ont pas besoin d'être déclarés. Toute référence à un paramètre doté d'un indice valide est autorisée et permet, le cas échéant, la création d'un tableau. Référencer un tableau sans indice équivaut à référencer l'élément zéro.

La commande spéciale **set** sert à affecter des valeurs aux paramètres positionnels.
A l'appel du shell, le paramètre **$0** est positionné à partir de l'argument zéro.
Le caractère $ permet d'introduire des paramètres remplaçables.

Substitution de paramètres

Voici la liste des paramètres remplaçables:

${ *Paramètre* **}**	Le shell lit tous les caractères à partir du signe $ { (signe dollar, accolade gauche) jusqu'au signe } (accolade droite) correspondant faisant partie d'un même mot, même si celui–ci contient des accolades ou des métacaractères. La valeur éventuelle du *paramètre* spécifié est remplacée. Les accolades sont obligatoires lorsque Paramètre est suivi d'une lettre, d'un chiffre ou d'un soulignement qui ne font pas partie du nom ou pour un paramètre nommé indexé. Un paramètre spécifié avec un ou plusieurs chiffres est un *paramètre positionnel*. Tout paramètre positionnel de plus d'un chiffre doit être mis entre accolades. Lorsque la valeur de la variable est un * (astérisque) ou un @ (arobase), chaque paramètre positionnel commençant par **$1**, est remplacé (séparé par un caractère séparateur de zone). Lorsqu'un identificateur de tableau doté d'un indice * (astérisque) ou d'un @ (arobase) est utilisé, la valeur de chacun des éléments (séparé par un caractère séparateur de zone) est remplacé.
${# Paramètre }	Lorsque la valeur de *Paramètre* est un * ou un @, le nombre de paramètres positionnels fait l'objet d'une substitution. Sinon, la longueur spécifiée par *Paramètre* est remplacée.
${# Identificateur [*] }	Le nombre d'éléments du tableau spécifié par le paramètre *Identificateur* est remplacé.
${ Paramètre:– Mot }	Si *Paramètre* est défini et non nul, sa valeur est remplacée ; sinon, la substitution porte sur *Mot*.
${ Paramètre:= Mot }	Si Paramètre est défini et non nul, sa valeur est remplacée ; sinon, la substitution porte sur *Mot*. Les paramètres positionnels ne peuvent être affectés de cette façon.
${ Paramètre:? Mot }	Si Paramètre est défini et non nul, sa valeur est remplacée. Sinon, la valeur de *Mot* s'affiche et vous quittez le shell. Si vous avez omis de définir *Mot*, un message standard s'affiche.
${ Paramètre:+ Mot }	Si *Paramètre* est défini et non nul, sa valeur est remplacée.

${ Paramètre # Trame } | ${ Paramètre ## Trame }	Si la *Trame* shell correspond aux premiers caractères de Paramètre, la valeur de remplacement est celle de Paramètre après suppression de la partie correspondante. Sinon, la valeur de *Paramètre* est remplacée. Dans le premier format, c'est la plus petite correspondance de caractères génériques qui est éliminée. Dans le second format, c'est la plus longue correspondance de caractères génériques qui est éliminée.
${ Paramètre % Trame } | ${ Paramètre %% Trame }	Si la *Trame* shell correspond aux derniers caractères de Paramètre, la valeur de remplacement est celle de Paramètre après suppression de la partie correspondante. Sinon, la valeur de *Paramètre* est remplacée. Dans le premier format, c'est la plus petite correspondance de caractères génériques qui est éliminée, dans le deuxième format, c'est la plus longue. Dans les expressions précédentes, la variable *Mot* n'est évaluée que si elle doit être utilisée comme chaîne remplacée. Dans l'exemple suivant, la commande **pwd** ne sera donc exécutée que si l'indicateur **–d** est nul ou non défini : `echo ${d:-$(pwd)}`

Remarque : Si le signe : (deux points) est omis, le shell vérifie uniquement si Paramètre est défini.

Paramètres spéciaux prédéfinis

Les paramètres automatiquement définis par le shell sont les suivants:

@	Développe les paramètres positionnels, en commençant par **$1**. Les paramètres sont séparés par des espaces. Si vous encadrez **$@** de guillemets("), le shell interprète chaque paramètre positionnel comme une chaîne distincte. En l'absence de paramètre positionnel, le shell développe l'instruction en chaîne nulle non déclarée.
*****	Développe les paramètres positionnels, en commençant par **$1**. Le shell sépare chaque paramètre par le premier caractère du paramètre **IFS** (voir page 12-25. Si vous encadrez **$*** de guillemets("), le shell encadre de guillemets les valeurs du paramètre positionnel. Les valeurs sont séparées par le premier caractère du paramètre **IFS**.
#	Spécifie le nombre (décimal) de paramètres positionnels transmis au shell, nom de la procédure shell elle–même exclu. Le paramètre **$#** génère donc le paramètre positionnel ayant la valeur la plus élevée. Ce paramètre sert principalement à vérifier que le nombre d'arguments requis est présent.
–	Fournit les indicateurs pour le lancement du shell ou de la commande **set**.
?	Valeur de sortie de la dernière commande exécutée (chaîne décimale). La plupart des commandes renvoient 0 pour indiquer qu'elles ont été correctement exécutées. Le shell renvoie la valeur courante **$?** de la variable.

$	Numéro de process du shell. Un numéro de process étant unique, cette chaîne (de 5 chiffres au maximum) est souvent utilisée pour générer des noms uniques pour les fichiers temporaires.
	L'exemple suivant illustre comment créer des fichiers temporaires dans un répertoire réservé à cet effet :
	`temp=$HOME/temp/$$`
	`ls >$temp`
	`.`
	`.`
	`.`
	`rm $temp`

!	Numéro de process de la dernière commande d'arrière–plan invoquée.
0 (zéro)	Développe le nom du shell ou du script shell.

Variables définies par le shell Korn ou POSIX

Les variables définies par le shell sont les suivantes :

_ (souligné)	Indique initialement le nom de chemin du shell ou du script en cours d'exécution tel que transmis à l'environnement. Le dernier argument de la commande précédente lui est ensuite affecté. Ce paramètre n'est pas défini pour les commandes asynchrones. Il sert également à bloquer le nom du fichier **MAIL** correspondant pendant la vérification du courrier.
ERRNO	Valeur définie par la dernière sous–routine non aboutie. Cette valeur, dépendante du système, sert au débogage.
LINENO	Numéro de la ligne courante dans le script ou dans la fonction en cours d'exécution.
OLDPWD	Répertoire de travail précédemment défini par la commande **cd**.
OPTARG	Valeur du dernier argument de l'option traitée par la commande intégrée standard **getopts**.
OPTIND	Valeur du dernier indice de l'option traitée par la commande intégrée standard **getopts**.
PPID	Numéro de process du shell parent.
PWD	Répertoire de travail courant défini par la commande cd.
RANDOM	Génère un nombre entier aléatoire compris entre 0 et 32767. La séquence des nombres aléatoires peut être initialisée en affectant une valeur numérique à la variable **RANDOM**.
REPLY	Défini par l'instruction **select** et la commande intégrée standard **read** en l'absence d'arguments.
SECONDS	Nombre de secondes écoulées depuis le renvoi de l'appel du shell. Lorsque cette variable est renseignée, la valeur renvoyée est la valeur affectée incrémentée du nombre de secondes écoulées depuis l'affectation.

Variables utilisées par le shell Korn ou POSIX

Les variables suivantes sont utilisées par le shell :

CDPATH	Chemin d'accès à la commande **cd** (changement de répertoire).
COLUMNS	Largeur de la fenêtre d'édition, pour les modes d'édition du shell et l'impression des listes **select**.
EDITOR	Si ce paramètre se termine par emacs, gmacs, ou vi, et que la variable **VISUAL** n'est pas définie par la commande set, l'option correspondante est activée.
ENV	Si cette variable est renseignée, une substitution de paramètre est exécutée sur cette valeur pour créer le nom de chemin du script exécuté à l'appel du shell. Cette variable est ignorée pour les shells non interactifs.
FCEDIT	Nom de l'éditeur par défaut pour la commande **fc**.
FPATH	Chemin d'accès aux définitions de fonction. Il est exploré pour les fonctions assorties de l'indicateur **–u** et les commandes introuvables. Si un fichier exécutable est trouvé, il est lu et exécuté dans l'environnement courant.
HISTFILE	Si défini à l'appel du shell, nom du chemin d'accès au fichier enregistrant l'historique des commandes. Le process d'initialisation du fichier **history** peut dépendre des fichiers de lancement du système car certains de ces fichiers peuvent contenir des commandes qui anticipent les paramétrages spécifiés par l'utilisateur pour HISTFILE et HISTSIZE. Par exemple, les commandes de définition de fonction sont enregistrées dans le fichier **history**. Si l'administrateur du système inclue des définitions de fonction dans un fichier de lancement de système qui est appelé avant le fichier **ENV** ou avant que les variables HISTFILE ou HISTSIZE ne soient paramétrées, le fichier **history** est initialisé avant que l'utilisateur ne puisse avoir une influence sur ces caractéristiques.
HISTSIZE	Si défini à l'appel du shell, nombre minimal de commandes précédemment entrées et accessibles par ce shell. La valeur par défaut est de 128 commandes pour les utilisateurs non racine et de 512 commandes pour l'utilisateur racine.
HOME	Répertoire de connexion, qui est le répertoire courant dès que la connexion est établie. Cette variable est initialisée par le programme **login**. La commande **cd** utilise par défaut la valeur du paramètre **$HOME**. Utiliser cette variable plutôt qu'un nom de chemin explicite dans une procédure shell permet d'exécuter la procédure à partir d'un répertoire différent sans modification.
IFS	Séparateurs de zone internes (en général, espace, tabulation et nouvelle ligne), qui servent à séparer les mots d'une commande résultant d'une substitution de paramètre ou de commande, ou issus d'une commande intégrée régulière **read**. Le premier caractère du paramètre **IFS** sépare les arguments pour la substitution`$*`.
LANG	Valeur par défaut des variables **LC_*****.
LC_ALL	Prime sur les valeurs des variables **LANG** et **LC_*****.
LC_COLLATE	Détermine la réaction d'une expression de type intervalle dans le cadre d'une recherche générique.

LC_CTYPE	Définit la classification des caractères, la conversion majuscules/minuscules et autres attributs de caractères.
LC_MESSAGES	Langue d'affichage des messages.
LINES	Longueur des colonnes pour l'affichage des listes de sélection. Les listes de sélection s'affichent verticalement jusqu'au deux tiers environ des lignes spécifiées par le paramètre **LINES**.
MAIL	Chemin d'accès au fichier utilisé par la messagerie pour détecter l'arrivée de courrier. Si vous indiquez le nom d'un fichier courrier et que le paramètre **MAILPATH** n'est pas défini, le shell informe l'utilisateur que du courrier est arrivé dans le fichier spécifié.
MAILCHECK	Fréquence (en secondes) des contrôles effectués par le shell sur d'éventuels changements apportés au délai de modification des fichiers (défini via les variables **MAILPATH** ou **MAIL**). La valeur par défaut est 600 secondes. Lorsque le délai est écoulé, le shell effectue un contrôle avant l'affichage de l'invite suivante.
MAILPATH	Liste de noms de fichiers, séparés par des signes deux points. Si cette variable est renseignée, le shell informe l'utilisateur de toute modification des fichiers spécifiés, survenue au cours de l'intervalle, en secondes, spécifié par **MAILCHECK**. Chaque nom de fichier peut être suivi d'un **?** (point d'interrogation) et d'un message. Le message fera l'objet d'une substitution de paramètre avec la variable **$ _** définie comme nom de fichier modifié. Le message par défaut est `you have mail in $ _`.
NLSPATH	Emplacement des catalogues de messages utilisés pour le traitement de **LC_MESSAGES**.
PATH	Chemin d'accès aux commandes, sous forme de liste de chemins à des répertoires, séparés par des deux–points. Lorsqu'il recherche une commande, le shell explore ces répertoires dans l'ordre spécifié. Le répertoire courant est représenté par une chaîne nulle.
PS1	Chaîne constituant l'invite système principale (**$** par défaut), dont la valeur est développée pour la substitution de paramètres. Le caractère ! (point d'exclamation) dans l'invite principale est remplacé par le numéro de commande.
PS2	Chaîne constituant l'invite système secondaire (**>** (supérieur à), par défaut).
PS3	Chaîne constituant l'invite de sélection dans une boucle **select** (**#?** (dièse, point d'interrogation), par défaut).
PS4	La valeur de cette variable (**+**, par défaut), développée pour la substitution de paramètres, précède chaque ligne des suivis d'exécution.
SHELL	Chemin d'accès au shell, enregistré dans l'environnement.
SHELL PROMPT	En mode interactif, le shell affiche l'invite définie par le paramètre **PS1**, avant de lire une commande. Puis à chaque caractère ligne suivante entré, le shell affiche la deuxième invite (définie par le paramètre **PS2**) pour vous demander de compléter la commande.

TMOUT	**Délai** (en secondes) pendant lequel le shell reste inactif avant de quitter. Si ce délai est positif, le shell quitte en l'absence de commande entrée dans ce délai, après l'affichage de l'invite **PS1**. (Notez que le shell peut être compilé avec l'indication d'une limite maximale pour le délai.) **Remarque :** A l'expiration du délai imparti, une pause de 60-secondes s'écoule encore avant que le shell ne quitte.
VISUAL	Lorsque la valeur de cette variable se termine par emacs, gmacs ou vi, l'option correspondante est activée.

Le shell attribue des valeurs par défaut aux paramètres **PATH**, **PS1**, **PS2**, **MAILCHECK**, **TMOUT** et **IFS**, tandis que les paramètres **HOME**, **SHELL**, **ENV** et **MAIL** *ne sont pas* définis par le shell (bien que le paramètre **HOME** soit défini par la commande **login**).

Substitution de commandes (shell Korn ou POSIX)

Le shell Korn, ou shell POSIX, vous permet d'effectuer des substitutions de commande. Lors d'une substitution de commandes, le shell exécute une commande dans un environnement de sous–shell et remplace cette commande par son résultat.

Pour exécuter une substitution de commande dans le shell Korn ou POSIX, entrez :

```
$(command)
```

ou, avec des apostrophes inverses :

```
`command`
```

> **Remarque :** Bien que **ksh** accepte les apostrophes inverses, les normes XPG4 (X/Open Portability Guide Issue 4) et POSIX les considèrent comme étant obsolètes. Ces dernières prônent l'utilisation de la syntaxe `$(commande)` par les applications portables.

Le shell exécute *commande* dans un environnement sous-shell, et remplace le texte de *commande,* encadré de ( ) ou d'apostrophes inverses, par le résultat standard de la commande, en supprimant les caractères nouvelle ligne à la fin de la substitution.

Dans l'exemple suivant, les signes $( ) (dollar, parenthèses) entourant la commande indiquent la substitution du résultat de la commande **whoami** :

```
echo My name is: $(whoami)
```

La commande suivante a le même effet :

```
echo My name is: `whoami`
```

Dans les deux cas, le résultat (pour l'utilisateur `dee`) est le suivant :

```
My name is: dee
```

Vous pouvez également substituer des expressions arithmétiques en les mettant entre parenthèses. Par exemple, la commande :

```
echo Each hour contains $((60 * 60)) seconds
```

génère :

```
Each hour contains 3600 seconds
```

Le shell Korn ou POSIX supprime tous les caractères nouvelle ligne situés à droite. Par exemple, si votre répertoire courant contient les fichiers `file1`, `file2`, et `file3`, la commande :

```
echo $(ls)
```

supprime les caractères nouvelle ligne et affiche :

```
file1 file2 file3
```

Si vous souhaitez conserver les caractères nouvelle ligne, mettez la commande entre guillemets (" ") :

```
echo "$(ls)"
```

Evaluation arithmétique (shell Korn ou POSIX)

La commande **let** intégrée du shell Korn ou POSIX permet d'effectuer des calculs sur des nombres entiers. Le format des constantes est [*Base*]*Nombre*. Le paramètre *Base* est un nombre décimal compris entre 2 et 36, correspondant à la base arithmétique. Le paramètre *Nombre* est un nombre dans cette base. Si vous ne spécifiez pas le paramètre *Base*, le shell utilise une base de 10.

Les expressions arithmétiques adoptent la même syntaxe, la même priorité et la même associativité qu'en langage de programmation C. Tous les opérateurs entiers, à l'exception du double signe plus (`++`), du double tiret `--`, du point d'interrogation suivi des deux–points `(?:)` et de la virgule (`,`), sont admis. Le tableau suivant contient la liste des opérateurs shell Korn ou POSIX valables, classés par ordre de priorité décroissant :

Opérateur	**Définition**
–	Signe moins (unaire)
!	Négation logique
~	Négation niveau bit
*	Multiplication
/	Division
%	Reste
+	Addition
–	Soustraction
<<, >>	Décalage arithmétique gauche, droite
<=,>=, , ==, !=	Comparaison
&	ET niveau bit
^	OU niveau bit exclus if
|	OU niveau bit
&&	ET logique
||	OU logique
= *=, /=, &= +=, –=, <<=, > >=, &=, ^=, |=	Affectation

Nombre d'opérateurs arithmétiques (`*`, `&`, `<` et `>`, par exemple) ont une signification particulière pour le shell Korn ou POSIX. Ils doivent être placés entre guillemets. Par exemple, pour multiplier la valeur courante de `y` par `5` et réaffecter la nouvelle valeur à `y`, utilisez l'expression :

```
let "y = y * 5"
```

Mis entre guillemets, le caractère `*` perd sa signification spéciale.

Vous pouvez regrouper des opérations dans le cadre de la commande **let**.
Par exemple, l'expression :

```
let "z = q * (z - 10)"
```

multiplie `q` par la valeur de `z` diminuée de 10.

Si l'évaluation ne porte que sur une seule expression, le shell Korn ou POSIX propose un deuxième format de la commande **let**. Le shell traite les commandes entre `(())` comme des expressions déclarées. Ainsi, l'expression :

```
((x = x / 3))
```

équivaut à :

```
let "x = x / 3"
```

Les paramètres nommés sont référencés par nom à l'intérieur d'une expression arithmétique, sans utilisation de la syntaxe de substitution de paramètres. Lorsqu'un paramètre nommé est référencé, sa valeur est évaluée comme une expression arithmétique.

Pour spécifier une représentation interne entière d'un paramètre nommé, utilisez la commande intégrée **typeset** assortie de l'indicateur **–i**. Avec l'indicateur **–i**, l'évaluation arithmétique portera sur la valeur de chaque affectation à un paramètre nommé. Si vous ne spécifiez pas de base arithmétique, la première attribution au paramètre détermine la base arithmétique. Cette base est utilisée lorsque la substitution de paramètres se produit.

Séparation de zones (shell Korn ou POSIX)

Après exécution d'une substitution de commande, le shell Korn balaye les résultats des substitutions, à la recherche des séparateurs de zones définis dans la variable **IFS** (Internal Field Separator). A chaque occurrence d'un séparateur, le shell partage la ligne en arguments distincts. Il conserve les arguments nuls explicites ("" ou ''), mais élimine ceux qui sont implicites (résultant de paramètres non renseignés).

- Si la valeur d'**IFS** est un espace, une tabulation, un caractère nouvelle ligne, ou qu'elle n'est pas définie, toute séquence d'espaces, de tabulations, de caractères nouvelle ligne est ignorée si elle se trouve au début ou à la fin de l'entrée, ou délimite une zone si elle se trouve à l'intérieur de l'entrée. Ainsi, l'entrée suivante génère deux zones, **school** et **days**:

```
<newline><space><tab>school<tab><tab>days<space>
```

- Sinon, et si la valeur d'**IFS** est non nulle, les règles suivantes sont applicables (dans l'ordre). Le terme *espace **IFS*** désigne toute séquence (zéro ou plusieurs occurrences) de caractères définie dans **IFS** (par exemple, si **IFS** contient espace/virgule/tabulation, toute combinaison d'espaces et de tabulations constitue un espace **IFS**).

  1. Un espace ***IFS*** en début ou en fin d'entrée est ignoré.
  2. Chaque occurrence d'un caractère **IFS** autre que l'espace **IFS**, associé à un espace **IFS** adjacent, délimite une zone.
  3. Des espaces **IFS** de longueur non nulle délimitent une zone.

Substitution de noms de fichiers (shell Korn ou POSIX)

Pour procéder à une substitution de noms de fichier, le shell Korn, ou POSIX, balaye les mots de la commande spécifiée par la variable *Mot*, à la recherche de certains caractères. Si un mot contient les caractères `*` (astérisque), `?` (point d'interrogation) ou `[` (crochet gauche) et en l'absence de l'indicateur **–f**, le shell le considère comme une trame. Il les remplace par les noms de fichiers correspondant à la trame, triés selon l'ordre en vigueur dans l'environnement local. Si aucun nom de fichier ne correspond, le mot reste tel quel.

Lorsque le shell utilise une trame pour la substitution de noms de fichiers, les caractères `.` (point) et `/` (barre oblique) doivent correspondre explicitement.

Remarque : Dans d'autres cas de recherche sur la base d'une trame, le shell Korn ne traite pas les caractères distinctement.

Voici les substitutions générées :

***** Correspond à n'importe quelle chaîne, chaîne nulle comprise.

? Correspond à un seul caractère (quelconque).

[... **]** Correspond à n'importe lequel des caractères entre crochets. Une paire de caractères séparés par un trait d'union (`-`) correspond à n'importe quel caractère compris (selon l'ordre local en vigueur) entre ces deux valeurs. Si le premier caractère qui suit le crochet gauche d'ouverture `[` est un point d'exclamation `!`, n'importe quel caractère à l'exclusion de ceux entre crochets correspond. Pour inclure un trait d'union (`-`) dans l'ensemble de caractères considéré, placez–le au début ou à la fin de l'ensemble.

Vous pouvez également utiliser la notation `[: charclass:]` pour rechercher des noms de fichiers dans une certaine plage. Tout caractère de la `classe` est réputé correspondre. La définition du contenu des classes de caractères est indiqué via la catégorie **LC_CTYPE** de la sous–routine **setlocale**. Toutes les classes de caractères spécifiées dans l'environnement local sont reconnues.

Voici quelques unes de ces classes :

- **alnum**
- **alpha**
- **cntrl**
- **digit**
- **graph**
- **lower**
- **print**
- **punct**
- **space**
- **upper**
- **xdigit**

Par exemple, `[[:upper:]]` correspond à toute majuscule.

Le shell Korn prend en charge l'extension de noms de fichiers basée sur le classement des éléments, des symboles ou des classes d'équivalence.

Une *ListeTrames* est une liste d'une ou de plusieurs trames, séparées par une barre verticale (|). Les trames composées sont formées à partir des caractères :

?(*ListeTrames***)**	Correspond éventuellement à l'une des trames spécifiées.
(ListeTrames***)**	Correspond à zéro ou plusieurs occurrences des trames spécifiées
+(*ListeTrames***)**	Correspond à une ou plusieurs occurrences des trames spécifiées
@(*ListeTrames***)**	Correspond exactement à l'une des trames spécifiées
!(*ListeTrames***)**	Correspond à toute trame, à l'exclusion de celles spécifiées

L'utilisation de trames souffre quelques restrictions. Si le premier caractère d'un nom de fichier est un point (.), il ne peut être mis en correspondance qu'avec une trame commençant aussi par un point. Par exemple, * (astérisque) correspond aux fichiers **monfichier** et **votrefichier**, mais pas aux fichiers**.monfichier** et **.votrefichier**. Pour ces fichiers, utilisez, par exemple :

```
.*file
```

Si la recherche n'aboutit pas, c'est la trame elle–même qui est renvoyée en sortie.

Excluez des noms de fichiers et de répertoires les caractères *, ?, [et] : ils risquent de provoquer une boucle infinie lors des recherches sur la base de trames.

Suppression des caractères de déclaration

Les caractères de déclaration barre oblique inverse (\), apostrophe (') et guillemets (") présents dans le mot d'origine, sont supprimés – sauf s'ils sont eux–mêmes déclarés.

Réacheminement des entrées/sorties (shell Korn ou POSIX)

Avant d'exécuter une commande, le shell Korn balaye la ligne d'entrée à la recherche de caractères de réacheminement. Ces derniers lui indiquent de réacheminer les entrées et les sorties. Ils peuvent se trouver n'importe où dans une commande simple, ou la précéder ou la suivre. Ils ne sont pas transmis à la commande appelée.

Le shell exécute les substitutions de commandes et de paramètres avant de prendre en compte les paramètres *Mot* et *Chiffre,* sauf indication spéciale. La substitution de nom de fichier ne se produit que si la trame correspond à un seul fichier et que les blancs ne sont pas interprétés.

< *Mot*	Le fichier *Mot* est l'entrée standard (descripteur de fichier 0).
> *Mot*	Le fichier *Mot* est l'entrée standard (descripteur de fichier 1). Si le fichier n'existe pas, il est créé. Si le fichier existe et que l'option **noclobber** est activée, une erreur est générée ; sinon, le fichier est tronqué à la longueur zéro. **Remarque :** Si l'option **noclobber** est activée sur de multiples shells et qu'ils réacheminent la sortie vers le même fichier, une concurrence critique peut se produire. Cette situation peut aboutir à ce que plusieurs process de shells écrivent dans le fichier. Le shell ne peut ni détecter ni empêcher de telles concurrences critiques.
>| *Mot*	Semblable à la commande **>** *Mot*, sauf que cette instruction de réacheminement prime sur l'option **noclobber**.
> > *Motl*	Le fichier *Mot* est la sortie standard. S'il existe, le shell y ajoute la sortie (après recherche du caractère de fin de fichier). Si le fichier n'existe pas, il est créé.
Mot	Ouvre en lecture et en écriture le fichier *Mot*, en tant qu'entrée standard.
<< [**–**] *Mot*	Lit chaque ligne de l'entrée shell jusqu'à en trouver une ne contenant que la valeur du paramètre *Mot* ou un caractère de fin de fichier. Le shell n'effectue aucune substitution (paramètre, commande ou nom de fichier) sur le fichier spécifié. Le document obtenu, appelé *document here*, page 5-5, devient l'entrée standard. Pour plus d'informations sur les documents *here*, reportez–vous à la section Réacheminement des sorties vers des documents entrée en ligne (here), page 5-5. Si l'un des caractères de *Mot* est déclaré, les caractères du document ne sont pas interprétés.

Le document *here* est traité comme un seul mot, commençant après le caractère nouvelle ligne suivant, et continuant jusqu'à une ligne ne contenant que le délimiteur, sans espaces finaux. Le document here, s'il existe, est alors démarré. Le format est le suivant :

```
[n]<<mot
    document here
 délimiteur
```

Si l'un des caractères de *mot* est déclaré, le délimiteur est créé par suppression des guillemets de *mot*. Les lignes du document *here* ne sont pas développées. Sinon, le délimiteur est *mot* lui–même. Si aucun caractère de *mot* n'est déclaré, toutes les lignes du document here sont développées pour procéder au développement des paramètres, aux substitutions de commandes et au développement arithmétique.

Le shell effectue une substitution de paramètre sur les données réacheminées. Pour empêcher le shell d'interpréter les caractères `\` (barre oblique inverse), `$` (dollar) et `` ` `` (apostrophe), ainsi que le premier caractère du paramètre *Mot*, faites–les précéder du caractère `\`.

Si un signe `-` (moins) est ajouté à la suite de `<<`, le shell élimine toutes les tabulations à gauche du paramètre *Mot* et du document.

<& *Chiffre*	Duplique l'entrée standard à partir du descripteur de fichier spécifié par le paramètre *Chiffre*.
>& *Chiffre*	Duplique la sortie standard à partir du descripteur de fichier spécifié par le paramètre *Chiffre*.
<&–	Ferme l'entrée standard
>&–	Ferme la sortie standard
<&p	Transfère l'entrée du co–processus vers l'entrée standard
>&p	Transfère la sortie du co–process vers la sortie standard

Lorsqu'un chiffre précède l'une des options de réacheminement, le descripteur de fichier indiqué est spécifié par le chiffre (et non par la valeur par défaut 0 ou 1). Dans l'exemple suivant, le shell ouvre le descripteur de fichier 2 pour écriture comme duplicata du descripteur de fichier 1 :

```
... 2>&1
```

L'ordre de spécification des réacheminements est important. Le shell évalue chaque réacheminement selon l'association (*DescripteurFichier*, *Fichier*) au moment de l'évaluation. Dans l'exemple :

```
... 1>Fichier 2>&1
```

le descripteur de fichier 1 est associé au paramètre *Fichier*. Le shell associe le descripteur de fichier 2 au fichier associé au descripteur de fichier 1 (*Fichier*). Si l'on inversait l'ordre des réacheminements, le descripteur de fichier 2 serait associé au terminal (dans l'hypothèse où le descripteur de fichier 1 l'était) et le descripteur de fichier 1, serait associé au paramètre *Fichier*.

Lorsqu'une commande est suivie d'un signe `&` (perluète) et que la fonction de contrôle des travaux n'est pas active, l'entrée standard par défaut de la commande est le fichier vide, **/dev/null**. Sinon, l'environnement d'exécution d'une commande contient les descripteurs de fichier du shell appelant, modifiés par les spécifications d'entrée et de sortie.

Pour en savoir plus sur le réacheminement, voir Réacheminement des entrées/sorties, page 5-1.

Fonction coprocess

Le Korn shell, ou le shell POSIX, vous permet d'exécuter une ou plusieurs commandes comme processus d'arrière–plan. Ces commandes, lancées à partir d'un script shell, sont appelées *coprocess*.

Pour désigner un coprocess, placez un opérateur `|&` (barre verticale, perluète) après une commande. L'entrée et la sortie standard sont toutes deux redirigées vers le script.

Un coprocessus doit satisfaire aux conditions suivantes :

- il doit inclure un caractère de nouvelle ligne à la fin de chaque message,
- il doit envoyer chaque message de sortie vers la sortie standard,
- il doit effacer la sortie standard après chaque message.

L'exemple ci–après illustre la façon dont l'entrée est transmise à un coprocessus et est ensuite renvoyée par ce dernier :

```
echo "Initial process"
./FileB.sh |&
read -p a b c d
echo "Read from coprocess: $a $b $c $d"
print -p "Passed to the coprocess"
read -p a b c d
echo "Passed back from coprocess: $a $b $c $d"
```

```
FileB.sh
   echo "The coprocess is running"
   read a b c d
   echo $a $b $c $d
```

La sortie standard résultante est la suivante :

```
Initial process
Read from coprocess: The coprocess is running
Passed back from coprocess: Passed to the coprocess
```

La commande **print –p** permet d'écrire dans le coprocess. La commande **read –p** permet de lire dans le coprocess.

Réacheminement des entrées/sorties du coprocess

La fonction de réacheminement des E/S permet de réaffecter l'entrée et la sortie standard d'un coprocess à un descripteur de fichier numéroté. Par exemple, la commande :

```
exec 5>&p
```

transfère l'entrée du coprocess vers le descripteur de fichier `5`.

Vous pouvez ensuite, via la syntaxe de réacheminement standard ou par appel d'un autre coprocess, réacheminer la sortie de la commande vers le coprocess. Vous pouvez également démarrer un autre coprocess. La sortie des deux coprocess est connectée au même canal de communication, lisible via la commande **read –p**. Pour mettre fin au coprocess, tapez :

```
read -u5
```

Etat de sortie (shell Korn ou POSIX)

Si le shell détecte une erreur (de syntaxe, par exemple), il renvoie un état de sortie non nul. Sinon, il renvoie l'état de sortie de la dernière commande exécutée. Le shell fait état des erreurs d'exécution en indiquant le nom de la fonction ou de la commande en cause, ainsi que le numéro d'erreur correspondant. Si le numéro de la ligne comportant l'erreur est supérieur à 1, il est également indiqué (entre crochets [ ]), après le nom de la fonction ou de la commande.

Avec un shell non interactif, une erreur rencontrée par une commande, intégrée ou non, génère un message de diagnostic, comme indiqué dans le tableau ci–après.

Erreur	**Commande intégrée**	**Autres fonctions**
Erreur de syntaxe du langage shell	quitte	quitte
Erreur de syntaxe sur une opération (erreur sur une option ou un opérande)	quitte	ne quitte pas
Erreur de réacheminement	quitte	ne quitte pas
Erreur d'affectation de variable	quitte	ne quitte pas
Erreur de développement	quitte	quitte
Commande non trouvée	non applicable	quitte parfois
Script dot non trouvé	quitte	non applicable

Si une erreur indiquée “quitte (parfois)” se produit dans un sous–shell, le sous–shell est (parfois) quitté, avec un état de sortie non nul, mais pas le script contenant le sous–shell.

Dans tous les cas indiqués sur le tableau, un shell interactif émet un message de diagnostic vers la sortie standard, sans quitter.

Commandes intégrées du shell Korn ou POSIX

Les commandes spéciales sont intégrées au shell Korn ou POSIX, et exécutées dans le process shell. Sauf indication contraire, la sortie est dirigée vers le descripteur de fichier 1, avec un état de sortie de zéro (0) si la commande est exempte d'erreurs de syntaxe. Le réacheminement des entrées/sorties est autorisé. Il existe deux types de commandes intégrées : les *commandes intégrées spéciales*, page 12-39 et les *commandes intégrées standard*, page 12-47.

Voici les principales différences entre une commande standard et une commande spéciale :

- Une erreur de syntaxe dans une commande spéciale risque d'entraîner l'arrêt du shell, contrairement à ce qui se passe avec une commande standard. Dans une commande spéciale, une erreur de syntaxe ne générant pas d'arrêt du shell renvoie une valeur non nulle.
- Les affectations de variables spécifiées pour une commande spéciale restent en vigueur après exécution de la commande.
- Les réacheminements d'E/S sont traités après les affectations de paramètres.

En outre, les mots sous forme d'une affectation de paramètre, qui suivent une commande spéciale **export**, **readonly** ou **typeset**, sont développés selon les règles applicables à l'affectation de paramètres. Ainsi, la substitution de tilde est effectuée après le signe =, et la séparation de mot et la substitution de noms de fichiers ne sont pas exécutées.

Pour obtenir un classement alphabétique de ces commandes, reportez–vous à la section Liste des commandes intégrées (shell Korn ou POSIX), page 12-52.

Description des commandes intégrées spéciales

Voici les commandes intégrées spéciales du shell Korn :

`:`, page 12-39	`eval`, page 12-40	`newgrp`	`shift`, page 12-44
`.`, page 12-39	`exec`, page 12-40	`readonly`	`times`, page 12-44
`break`, page 12-40	`exit`, page 12-40	`return`	`trap`, page 12-44
`continue`, page 12-40	`export`, page 12-40	`set`	`typeset`, page 12-45
		`unset`, page 12-46	

: [*Argument*...]	Ne développe que les arguments. Utilisée lorsqu'une commande est requise, comme la condition *then* dans une instruction **if**, mais que cette commande n'exécute aucune action.
. *Fichier* [*Argument*...]	Lit le fichier spécifié et exécute les commandes, dans l'environnement shell courant. Le chemin de recherche, spécifié par la variable **PATH** page 12-25, donne accès au répertoire contenant le fichier. Les éventuels arguments sont traités comme des paramètres positionnels. A défaut, les paramètres positionnels restent inchangés. L'état de sortie est celui de la dernière commande exécutée. Pour plus d'informations sur les paramètres positionnels, reportez–vous à la section Substitution de paramètres (shell Korn ou POSIX), page 12-22. **Remarque :** La commande *.Fichier* [*Argument*...] lit l'intégralité du fichier avant d'exécuter une commande. Aussi les commandes **alias** et **unalias** du fichier ne peuvent–elles s'appliquer aux fonctions définies dans le fichier.

break [*n*]	Quitte, le cas échéant, la boucle **for**, **while**, **until** ou **select**. Si vous spécifiez *n*, la commande sort à la *nième* boucle. La valeur de *n* est un entier supérieur ou égal à 1.
continue [*n*]	Reprend à l'itération suivante de la boucle **for**, **while**, **until** ou **select**. Si vous spécifiez le paramètre *n*, la commande reprend à la *n* ième boucle. La valeur de *n*est un entier supérieur ou égal à 1.
eval [*Argument*...]	Lit les arguments spécifiés comme entrée et exécute la ou les commandes qui en résultent.
exec [*Argument*...]	Exécute, à la place du shell, la commande spécifiée par l'argument (sans créer de nouveau process). Le process courant peut être perturbé par des arguments en entrée ou en sortie. Si vous omettez un argument, la commande **exec** modifie les descripteurs de fichiers comme préconisé par la liste de réacheminement des entrées/sorties. Dans ce cas, tous les numéros de descripteurs de fichiers supérieurs à 2 ouverts par ce mécanisme sont fermés dès qu'un autre programme est appelé.
exit [*n*]	Quitte le shell, l'état de sortie étant soit celui indiqué par le paramètre *n* (entier décimal compris entre 0 et 255), soit celui de la dernière commande exécutée. La sortie du shell est également induite par le caractère fin de fichier, à moins que l'option **ignoreeof** de la commande spéciale **set**, page 12-52 ne soit active.
export –p [*Nom* [**=** *Valeur*]]..	Marque les noms spécifiés, pour exportation automatique vers l'environnement des commande ultérieures. **–p** écrit sur la sortie standard les noms et valeurs de toutes les variables exportées, selon le format : `"export %s= %s\n", <nom> <valeur>`
newgrp [*Groupe*]	Equivalent à la commande **exec/usr/bin/newgrp** [*Groupe*]. **Remarque :** Cette commande ne génère pas de retour.
readonly –p [*Nom* [**=** *Valeur*]] ...	Marque en lecture seule le `Nom` spécifié. Aucune autre affectation ultérieure n'est possible. **–p** écrit sur la sortie standard les noms et valeurs de toutes les variables exportées, selon le format : `"ex port %s= %s\n", <nom> <valeur>`
return [*n*]	Provoque le renvoi d'une fonction shell au script appelant. Le statut de retour est spécifié par le paramètre *n*. Si vous ne spécifiez pas le paramètre *n*, le statut de retour est celui de la commande dernièrement exécutée. Si vous appelez la commande **return** en dehors d'une fonction ou d'un script, elle équivaut à la commande **exit**.

set [**+** | **–abCefhkmnostuvx**] [**+** | **–o** *Option*]... [**+** | **–A** *Nom*] [*Argument*...]	En l'absence de spécification d'options ou d'arguments, la commande **set** écrit les noms et les valeurs des variables shell dans l'ordre défini par l'environnement local. Lorsque des options sont spécifiées, les attributs du shell sont modifiés en conséquence (voir ci–après). **–A** Affectation séquentielle. Annule la définition du paramètre *Nom* et affecte séquentiellement les valeurs à partir de la liste de paramètres *Argument*. Si l'indicateur **+A** est défini, le paramètre *Nom* n'est pas annulé en premier. **–a** Exporte automatiquement tous les autres paramètres définis. **–b** Avertit l'utilisateur, en mode asynchrone, de la fin des travaux en arrière–plan. **–C** Equivalent à `set –o noclobber`. **–e** Exécute le traitement d'interruption **ERR**, le cas échéant, et sort lorsqu'une commande renvoie un état de sortie non nul. Ce mode est désactivé au cours de la lecture des profils. **–f** Désactive la substitution de noms de fichiers. **–h** Désigne chaque commande comme alias tracé, à la première occurrence de la commande. **–k** Place tous les arguments d'affectation de paramètre dans l'environnement de la commande, et pas uniquement les arguments précédant le nom de commande. **–m** Exécute les travaux d'arrière–plan dans un process indépendant et affiche une ligne lorsqu'ils sont terminés. L'état de sortie des travaux d'arrière–plan est indiqué dans un message de fin. Sur les systèmes avec contrôle des travaux, cet indicateur est automatiquement activé pour les shells interactifs. Pour plus d'informations, reportez–vous à la section Contrôle des travaux (shell Korn ou POSIX), page 12-56. **–n** Lit les commandes à la recherche d'erreurs de syntaxe, mais ne les exécute pas. Cet indicateur est ignoré par les shells interactifs.

	–o *Option* Imprime les paramètres de l'option courante et un message d'erreur si vous n'avez pas spécifié d'argument. Vous pouvez paramétrer plusieurs options sur une même ligne de commande **ksh**. Si l'indicateur **+o** est utilisé, l'option spécifiée est désactivée. Lorsque des arguments sont spécifiés, des paramètres positionnels sont paramétrés ou déparamétrés. Les arguments disponibles, spécifiés par la variable *Option*, sont les suivants : **allexport** Identique à l'indicateur **–a**. **bgnice** Exécute tous les travaux d'arrière–plan avec une priorité inférieure. C'est le mode par défaut. **emacs** Entre un éditeur en ligne du type emacs pour l'entrée des commandes. **errexit** Identique à l'indicateur **–e**. **gmacs** Entre un éditeur en ligne de type gmacs pour l'entrée des commandes. **ignoreeof** Ne quitte pas le shell lorsqu'un caractère fin de fichier est rencontré. Pour quitter le shell, vous devez lancer la commande **exit** ou appuyer sur Ctrl–D plus de 11 fois. **mot clé** Identique à l'indicateur **–k**. **Remarque :** Cet indicateur est uniquement réservé pour la rétrocompatibilité avec le shell Bourne. Son utilisation est fortement déconseillée. **markdirs** Ajoute une barre oblique inverse (/) à tous les noms de répertoire générés par la substitution de nom de fichier. **monitor** Identique à l'indicateur **–km**. **noclobber** Empêche la procédure de réacheminement de tronquer les fichiers. Lorsque vous spécifiez cette option, une barre verticale doit être placée à la suite du symbole de réacheminement (>|) pour tronquer un fichier. **noexec** Identique à l'indicateur **–n**. **noglob** Identique à l'indicateur **–f**. **nolog** Empêche les définitions de fonction contenues dans les fichiers **.profile** and **$ENV** d'être sauvegardées dans le fichier d'historique. **nounset** Identique à l'indicateur **–u**. **privileged** Identique à l'indicateur **–p**.

trackall
Identique à l'indicateur **–h**.

verbose
Identique à l'indicateur **–v**.

vi
Appelle le mode insertion d'un éditeur en ligne de type vi pour l'entrée des commandes. Entrer le caractère d'échappement 033 met l'éditeur en mode transfert. Un retour envoie la ligne.

viraw
Traite chaque caractère comme s'il était tapé en mode vi.

xtrace
Identique à l'indicateur **–x**.

–p
Désactive le traitement du fichier **$HOME/.profile** et utilise le fichier **/etc/suid_profile** à la place du fichier **ENV**. Ce mode est activé chaque fois que l'ID utilisateur (UID) ou l'ID groupe (GID) effectif est différent du véritable UID ou du véritable GID. Désactiver cette option positionne l'UID ou le GID effectifs à la valeur de l'UID et du GID réels.
Remarque : Le système ne prend pas en charge pas l'option **–p** car le système d'exploitation ne prend pas en charge les scripts shell **setuid**.

–s
Effectue le tri lexicographique des paramètres positionnels.

–t
Quitte après lecture et exécution d'une seule commande.
Remarque : Cet indicateur est uniquement réservé pour la rétrocompatibilité avec le shell Bourne. Son utilisation est fortement déconseillée.

–u
Traite les paramètres non définis comme des erreurs lors de la substitution.

–v
Affiche les lignes d'entrée au fur et à mesure de leur lecture.

–x
Affiche les commandes et leurs arguments au fur et à mesure de leur exécution.

–
Désactive les indicateurs **–x** et **–v** et met fin à l'examen des arguments des indicateurs.

––
Empêche toute modification des indicateurs. Cette option est très utile pour affecter au paramètre **$1** une valeur commençant par un signe –. Si aucun argument ne suit cet indicateur, les paramètres positionnels ne sont pas définis.

Faire précéder un des indicateurs de la commande **set** d'un signe + plutôt que d'un signe – désactive l'indicateur. Utilisez ces indicateurs à l'appel du shell. Lorsque ' `set +o` ' est appelé sans aucun argument, il affiche les paramètres de l'option en cours dans un format tel que ces paramètres peuvent être réintroduits dans le shell en tant que commandes qui effectuent le même paramétrage de l'option. L'ensemble des indicateurs courant se trouve dans le paramètre **$–**. Sauf si vous spécifiez l'indicateur **–A**, les arguments restants sont des paramètres positionnels et sont attribués, en ordre, à `$1`, `$2`, . . . , etc. Si aucun argument n'est attribué, les noms et valeurs des paramètres nommés sont imprimés sur sortie standard.

shift [*n*]	Renomme les paramètres positionnels, de `$n+1` ... à `$1` .... La valeur par défaut du paramètre *n* est 1. Le paramètre *n* est une expression arithmétique représentée par un nombre non négatif inférieur ou égal au paramètre $#.
times	Affiche les temps utilisateur et système cumulés pour le shell et les process exécutés à partir du shell.
trap [*Commande*] [*Signal*] ...	Exécute la commande spécifiée dès réception par le shell du signal ou des signaux spécifiés. Le paramètre *Commande* est lu une fois à l'activation du traitement d'interruption et une fois à son exécution. Le paramètre *Signal* peut être un numéro ou le nom du signal. Les commandes de traitement d'interruption sont exécutées dans l'ordre des numéros de signal. Toute tentative de définir un traitement d'interruption sur un signal qui a été ignoré en entrée du shell courant est sans effet. Si la commande est un signe **–**, tous les traitements d'interruption sont restaurés à leurs valeurs d'origine. Si vous avez omis la commande et que le premier signal est un signal numérique, la commande **ksh** restaure les valeurs d'origine du ou des paramètres *Signal*. **Remarque :** Si vous avez omis la commande et que le premier signal est un nom symbolique, le signal est interprété comme une commande. Si la valeur de *Signal* est **ERR**, la commande spécifiée est exécutée chaque fois qu'une commande a un état de sortie non nul. Si la valeur de signal est **DEBUG**, la commande spécifiée est exécutée après chaque commande. Si la valeur de *Signal* est **0** ou **EXIT** et que la commande **trap** est définie à l'intérieur d'une fonction, l'action correspondante est exécutée à la fin de la fonction. Si la valeur de *Signal* est **0** ou **EXIT** et que la commande **trap** est définie à l'extérieur de la fonction, l'action correspondante est exécutée à la sortie du shell. La commande **trap** sans arguments affiche la liste des commandes associées à chaque numéro de signal. Pour une liste complète des paramètres *Signal* utilisés dans la commande **trap** sans le préfixe **SIG**, reportez–vous à la description des sous–routines **sigaction, sigvec** ou **signal** dans le manuel *AIX 5L Version 5.3 Technical Reference : Base Operating System and Extensions Volume 2*.

typeset [**+HLRZfiIrtux** [*n*]] [*Nom* [**=** *Valeur*]] ...	Définit les attributs et les valeurs des paramètres shell. Lorsqu'il est appelé à l'intérieur d'une fonction, une nouvelle instance du paramètre *Nom* est créée. Le type et la valeur du paramètre sont restaurés à la fin de la fonction. Vous pouvez spécifier les indicateurs suivants avec la commande **typeset** : **–H** Assure le mappage AIX fichier hôte sur des machines non AIX. **–L** Justifie à gauche et supprime les blancs à gauche du paramètre *Valeur*. Si le paramètre *n* est non nul, il définit la largeur de la zone ; sinon, c'est la valeur de sa première affectation qui le détermine. Une fois affecté, le paramètre est rempli à droite par des blancs ou tronqué, si nécessaire, pour s'ajuster à la zone. Lorsque l'indicateur **–Z** est défini, les zéros à gauche sont supprimés. L'indicateur **–R** est désactivé. **–R** Justifie à droite et remplit avec des blancs à gauche. Si le paramètre *n* est non nul, il définit la largeur de la zone ; sinon, c'est la valeur de sa première affectation qui le détermine. La zone reste remplie par des blancs ou est tronquée à partir de la fin lorsque le paramètre est réaffecté. L'indicateur**L** est désactivé. **–Z** Justifie à droite et remplit avec des zéros à gauche lorsque le premier caractère significatif est un chiffre et que l'indicateur **–L** n'a pas été défini. Si le paramètre *n* est non nul, il définit la largeur de la zone ; sinon, c'est la valeur de sa première affectation qui le détermine. **–f** Indique que les noms se réfèrent à la fonction, et non aux paramètres. Aucune affectation ne peut être effectuée, les seuls indicateurs utilisables sont **–t**, **–u** et **–x**. L'indicateur **–t** active le suivi de la fonction. L'indicateur **–u** déclare non définie la fonction. Lorsque la fonction est référencée, le système recherche sa définition dans la variable **FPATH**. L'indicateur **–x** maintient en vigueur la définition de la fonction dans tous les scripts shell qui ne constituent pas un appel séparé de la commande **ksh**. **–i** Identifie le paramètre comme nombre entier, ce qui accélère le traitement arithmétique. Lorsqu'une valeur non nulle est affectée au paramètre *n*, cette valeur définit la base arithmétique en sortie ; sinon, la première attribution détermine la base en sortie. **–I** Convertit tous les caractères majuscules en minuscules. L'indicateur de conversion des majuscules **–u** est désactivé. **–r** Marque les noms spécifiés par le paramètre *Nom* en lecture seule. Aucune autre affectation ultérieure n'est possible.

	–t Etiquette les paramètres nommés. Les étiquettes, éventuellement définies par l'utilisateur, n'ont aucune signification spéciale pour le shell. **–u** Convertit tous les caractères minuscules en majuscules. L'indicateur **–l** est désactivé. **–x** Marque pour exportation automatique vers l'environnement des commandes ultérieures le nom spécifié par le paramètre *Nom*. Un signe **+** à la place d'un signe **–** désactive les indicateurs de la commande **typeset**. Si vous ne spécifiez pas de paramètres *Nom* mais que vous spécifiez des indicateurs, la liste des noms de paramètres marqués (et éventellement, leurs valeurs) s'affiche. (Si vous ne voulez pas que leurs valeurs soient affichées, utilisez **+** à la place de **–**.) A défaut de noms et d'indicateurs, les noms et attributs de tous les paramètres s'affichent.
unset [**–fv**] *Nom...*	Annule les valeurs et les attributs définis pour les paramètres indiqués par la liste Nom. Si **–v** est spécifié, *Nom* référence un nom de variable : le shell supprime sa définition et le retire de l'environnement. La définition des variables en lecture seule ne peut être supprimée. Supprimer la définition des variables **ERRNO**, **LINENO**, **MAILCHECK**, **OPTARG**, **OPTIND**, **RANDOM**, **SECONDS**, **TMOUT** et souligné (_) leur fait perdre leur signification spéciale, même si elles sont affectées par la suite. Si **–f** est spécifié, *Nom* fait référence à un nom de fonction : le shell supprime la définition de la fonction.

Description des commandes intégrées standard

Voici les commandes intégrées standard du shell Korn :

`alias`, page 12-47	`fg`, page 12-48	`print`	`ulimit`, page 12-50	
`bg`, page 12-47	`getopts`, page 12-48	`pwd`	`umask`, page 12-51	
`cd`, page 12-47	`jobs`, page 12-48	`read`	`unalias`, page 12-51	
`command`, page 12-47	`kill`, page 12-48	`set-groups`	`wait`, page 12-51	
`echo`, page 12-48	`let`, page 12-40	`setsenv`	`test`, page 12-49	`whence`
`fc`, page 12-48				

alias [**–t**] [**–x**] [*AliasName* [**=** *String*]] ...	Crée ou redéfinit les alias, ou écrit les définitions d'alias existantes sur la sortie standard. Pour plus d'informations, reportez–vous à la description de la commande **alias** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
bg [*IDTravail*...]	Place chaque travail spécifié en arrière–plan. Le travail en cours est placé en arrière–plan si vous avez omis le paramètre *IDTravail*. Pour en savoir plus sur le contrôle des travaux, reportez–vous à "Contrôle des travaux (shell Korn ou POSIX), page 12-56. Pour obtenir plus d'informations sur l'exécution des travaux en arrière–plan, reportez–vous à la commande **bg** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
cd [*Argument*]	
cd *Ancien Nouveau*	Cette commande a deux formats. Premier format : le répertoire spécifié par *Argument* devient le répertoire courant. Si *Argument* a la valeur –, vous passez au répertoire précédent. Par défaut *Argument* est la variable shell **HOME**, la variable **PWD** ayant la valeur du répertoire courant. La variable shell **CDPATH** définit le chemin d'accès au répertoire contenant la valeur d'*Argument*. Les autres répertoires sont séparés par un signe :. Le chemin vide (qui est le chemin par défaut) désigne le répertoire courant. Ce répertoire est affiché immédiatement après le signe égal ou entre des deux–points dans la liste des chemins. Si l'argument commence par un /, le chemin n'est pas utilisé. Sinon, tous les répertoires du chemin sont explorés. Second format : le *Nouveau* nom remplace l'*Ancien* nom du répertoire courant, **PWD**, et tente de passer à ce répertoire.
command [**–p**] *Commande* [*Argument*...]	
command [**–v** | **–V**] *Commande*	Cette commande amène le shell à traiter la commande spécifié comme une commande simple, éliminant la fonction shell de consultation. Pour en savoir plus, reportez–vous à la commande **command** dans le manuel *AIX 5L Version 5.3 Commands Reference*.

echo [*Chaîne*...]	Ecrit les chaînes de caractères sur la sortie standard. Reportez–vous à la description de la commande **echo**. L'indicateur **–n** n'est pas pris en charge.
fc [**–r**] [**–e** *Editeur*] [*Premier* [*Dernier*]]	
fc –l [**–n**] [**–r**] [*Premier* [*Dernier*]]	
fc –s [*Ancien* **=** *Nouveau*] [*Premier*]	Affiche le contenu du fichier d'historique des commandes ou appelle un éditeur pour modifier et réexécuter les commandes précédemment définies dans le shell. Pour en savoir plus, reportez–vous à la commande **fc** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
fg [*IDTravail*]	Amène à l'avant–plan le travail spécifié. En l'absence de spécification de travail, la commande amène le travail courant à l'avant–plan. Pour obtenir plus d'informations sur les travaux en cours au premier plan, reportez–vous à la commande **fg** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
getopts *ChaîneOption Nom* [*Argument*...]	Contrôle les options légales du paramètres *Argument.* Pour en savoir plus, reportez–vous à la commande **getopts** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
jobs [**–l** | **–n** | **–p**] [*IDTravail*...]	Affiche l'état des travaux lancés dans l'environnement shell courant. En l'absence de spécification d'un travail spécifique, l'état de tous les travaux actifs est affiché. Si un travail est signalé terminé, le shell supprime l'ID correspondant de la liste des travaux reconnus par l'environnement shell courant. Pour en savoir plus, reportez–vous à la commande **jobs** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
kill [**–s** { *NomSignal* | *NuméroSignal* }] *IDProcess...*	Envoie un signal (par défaut, le signal **SIGTERM**) à un process en cours. En général, ce dernier s'arrête alors. Pour arrêter un processus, indiquez son identificateur (PID) dans la variable *IDProcess*. Le shell fait état des PID de chaque process exécuté en arrière–plan (sauf si vous avez lancé plusieurs process via un pipeline, auquel cas il fait état du numéro de dernier process). Pour déterminer le PID des commandes, vous disposez également de la commande **ps**.
kill [*–NomSignal* | *–NuméroSignal*] *IDProcess...*	
kill –l [*EtatSortie*]	Affiche la liste des noms de signaux. Pour en savoir plus, reportez–vous à la commande **kill** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
let *Expression*...	Evalue les expressions arithmétiques spécifiées. L'état de sortie est 0 si la valeur de la dernière expression est non nulle, 1 sinon. Pour obtenir plus d'informations, reportez–vous à la section Evaluation arithmétique (shell Korn ou POSIX), page 12-30.

print [**–Rnprsu** [*n*]] [*Argument...*]	Imprime la sortie shell. En l'absence d'indicateurs, ou en présence des indicateurs tiret (–) ou double tiret (—), les arguments sont dirigés vers la sortie standard comme décrit par la commande **echo**. Action des indicateurs : **–R** Affiche en mode brut (les conventions d'échappement de la commande **echo** sont ignorées). L'indicateur **–R** affiche tous les arguments et indicateurs autres que **–n**. **–n** Empêche un caractère nouvelle ligne d'être ajoutée à la sortie. **–p** Transmet les arguments au canal du process exécuté via |&, et non à la sortie standard. **–r** Affiche en mode brut. Les conventions d'échappement de la commande **echo** sont ignorées. **–s** Transmet les arguments au fichier d'historique et non à la sortie standard. **–u** Spécifie un numéro de descripteur de fichier *n* de 1 chiffre, dans lequel la sortie est placée. La valeur par défaut est 1.
pwd	Equivalent à **print –r – $PWD**. **Remarque :** La commande interne **pwd** du shell Korn ne prend pas en charge les liens symboliques.
read [**–prsu** [**n**]] [*Nom?Invite*] [*Nom...*]	Lit l'entrée shell. Chaque ligne lue est divisée en zones, séparées par les caractères définis dans la variable **IFS**. Pour en savoir plus, reportez–vous à la commande **read** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
setgroups	Exécute la commande **/usr/bin/setgroups** qui fonctionne comme un shell distinct. Pour en savoir plus, reportez–vous à la commande **setgroups**. Il existe toutefois une différence. Si la commande intégrée **setgroups** appelle un sous–shell, la commande **setgroups** remplace le shell en cours d'exécution. Comme la commande intégrée n'est prise en charge que pour une question de compatibilité, les scripts doivent utiliser le chemin d'accès absolu **/usr/bin/setgroups** plutôt que la commande shell intégrée.
setsenv	Exécute la commande **/usr/bin/setsenv**, qui remplace le shell en cours d'exécution. Pour en savoir plus, reportez–vous à la commande **setsenv**.
test	Identique à [*expression*]. Pour en savoir plus, reportez–vous à la section Expressions conditionnelles pour le shell Korn ou POSIX, page 12-54.

ulimit [**–HSacdfmst**] [*Limite*]	Définit ou affiche les limites des ressources utilisateur, telles que définies dans le fichier **/etc/security/limits**. Ce fichier contient les limites par défaut suivantes :
	`fsize = 2097151`
	`core = 2048`
	`cpu = 3600`
	`data = 131072`
	`rss = 65536stack = 8192`
	Ces valeurs sont utilisées par défaut lorsqu'un nouvel utilisateur intègre le système. Elles sont définies par la commande **mkuser** lorsque l'utilisateur est ajouté au système, ou modifiées par la commande **chuser**.
	Les limites sont d'ordre logiciel ou matériel. Un utilisateur peut, via la commande **ulimit**, modifier une limite logicielle, jusqu'au maximum défini par la limite matérielle. Les limites matérielles ne peuvent être modifiées que par un utilisateur racine.
	–H
	Indique qu'une limite matérielle est définie pour cette ressource. Si vous possédez les droits d'utilisateurs racine, vous pouvez augmenter la limite matérielle. Tout utilisateur dispose des droits pour la réduire.
	–S
	Indique qu'une limite logicielle est définie pour cette ressource. Une limite logicielle peut être augmentée à concurrence de la limite matérielle fixée. Si aucune des options **–H** ou **–S** ne sont spécifiées, la limite s'applique aux deux options.
	–a
	Liste toutes les limites définies.
	–c
	Spécifie le nombre de blocs de 512 octets utilisables pour les vidages mémoire.
	–d
	Spécifie la taille (en ko) de la zone de données.
	–f
	Spécifie le nombre de blocs de 512 octets de fichiers écrits par les process enfants (aucune limite n'est imposée pour la lecture de ces fichiers).
	–m
	Taille de la mémoire physique (en ko).
	–n
	Limite du nombre de descripteurs de fichiers pouvant être ouvert par un process.
	–s
	Spécifie la taille de la zone pile (en ko).
	–t
	Spécifie le temps (en secondes) alloué à chaque process.
	Nombre de systèmes ne sont pas configurés avec toutes les limites : une ressource est limitée lorsque le paramètre *Limite* correspondant est défini. Ce dernier peut être un chiffre propre à la ressource, ou la valeur `unlimited`. La commande **ulimit** accepte les indicateurs suivants :

	Si vous omettez la variable *Limite*, la limite affectée à la ressource courante s'affiche. Il s'agit de la limite logicielle si vous n'avez pas précisé l'indicateur **–H**. Si vous spécifiez plus d'une ressource, le nom et l'unité de la limite sont indiqués avant sa valeur. En l'absence de spécification d'option, l'indicateur **–f** est utilisé par défaut. Si vous changez la valeur et que vous n'avez pas précisé les indicateurs **–H** ou **–S**, definissez les limites matérielles et logicielles. Pour obtenir plus d'informations sur les limites utilisateur et ressources système, reportez–vous aux sous–routines **getrlimit, setrlimit** ou **vlimit** dans le manuel *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 1*.
umask [**–S**] [*Masque*]	Détermine les droits d'accès au fichier. Cette valeur, associée aux droits du process créateur, définit les droits sur un fichier, à sa création (valeur par défaut 022). Si Masque est omis, la commande **umask** affiche, sur la sortie standard, le masque de création du mode fichier de l'environnement shell courant. Pour en savoir plus sur les droits d'accès aux fichiers, reportez–vous à la commande **umask** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
unalias { **–a** | *NomAlias...* }	Supprime la définition des alias spécifiés, ou celle de tous les alias si **–a** est spécifié. Ces définitions sont supprimées de l'environnement shell courant. Pour en savoir plus, reportez–vous à la commande **unalias** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
wait [*IDProcess...*]	Attend le travail spécifié et quitte. En l'absence de spécification d'un travail, la commande attend tous les process enfants actifs. L'état de sortie de la commande est celui du process attendu. Pour en savoir plus, reportez–vous à la commande **wait** dans le manuel *AIX 5L Version 5.3 Commands Reference*.
whence [**–pv**] *Nom...*	Indique l'interprétation à donner à chaque nom spécifié, lorsqu'utilisé comme nom de commande. Utilisé sans autre indicateur, **whence** affiche le chemin d'accès absolu, s'il existe, qui correspond à chaque nom. **–p** Effectue une recherche sur les chemins d'accès au(x) nom(s), même s'il s'agit d'alias, de fonctions ou de mots réservés. **–v** Créée un rapport plus prolixe qui spécifie le type de chaque nom.

Liste des commandes intégrées (shell Korn ou POSIX)

Commandes intégrées spéciales

: (deux points)	Ne développe que les arguments.
. (point)	Lit un fichier et en exécute les commandes.
break, page 12-40	Quitte, le cas échéant, la boucle **for**, **while**, **until** ou **select**.
continue, page 12-40	Reprend à l'itération suivante de la boucle **for**, **while**, **until** ou **select.**
eval, page 12-40	Lit les arguments comme entrée et exécute la ou les commandes qui en résultent.
exec, page 12-40	Exécute, à la place du shell, la commande spécifiée par *Argument,* sans créer de nouveau process.
exit, page 12-40	Quitte le shell dont l'état de sortie est spécifié par le paramètre *n*.
export, page 12-40	Marque les noms spécifiés, pour exportation automatique vers l'environnement des commandes ultérieures.
newgrp, page 12-40	Equivalent à la commande **exec/usr/bin/newgrp** [*Groupe ...*].
readonly, page 12-40	Marque en lecture seule les noms spécifiés.
return, page 12-40	Provoque le renvoi d'une fonction shell au script appelant.
set, page 12-41	Sauf options ou arguments contraires, écrit les noms et les valeurs des variables shell dans l'ordre défini par l'environnement local.
shift, page 12-44	Renomme les paramètres positionnels.
times, page 12-44	Affiche les temps utilisateur et système cumulés pour le shell et les process exécutés à partir du shell.
trap, page 12-44	Exécute une commande donnée lorsque le shell reçoit un ou plusieurs signaux déterminés.
typeset, page 12-45	Définit les attributs et les valeurs des paramètres shell.
unset, page 12-46	Annule les définitions des paramètres et les valeurs spécifiées.

Commandes intégrées standard

alias	Imprime une liste d'alias sur la sortie standard.
bg, page 12-47	Place en arrière–plan les travaux spécifiés.
cd, page 12-47	Passe au répertoire spécifié ou remplace la chaîne courante par la chaîne spécifiée.
echo, page 12-48	Ecrit les chaînes de caractères sur la sortie standard.
fc, page 12-48	Sélectionne un ensemble de commandes à partir de la dernière commande **HISTSIZE** lancée au terminal. Réexécute la commande après substitution ancien–nouveau.
fg, page 12-48	Amène à l'avant–plan le travail spécifié.
getopts, page 12-48	Contrôle les options légales du paramètres *Argument.*
jobs, page 12-48	Affiche des informations sur les travaux spécifiés.
kill	Envoie le signal **TERM** (terminate) aux travaux ou aux process spécifiés.
let, page 12-40	Evalue les expressions arithmétiques spécifiées.
print, page 12-48	Imprime la sortie shell.
pwd, page 12-48	Equivalent à la commande **print –r –$PWD.**
read, page 12-49	Lit l'entrée shell.
ulimit, page 12-50	Définit ou affiche les limites des ressources utilisateur, telles que définies dans le fichier **/etc/security/limits.**
umask, page 12-51	Détermine les droits d'accès au fichier.
unalias	Supprime de la liste des alias les paramètres correspondant à la liste de noms spécifiée.
wait, page 12-51	Attend le travail spécifié et quitte.
whence, page 12-51	Indique l'interprétation à donner à chaque nom spécifié, lorsqu'utilisé comme nom de commande.

Pour plus d'informations, voir Commandes intégrées (shell Korn ou POSIX), page 12-39.

Expressions conditionnelles

Une expression conditionnelle associée à la commande composée **[[** (double crochet) permet de tester les attributs des fichiers et de comparer des chaînes. La substitution de nom de fichier et le fractionnement des mots ne s'appliquent pas aux mots entre **[[** et **]]** (double crochets). Chaque expression est constituée d'une ou de plusieurs expressions binaires ou unaires.

–a *Fichier*	Vrai si le fichier spécifié est un lien symbolique pointant sur un fichier existant.
–b *Fichier*	Vrai si le fichier existe et est un fichier bloc spécial.
–c *Fichier*	Vrai si le fichier existe et est un fichier caractère spécial.
–d *Fichier*	Vrai si le fichier existe et est un répertoire.
–e *Fichier*	Vrai si le fichier existe.
–f *Fichier*	Vrai si le fichier existe et est un fichier ordinaire.
–g *Fichier*	Vrai si le fichier existe et que le bit **setgid** est défini.
–h *Fichier*	Vrai si le fichier existe et est un lien symbolique.
–k *Fichier*	Vrai si le fichier existe et que son bit collant est défini.
–n *Chaîne*	Vrai si la chaîne est de longueur non nulle.
–o *Option*	Vrai si l'option spécifiée est active.
–p *Fichier*	Vrai si le fichier existe et qu'il s'agit d'un fichier FIFO spécial ou d'un tube.
–r *Fichier*	Vrai si le fichier existe et qu'il est lisible par le process courant.
–s *Fichier*	Vrai si le fichier existe et qu'il est de taille supérieure à 0.
–t *DescripteurFichier*	Vrai si le numéro de descripteur de fichier est ouvert et associé à un terminal.
–u *Fichier*	Vrai si le fichier existe et que le bit **setuid** est défini.
–w *Fichier*	Vrai si le fichier existe et que le bit d'écriture est activé. Le fichier peut toutefois ne pas être inscriptible s'il se trouve sur un système de fichiers en lecture seule, même si le test renvoie la valeur vrai.
–x *Fichier*	Vrai si le fichier existe et que l'indicateur **execute** est défini. S'il existe et qu'il s'agit d'un répertoire, le process courant est autorisé à l'explorer.
–z *Chaîne*	Vrai si la chaîne est de longueur nulle.
–L *Fichier*	Vrai si le fichier existe et est un lien symbolique.
–O *Fichier*	Vrai si le fichier existe et qu'il est la propriété de l'ID utilisateur effectif de ce process.
–G *Fichier*	Vrai si le fichier existe et qu'il est la propriété de l'ID groupe effectif de ce process.
–S *Fichier*	Vrai si le fichier existe et qu'il s'agit d'un socket.
Fichier1 **–nt** *Fichier2*	Vrai si *Fichier1* existe et qu'il est plus récent que *Fichier2*.
Fichier1 **–ot** *Fichier2*	Vrai si *Fichier1* existe et qu'il est moins récent que *Fichier2*.
Fichier1 **–ef** *Fichier2*	Vrai si *Fichier1* et *Fichier2* existent et qu'ils font référence au même fichier.
Chaîne1 **=** *Chaîne2*	Vrai si *Chaîne1* est égale à *Chaîne2*.

Chaîne1 **!=** *Chaîne2*	Vrai si *Chaîne1* n'est pas égale à *Chaîne2*.
Chaîne **=** *Trame*	Vrai si la chaîne correspond à la trame.
Chaîne **!=** *Trame*	Vrai si la chaîne ne correspond pas à la trame.
Chaîne1 **<** *Chaîne2*	Vrai si *Chaîne1* précède *Chaîne2*, au sens de la valeur ASCII de leurs caractères.
Chaîne1 **>** *Chaîne2*	Vrai si *Chaîne1* suit *Chaîne2*, au sens de la valeur ASCII de leurs caractères.
Expression1 **–eq** *Expression2*	Vrai si *Expression1* est égale à *Expression2*.
Expression1 **–ne** *Expression2*	Vrai si *Expression1* n'est pas égale à *Expression2*.
Expression1 **–lt** *Expression2*	Vrai si *Expression1* est inférieure à *Expression2*.
Expression1 **–gt** *Expression2*	Vrai si *Expression1* est supérieure à *Expression2*.
Expression1 **–le** *Expression2*	Vrai si *Expression1* est inférieure ou égale à *Expression2*.
Expression1 **–ge** *Expression2*	Vrai si *Expression1* est supérieure ou égale à *Expression2*.

Remarque : Dans les expressions ci–dessus, si la variable *Fichier* est semblable à `/dev/fd/ n`, `n` étant un entier, le test s'applique au fichier ouvert dont le numéro de descripteur est *n*.

Vous pouvez créer une expression composée à partir de ces primitives (petites parties) en vous servant de l'une des expressions suivantes.

(*Expression* **)**	Vrai si l'expression spécifiée est vraie. Utilisée pour grouper des expressions.
! *Expression*	Vrai si l'expression spécifiée est fausse.
Expression1 **&&** *Expression2*	Vrai si *Expression1* et *Expression2* sont simultanément vraies.
Expression1 **||** *Expression2*	Vrai si *Expression1* ou *Expression2* est vraie.

Contrôle des travaux (shell Korn ou POSIX)

Le shell Korn ou POSIX offre une fonction de gestion des séquences de commandes, ou travaux *(jobs)*. Lorsque vous exécutez la commande spéciale **set –m**,12-41 le shell Korn associe un travail à chaque pipeline. Il maintient une table des travaux courants, affichée via la commande **jobs**, et leur affecte des nombres entiers de faible valeur.

Lorsqu'un travail est lancé en arrière–plan avec un caractère `&` (perluète), le shell affiche une ligne du type :

```
[1] 1234
```

indiquant que le travail, en arrière-plan, porte le numéro 1 et est constitué d'un process (de haut niveau) dont l'ID est `1234`.

Lorsqu'un travail est en cours et que vous souhaitez exécuter une autre tâche, appuyez sur les touches Ctrl–Z : un signal **STOP** est envoyé au travail en cours. Le shell indique alors que le travail a été arrêté et affiche une invite. Vous pouvez alors intervenir sur l'état du travail (par exemple, le placer en arrière-plan via la commande **bg**), exécuter d'autres commandes et, éventuellement, le rappeler en avant-plan via la commande **fg**. La séquence Ctrl–Z agit immédiatement et est interprétée comme un arrêt par le shell – qui rejette alors toutes les sorties en attente et les entrées non lues.

Un travail exécuté en arrière-plan s'arrête dès qu'il tente de lire à partir du terminal. Les travaux d'arrière-plan génèrent habituellement une sortie : cette option peut être désactivée via la commande **stty tostop**. Si vous l'activez, les travaux en arrière–plan sont interrompus dès qu'ils tentent de générer une sortie ou de lire une entrée.

Vous pouvez référencer un travail dans le shell Korn de plusieurs façons : par son ID process (PID), mais aussi par :

% *Number*	Référence le travail au numéro donné.
% *Chaîne*	Référence les travaux dont la ligne de commande commence par la variable *Chaîne.*
%? *Chaîne*	Référence les travaux dont la ligne de commande contient la variable *Chaîne*.
%%	Référence le travail courant.
%+	Equivalent à **%%**.
%–	Référence le travail précédent.

Ce shell reconnaît immédiatement les modifications dans l'état du process. Il vous informe dès qu'un travail est bloqué, empêchant toute progression du traitement. Il vous informe avant même d'afficher une invite, afin de ne pas perturber votre travail.

Lorsque le mode Monitor est activé, chaque travail d'arrière-plan achevé déclenche les traitements d'interruption définis pour le **CHLD**.

Si vous tentez de quitter le shell (commande `exit` ou séquence Ctrl–D) alors que des travaux sont arrêtés ou en cours, le système affiche le message : `Travaux arrêtés. Travaux en cours d'exécution.` Lancez la commande **jobs** pour prendre connaissance de ces travaux. Si vous tentez encore de quitter le shell, il met fin aux travaux en cours et arrêtés, sans nouvel avertissement.

Traitement des signaux

Les signaux **SIGINT** et **SIGQUIT** d'une commande appelée sont ignorés si la commande est suivie d'une & (perluète) et que l'option **monitor** est inactive. Sinon, ils prennent les valeurs héritées du shell parent.

Si un signal, pour lequel un traitement d'interruption a été défini, est reçu pendant que le shell attend l'achèvement d'une commande en arrière-plan, le traitement d'interruption n'est exécuté qu'une fois la commande terminée. Ainsi, un traitement d'interruption sur un signal **CHILD** n'est exécuté qu'une fois le travail en arrière-plan terminé.

Edition en ligne (shell Korn ou POSIX)

Pour entrer vos commandes, vous tapez une ligne de commande à partir d'un terminal et la faites suivre d'un caractère de nouvelle ligne (RETOUR ou LIGNE SUIVANTE). Si vous activez une option d'édition en ligne (emacs, gmacs ou vi), vous pouvez modifier cette ligne de commande.

Voici les commandes d'appel des éditeurs :

set –o emacs	Appelle le mode d'édition emacs et ouvre un éditeur en ligne style emacs. Pour en savoir plus, reportez–vous à Mode d'édition emacs, page 12-58.
set –o gmacs	Appelle le mode d'édition emacs et ouvre un éditeur en ligne style gmacs. Pour en savoir plus, reportez–vous à Mode d'édition emacs, page 12-58.
set –o vi	Appelle le mode d'édition vi et ouvre un éditeur en ligne style vi. Pour en savoir plus, reportez–vous à Mode d'édition vi, page 12-61.

Une option d'édition est automatiquement sélectionnée chaque fois que la valeur affectée à **VISUAL** ou à **EDITOR** se termine par l'un de ces noms d'option.

> **Remarque :** Pour accéder aux fonctions d'édition, votre terminal doit accepter RETOUR comme retour chariot sas passage à la ligne. Un espace doit remplacer le caractère courant à l'écran.

Chaque mode d'édition ouvre une fenêtre au niveau de la ligne courante. Sa longueur est définie par la variable **COLUMNS** ; ou, à défaut, égale à 80 espaces caractères. Si la ligne dépasse la longueur de fenêtre diminuée de deux caractères, le système affiche une marque à l'extrémité de la fenêtre. Lorsque le curseur atteint les limites de la fenêtre, celle–ci se recentre sur le curseur. Les marques affichées sont :

>	La ligne se prolonge à droite de la fenêtre.
<	La ligne se prolonge à gauche de la fenêtre.
*****	La ligne se prolonge des deux côtés de la fenêtre.

Les commandes de recherche de chaque mode d'édition donnent accès au fichier d'historique du shell Korn. Seules les chaînes sont mises en correspondance. Si le premier caractère de la chaîne est un ^ (caret), la recherche doit débuter au premier caractère de la ligne.

Mode d'édition emacs

Pour sélectionner le mode d'édition emacs, activez indifféremment l'option **emacs** ou l'option **gmacs** : La seule différence entre ces deux modes est la façon dont ils traitent la commande d'édition Ctrl–T. Pour éditer, déplacez le curseur sur le point à corriger et insérez ou supprimez les caractères ou mots, en fonction de vos besoins. Toutes les commandes d'édition sont des caractères de contrôle ou des séquences d'échappement.

Les commandes d'édition opèrent n'importe où sur la ligne (et pas seulement au début). Sauf indication contraire, n'appuyez ni sur Entrée ni sur la touche ligne suivante (flèche vers le bas) après une commande d'édition.

Ctrl–F	Avance le curseur d'un caractère.
Echap–F	Avance le curseur au mot suivant (un mot est une chaîne composée exclusivement de lettres, chiffres et traits de soulignement).
Ctrl–B	Recule le curseur d'un caractère.

Echap–B	Ramène le curseur au mot précédent.
Ctrl–A	Amène le curseur au début de la ligne.
Ctrl–E	Avance le curseur à la fin de la ligne.
Ctrl–] *c*	Avance le curseur au caractère indiqué.
Echap–Ctrl–] *c*	Avance le curseur au caractère indiqué.
Ctrl–X Ctrl–X	Echange curseur et marque.
ERASE	Efface le caractère précédent. (Caractère d'effacement défini par l'utilisateur, via la commande **stty**, généralement la séquence Ctrl-H).
Ctrl–D	Efface le caractère courant.
Echap–D	Supprime le mot courant.
Echap–RetAr	Efface le mot précédent.
Echap–H	Efface le mot précédent.
Echap–Suppr	Efface le mot précédent. Si la touche Suppression est définie comme touche d'interruption, cette commande ne fonctionne pas.
Ctrl–T	Transpose le caractère courant et le caractère suivant (mode emacs). Transpose les deux caractères précédents (mode gmacs).
Ctrl–C	Met en majuscules le caractère courant.
Echap–C	Met en majuscules le mot courant.
Echap–L	Met en minuscules le mot courant.
Ctrl–K	Efface la zone entre le curseur et la fin de la ligne. Précédé d'un paramètre numérique dont la valeur est inférieure à celle de la position courante du curseur, efface la zone entre la position indiquée par ce paramètre et le curseur. Précédé d'un paramètre numérique dont la valeur est supérieure à la position courante du curseur, efface la zone entre la position indiquée par ce paramètre et le curseur.
Ctrl–W	Efface la zone entre le curseur et la marque.
Echap–P	Repousse dans la pile la zone entre le curseur et la marque.
KILL	Caractère kill défini par l'utilisateur via la commande **stty,** généralement la séquence Ctrl–G ou le caractère `@` (arobase). Tue l'intégralité de la ligne courante. Une succession de deux caractères kill génère un saut de ligne à chaque nouveau caractère kill (utile sur un terminal papier).
Ctrl–Y	Restaure le dernier élément supprimé de la ligne. (Restitue l'élément sur la ligne.)
Ctrl–L	Effectue un saut de ligne et affiche la ligne courante.
Ctrl–@	(Caractère nul) Définit une marque.
Echap–Espace	Définit une marque.
Ctrl–J	(Nouvelle ligne) Exécute la ligne courante.
Ctrl–M	(Retour) Exécute la ligne courante.
EOF	Traite le caractère fin de fichier (généralement, Ctrl-D) comme tel, uniquement si la ligne courante est nulle.
Ctrl–P	Extrait la commande précédente. Il s'agit d'une opération chronologique. Revient à la ligne précédente dans une commande de plusieurs lignes (si le curseur ne se trouve pas sur la première ligne).
Echap–<	Appelle la ligne d'historique la moins récente.

Echap–>	Appelle la ligne d'historique la plus récente.
Ctrl–N	Appelle la commande suivante. Il s'agit d'une opération chronologique.
Ctrl–R *Chaîne*	Inverse l'historique de recherche d'une commande antérieure contenant la *Chaîne*. Si la valeur spécifiée est zéro, la recherche s'effectue vers l'avant. La chaîne se termine par Entrée ou un caractère de nouvelle ligne. Si elle est précédée d'un caret (^), la ligne correspondante doit commencer par *Chaîne*. Si *Chaîne* est omis, la recherche porte sur la ligne de commande contenant le paramètre *Chaîne* le plus récent. Dans ce cas, la valeur 0 inverse la direction de la recherche.
Ctrl–O	Exécute la ligne courante et extrait du fichier d'historique la ligne suivant la ligne courante.
Echap *Chiffres*	Définit un paramètre numérique. Les chiffres sont considérés comme paramètre de la commande suivante. Les commandes admettant un paramètre sont **Ctrl–F**, **Ctrl–B**, **ERASE**, **Ctrl–C**, **Ctrl–D**, **Ctrl–K**, **Ctrl–R**, **Ctrl–P**, **Ctrl–N**, **Ctrl–]**, **Echap–.**, **Echap–Ctrl–]**, **Echap–_**, **Echap–B**, **Echap–C**, **Echap–D**, **Echap–F**, **Echap–H**, **Echap–L** et **Echap–Ctrl–H**.
Echap *Lettre*	(Touche de fonction) Recherche l'alias *_Lettre* dans la liste des alias. Si la recherche aboutit, la valeur de l'alias trouvé est placée dans la file d'entrée. Le paramètre *Lettre* ne doit pas spécifier de fonctions d'échappement (escape).
Echap–[*Lettre*	(Touche de fonction) Recherche l'alias *_Lettre* (double trait de soulignement) dans la liste des alias. Si la recherche aboutit, la valeur de l'alias trouvé est placée dans la file d'entrée. Cette commande permet de programmer des touches de fonction sur plusieurs terminaux.
Echap–.	Insère dans la ligne le dernier mot de la commande précédente. Précédée d'un paramètre numérique, la valeur de ce paramètre détermine quel mot insérer à la place du dernier mot.
Echap–_.	Identique à Echap–. (Echap, tiret, point).
Echap–*	Tente d'effectuer la substitution de nom sur le mot courant. Lorsque le mot ne correspond à aucun fichier ou contient des caractères spéciaux, un astérisque est ajouté.
Echap–Echap	Traitement de nom de fichier. Remplace le mot courant par le préfixe commun le plus long de tous les noms de fichier correspondant au mot courant doté d'un astérisque. S'il n'y a qu'une seule correspondance, une barre oblique (/) est ajoutée s'il s'agit d'un répertoire, ou un espace sinon.
Echap–=.	Liste les fichiers qui correspondent à la trame du mot courant comme si un astérisque (*) avait été ajouté.
Ctrl–U	Multiplie par 4 le paramètre de la commande suivante.
****	Annule l'effet du caractère suivant. Les caractères d'édition et les caractères **ERASE**, **KILL** et **INTERRUPT** (habituellement la touche Suppr) peuvent être entrés sur une ligne de commande ou de recherche sous réserve d'être précédés d'une barre oblique inverse (\). Cette dernière annule les caractéristiques d'édition du caractère suivant (le cas échéant).
Ctrl–V	Affiche la version du shell.
Echap–#	Insère un dièse (#) en début de ligne, puis l'exécute. Un commentaire est alors ajouté dans le fichier d'historique.

mode d’édition vi

L’éditeur vi offre deux modes :

- **Mode saisie.** Lorsque vous entrez une commande, l’éditeur vi est en mode saisie.
- **Mode de contrôle.** Appuyez sur la touche Echap. pour entrer le mode de contrôle.

La plupart des commandes de contrôle admettent un paramètre *Compte* de répétition, en option, avant la commande. En mode vi, sur la plupart des systèmes, le traitement normal est d’abord activé. La commande est réaffichée lorsque :

- le vitesse est supérieure ou égale à 1200 ;
- la commande contient des caractères de contrôle ;
- moins d’une seconde s’est écoulée depuis l’affichage de l’invite.

Le caractère Echap interrompt le traitement normal pour le reste de la commande, ce qui permet de modifier la ligne de commande. Ce schéma offre les avantages du traitement canonique avec écho du mode brut. Si l’option **viraw** est spécifiée, le traitement canonique est désactivé. Ce mode est le mode par défaut pour le systèmes qui n’admettent qu’un seul délimiteur de fin de ligne, et peut être pratique pour certains terminaux.

Les commandes d’édition vi disponibles sont classées par catégories.
Les catégories sont les suivantes :

Commandes d’entrée

Remarque : Par défaut, l’éditeur s’ouvre en mode Entrée.

ERASE	Efface le caractère précédent. (Caractère d’effacement défini par l’utilisateur, via la commande **stty**, généralement la séquence Ctrl–H ou #).
Ctrl–W	Efface le mot précédent séparé par un blanc.
Ctrl–D	Met fin au shell.
Ctrl–V	Annule l’effet du caractère suivant. Les caractères d’édition, comme les caractères ERASE ou KILL, peuvent être saisis dans la ligne de commande ou dans un chaîne de recherche s’ils sont précédés d’une séquence de touche Ctrl–V. Cette dernière annule les caractéristiques d’édition du caractère suivant (le cas échéant).
****	Annule l’effet du caractère ERASE ou KILL suivant.

Commandes de déplacement

Ces commandes servent à déplacer le curseur comme suit :

[*Compte*] **l**	Avance le curseur d'un caractère.
[*Compte*] **w**	Avance le curseur d'un mot (alphanumérique).
[*Compte*] **W**	Avance le curseur au début du mot suivant après un blanc.
[*Compte*] **e**	Avance le curseur à la fin du mot courant.
[*Compte*] **E**	Avance le curseur à la fin du mot séparé par un blanc courant.
[*Compte*] **h**	Recule le curseur d'un caractère.
[*Compte*] **b**	Ramène le curseur au mot précédent.
[*Compte*] **B**	Ramène le curseur au mot séparé par un blanc précédent.
[*Compte*] **l**	Amène le curseur à la colonne spécifiée par *Compte*.
[*Compte*] **f** *c*	Trouve le caractère *c* suivant de la ligne courante.
[*Compte*] **F** *c*	Trouve le caractère *c* précédent de la ligne courante.
[*Compte*] **f** *c*	Equivalent à **f** suivi de **h**.
[*Compte*] **T** *c*	Equivalent à **F** suivi de **l**.
[*Compte*]**;**	Répète, le nombre de fois spécifié par *Compte,* la dernière commande de recherche sur un caractère : **f**, **F**, **t** ou **T**.
[*Compte*]**,**	Inverse, le nombre de fois spécifié par *Compte,* la dernière commande de recherche sur un caractère.
0	Ramène le curseur au début de la ligne.
^	Amène le curseur au premier caractère non blanc de la ligne.
$	Amène le curseur à la fin de la ligne.

Commandes de recherche

Ces commandes donnent accès au fichier d'historique des commandes :

[*Compte*] **k**	Extrait la commande précédente.
[*Compte*] **–**	Equivalent à la commande **k.**
[*Compte*] **j**	Extrait la commande suivante. Entrer la commande **j** donne accès à la commande suivante.
[*Compte*] **+**	Equivalent à la commande **j.**
[*Compte*] **G**	Extrait la commande dont le numéro est spécifié par `Compte`. Il s'agit par défaut de la commande la moins récente de l'historique.
/ *Chaîne*	Recherche en amont dans l'historique d'une commande contenant Chaîne. La chaîne se termine par RETOUR ou un caractère de nouvelle ligne. Si la chaîne spécifiée est précédée d'un ^ (caret), la ligne correspondante doit commencer par *Chaîne*. Si *Chaîne* a une valeur nulle, la chaîne précédente est utilisée.
? *Chaîne*	Identique à **/***Chaîne*, à ceci près que la recherche s'effectue vers l'aval.
n	Recherche l'occurrence suivante de la dernière trame des commandes **/***Chaîne* ou **?**
N	Recherche l'occurrence suivante de la dernière trame des commandes **/***Chaîne* ou **?,** mais dans la direction opposée. Explore l'historique à la recherche de la chaîne entrée via la commande **/***Chaîne* précédente.

Commandes de modification de texte

Ces commandes permettent de modifier la ligne de commande :

a	Ouvre le mode Entrée et insère le texte après le caractère courant.
Un	Ajoute le texte à la fin de la ligne. Equivalent à la commande **$a.**
[*Compte*] **c** *Déplacement*	
c[Compte]Déplacement	Efface la zone entre le caractère courant et la caractère spécifié par *Déplacement*, et ouvre le mode Entrée. Si *Déplacement* a la valeur **c**, l'intégralité de la ligne est supprimée (le mode Entrée est ouvert).
C	Efface la zone entre le caractère courant et la fin de la ligne, et ouvre le mode Entrée. Equivalent à la commande **c$.**
S	Equivalent à la commande **cc.**
D	Efface la zone entre le caractère courant et la fin de la ligne. Equivalent à la commande **d$.**

[*Compte*] **d** *Déplacement* **d** [*Compte*] *Déplacement*	Efface la zone entre le caractère courant et le caractère spécifié par *Déplacement* (ce caractère compris). Si *Déplacement* a la valeur **d**, l'intégralité de la ligne est supprimée.
i	Ouvre le mode Entrée et insère le texte avant le caractère courant.
I	Insère le texte au début de la ligne. Equivalent à la commande **0i.**
[*Compte*] **P**	Place la dernière modification de texte avant le curseur.
[*Compte*] **p**	Place la dernière modification de texte après le curseur.
R	Ouvre le mode Entrée et remplace les caractères affichés à l'écran.
[*Compte*] **r** *c*	Remplace le nombre de caractères spécifiés par *Compte*, à partir de la position du curseur, par les caractères spécifiés par *c*. Cette commande fait ensuite avancer le curseur.
[*Compte*] **x**	Efface le caractère courant.
[*Compte*] **X**	Efface le caractère précédent.
[*Compte*]**.**	Répète la dernière modification de texte.
[*Compte*]~	Convertit les majuscules en minuscules (et réciproquement) le nombre de caractères spécifié par *Compte,* à partir de la position du curseur, et avance le curseur.
[*Compte*] _	Ajoute le mot spécifié par le paramètre *Compte* de la commande précédente et ouvre le mode Entrée. Il s'agit du dernier mot si *Compte* est omis.
*****	Ajoute un astérisque (*) au mot courant et tente une substitution de nom de fichier. Si la recherche n'aboutit pas, le système le signale. Sinon, le mot est remplacé par la trame correspondante et le mode Entrée s'ouvre.
****	Traitement de nom de fichier. Remplace le mot courant par le préfixe commun le plus long de tous les noms de fichier correspondant au mot courant doté d'un astérisque (*). S'il n'y a qu'une occurrence, une barre oblique / est ajoutée s'il s'agit d'un répertoire. Dans le cas contraire, un espace est ajouté.

Commandes diverses

Les commandes d'édition les plus couramment utilisées sont les suivantes :

[*Compte*] **y** *Déplacement*	
y [*Compte*] *Déplacement*	Sélectionne la zone entre le caractère courant et le caractère dont la position est spécifiée par *Déplacement* (ce caractère compris), et place le tout dans le tampon de suppression. Texte et curseur restent inchangés.
Y	Sélectionne la zone entre le caractère courant et la fin de la ligne. Equivalent à la commande **y$.**
u	Annule la dernière commande de modification de texte.
U	Annule toutes les commandes de modification de texte exécutées sur la ligne.
[`Count`] **v**	Renvoie la commande `fc -e ${VISUAL:-${EDITOR:-vi}} Compte` dans le tampon d'entrée. Si *Compte* est omis, la ligne courante est utilisée.
Ctrl–L	Effectue un saut de ligne et affiche la ligne courante. Cette commande ne s'applique qu'en mode Contrôle.
Ctrl–J	(Nouvelle ligne) Exécute la ligne courante, quel que soit le mode en vigueur.
Ctrl–M	(Retour) Exécute la ligne courante, quel que soit le mode en vigueur.
#	Envoie la ligne en la faisant précéder d'un signe dièse (`#`). Utile pour insérer la ligne courante dans l'historique sans l'exécuter. Si la ligne de commande comporte une barre verticale, un point–virgule ou un caractère nouvelle ligne, autant de dièses (`#`) complémentaires sont insérés avant ces symboles. Pour effacer tous les dièses, rappelez la ligne de commande depuis l'historique et entrez un autre dièse (`#`).
=	Liste les noms de fichier correspondant au mot courant comme s'il était doté d'un astérisque.
@ *Lettre*	Recherche l'alias `_Lettre` dans la liste des alias. Si la recherche aboutit, la valeur de l'alias trouvé est placée dans la file d'entrée, en vue de son traitement.

Shell Korn avancé (ksh93)

En plus du système par défaut du shell Korn (**/usr/bin/ksh**), AIX fournit une version avancée disponible comme **/usr/bin/ksh93**. Cette version avancée présente une compatibilité ascendante avec la version par défaut actuelle et comprend quelques fonctions supplémentaires non disponibles dans **/usr/bin/ksh**. Certains scripts peuvent fonctionner différemment sous **ksh93** par rapport au shell par défaut car la gestion des variables diffère quelque peu entre les deux shells.

Remarque : Il existe une version restreinte du shell Korn avancé, appelée **rksh93**.

Les fonctions suivantes sont disponibles dans **/usr/bin/ksh93**:

Améliorations arithmétiques	Vous pouvez utiliser des fonctions libm (fonctions mathématiques que l'on trouve généralement dans le langage de programmation C) dans des expressions arithmétiques, comme `$ value=$((sqrt(9)))`. D'autres opérateurs arithmétiques sont disponibles, y compris les unaires `+`, `++`, `--`, la construction `?:` (par exemple, "`x ? y : z`"), ainsi que l'opérateur `,` (virgule). Les bases arithmétiques sont prises en charge jusqu'à la base 64. L'arithmétique en virgule flottante est également prise en charge. "`typeset -E`" (exponentielle) peut être utilisé pour spécifier le nombre de chiffres significatifs et "`typeset -F`" (flottante) peut être utilisé pour spécifier le nombre de places décimales pour une variable arithmétique. La variable `SECONDS` affiche maintenant au centième de seconde près, plutôt qu'à la seconde près.

Variables composées	Les variables composées sont prises en charge. Une variable composée permet à un utilisateur de spécifier plusieurs valeurs dans un seul nom de variable. Toutes les valeurs sont affectées d'une variable indicielle, séparée de la variable parent par un `.` (point). Par exemple :
	`$ myvar=( x=1 y=2 )`
	`$ print "${myvar.x}"`
	`1`

Affectations composées	Les affectations composées sont prises en charge lors de l'initialisation de batteries de disques, qu'il s'agisse de batteries de disques indexées ou associatives. Les valeurs des affectations sont mises entre parenthèses, comme illustré dans l'exemple ci–après :
	`$ numbers=( zero one two three )`
	`$ print ${numbers[0]} ${numbers[3]}`
	`zero three`

Batteries de disques associatives	Une batterie de disques associative est une batterie de disques dont l'index est une chaîne.
	La commande **typeset** utilisée avec l'indicateur **–A** permet de spécifier les batteries de disques associatives dans ksh93. Par exemple :
	`$ typeset -A teammates`
	`$ teammates=( [john]=smith [mary]=jones )`
	`$ print ${teammates[mary]}`
	`jones`

Références de noms de variables	La commande **typeset** utilisée avec l'indicateur **–n** permet d'affecter un nom de variable comme référence à une autre. Ainsi, modifier la valeur d'une variable modifiera ensuite la valeur de la variable référencée. Par exemple :
	`$ greeting="bonjour"`
	`$ typeset –n welcome=greeting # établit la référence`
	`$ welcome="salut"  # annule la valeur précédente`
	`$ print $greeting`
	`salut`

Expansions des paramètres	Les constructions d'expansions de paramètres suivantes sont disponibles : • `${!varname}` est le nom de la variable. • `${! varname [@]}` nomme les index de la batterie de disques *varname*. • `${ param: offset }` est une sous–chaîne de *param*, commençant à *offset*. • `${param:offset:num}` est une sous–chaîne de *param*, commençant à *offset*, pour *num* nombre de caractères. • `${@: offset }` indique tous les paramètres positionnels commençant à *offset*. • `${@: offset: num }` indique *num* paramètres positionnels commençant à *offset*. • `${ param/pattern/repl}` évalue à *param*, avec la première occurrence de *pattern* remplacée par *repl*. • `${ param//pattern / repl }` évalue à *param*, avec chaque occurrence de *pattern* remplacée par *repl*. • `${ param / #pattern / repl }` si *param* commence par *pattern*, alors *param* est remplacé par *repl*. • `${ param /% pattern / repl }` si *param* se termine par *pattern*, alors *param* est remplacé par *repl*.

Fonctions de protocole	Une fonction de protocole est une fonction associée à une variable spécifique. Cela vous permet de définir et d'appeler une fonction à chaque fois que cette variable est référencée, définie ou retirée. Ces fonctions prennent la forme de *varname.function*, où *varname* est le nom de la variable et *function* est la fonction de protocole. Il existe trois fonctions de protocole prédéfinies : **get**, **set** et **unset**.
	• La fonction ***varname*.get** est appelée à chaque fois que *varname* est référencée. Si la variable spéciale **.sh.value** est définie dans cette fonction, alors la valeur de *varname* prend cette valeur. L'heure en est un simple exemple :
	`$ function time.get`
	` > {`
	` >  .sh.value=$(date +%r)`
	` > }`
	` $ print $time`
	`09:15:58 AM`
	`$ print $time  # cela changera dans quelques secondes`
	`09:16:04 AM`
	• La fonction ***varname*.set** est appelée à chaque fois que *varname* est définie. La valeur qui a été affectée est donnée à la variable **.sh.value**. La valeur affectée à *varname* est la valeur de **.sh.value** lorsque la fonction se termine. Par exemple :
	`$ function adder.set`
	` > {`
	` >  let .sh.value="`
	`$ {.sh.value} + 1"`
	` > }`
	`$ adder=0`
	`$ echo $adder`
	` 1`
	`$ adder=$adder`
	`$ echo $adder`
	` 2`
	• La fonction ***varname*.unset** est exécutée à chaque fois que *varname* est retirée. La variable n'est pas vraiment retirée sauf si elle est retirée dans la fonction elle–même ; sinon elle garde sa valeur.
	Dans toutes les fonctions de protocole, la variable spéciale **.sh.name** est définie au nom de la variable alors que **.sh.subscript** est définie à la valeur de l'indice de variables, le cas échéant.

Environnements de fonctions	Les fonctions déclarées avec le format `function`*`myfunc`* sont exécutées dans un environnement de fonction séparé. Les fonctions déclarées comme *`myfunc`*`()` s'exécutent avec le même environnement que le shell parent.

Variables	Les variables qui commencent par `.sh.` sont réservées par le shell et ont une signification spéciale. Reportez–vous à la description de Fonctions de protocole, page 12-67 ci–dessus pour obtenir une explication de **.sh.name**, **.sh.value** et **.sh.subscript**. **.sh.version**, qui représente la version du shell, est également disponible.
Valeurs de retour des commandes	Les valeurs de retour des commandes sont les suivantes : • Si la commande à exécuter n'est pas trouvée, la valeur de retour est définie à 127. • Si la commande à exécuter est trouvée mais n'est pas exécutable, la valeur de retour est 126. • Si la commande est exécutée mais arrêtée par un signal, la valeur de retour est 256 plus le numéro du signal.
Règles de recherche PATH	Les commandes intégrées spéciales sont d'abord recherchées, puis viennent toutes les fonctions (y compris celles des répertoires FPATH) et les autres commandes intégrées.
Historique du shell	La commande **hist** permet d'afficher et d'éditer l'historique de commandes shell. Dans le shell ksh, la commande **fc** a été utilisée. La commande **fc** est maintenant un alias de **hist**. Les variables sont HISTCMD, qui incrémente une fois pour chaque commande exécutée dans l'historique actuel du shell, et HISTEDIT, qui spécifie quel éditeur utiliser lors de l'utilisation de la commande **hist**.

Commandes intégrées	Voici les commandes intégrées du shell Korn avancé : • La commande **builtin** répertorie toutes les commandes intégrées disponibles. • La commande **printf** fonctionne globalement comme la routine de bibliothèque printf() C. Reportez–vous à la commande **printf**. • La commande **disown** empêche le shell d'envoyer un SIGHUP à la commande spécifiée. • La commande **getconf** fonctionne comme la commande autonome **/usr/bin/getconf**. Reportez–vous à la commande **getconf**. • La commande intégrée **read** dispose des indicateurs suivants : – **read –d { *char* }** permet de spécifier une délimitation de caractères au lieu de la nouvelle ligne par défaut. – **read –t { *seconds* }** permet de spécifier une limite de temps en secondes après laquelle la commande **read** s'arrêtera. Si **read** s'arrête, elle renverra FALSE. • La commande intégrée **exec** dispose des indicateurs suivants : – **exec –a { *name* } { *cmd* }** spécifie que l'argument 0 de *cmd* est remplacé par *name*. – **exec –c { *cmd* }** indique **exec** pour vider l'environnement avant d'exécuter *cmd*. • La commande intégrée **kill** dispose des indicateurs suivants : – **kill –n { *signum* }** est utilisé pour spécifier un numéro de signal à envoyer à un processus, alors que **kill –s { *signame* }** est utilisé pour spécifier un nom de signal. – **kill –l**, sans argument, répertorie tous les noms de signaux mais pas leurs numéros. • La commande intégrée **whence** dispose des indicateurs suivants : – L'indicateur **–a** affiche toutes les concordances, pas seulement la première trouvée. – L'indicateur **–f** indique à **whence** de ne pas rechercher de fonction. • Une séquence de caractère d'échappement a été ajoutée pour être utilisée par les commandes **print** et **echo**. La touche Echap (Echappement) peut être représentée par la séquence `\E`. • Toutes les commandes intégrées standard reconnaissent l'indicateur **–?**, qui affiche la syntaxe de la commande spécifiée.

Shell Bourne

Le shell Bourne est un interpréteur de commandes interactif et un langage de programmation. Il est appelé par la commande **bsh**.

Le shell Bourne peut être exécuté comme shell de connexion ou comme un sous-shell sous le shell de connexion. C'est la commande **login** uniquement qui permet d'exécuter le shell Bourne comme shell de connexion. Pour cela, il faut faire précéder la commande **bsh** d'un tiret (-) : `-bsh`. Sous cette forme, le shell commence par lire et exécuter les commandes du fichier système **/etc/profile** et, le cas échéant, celles de votre fichier **$HOME/.profile**. Le fichier **/etc/profile** définit les variables requises par tous les utilisateurs. Le shell est prêt à lire les commandes à partir de votre entrée standard.

Si le paramètre *Fichier* [*Paramètre*] est spécifié au lancement du shell Bourne, le shell exécute le fichier script identifié par *Fichier*, avec ses paramètres éventuels. Le fichier script doit être accessible en lecture. Les éventuels paramètres **setuid** et **setgid** sont ignorés. Le shell lit ensuite les commandes. Si l'indicateur **–c** ou **–s** est utilisé, ne spécifiez pas ce script.

Environnement du shell Bourne

Toutes les variables (avec leurs valeurs) connues d'une commande lors de son lancement constituent son *environnement* : variables héritées du process parent et variables spécifiées comme paramètres-clés sur la ligne de commande.

Le shell passe à ses process enfants les variables nommées comme arguments de la commande intégrée **export**. Cette commande place les variables nommées à la fois dans l'environnement du shell et dans celui de ses futurs process enfants.

Les mots-clés sont des paires nom-valeur spécifiées généralement avant le nom de procédure dans la ligne de commande (voir également l'indicateur de la commande **set**). Ces variables sont placées dans l'environnement de la procédure appelée.

Voir les exemple ci–après :

- Prenons pour exemple la procédure suivante, qui affiche les valeurs de deux variables (enregistrées dans le fichier de commande **key_command**) :

```
# key_command
echo $a $b
```

Voici les commandes et les sorties générées :

```
Entrée                                  Sortie
a=key1    b=key2  key_command           key1 key2
a=tom     b=john  key_command           tom john
```

Les mots-clés d'une procédure ne sont pas décomptés dans le chiffre enregistré dans $#.

Une procédure a accès aux valeurs de toutes les variables de son environnement. Mais si elle modifie l'une de ses valeurs, ces modifications ne sont pas répercutées dans l'environnement du shell. Elles ne s'appliquent que localement à la procédure concernée. Pour que les modifications s'appliquent à l'environnement transmis aux process enfants, exportez les nouvelles valeurs dans cette procédure.

Voir les exemple ci–après :

- Pour afficher la liste des variables exportables du shell courant, entrez :

```
export
```

- Pour afficher la liste des variables en lecture seule du shell courant, entrez :

```
readonly
```

- Pour afficher la liste des paires variable–valeur de l'environnement courant, entrez :

```
env
```

Pour plus d'informations sur les environnements utilisateur, reportez–vous à la description du fichier /etc/environment, page 11-2.

Shell restreint

Le shell restreint permet de définir des noms de connexion et des environnements d'exécution dont les pouvoirs sont plus contrôlés que ceux du shell Bourne normal. Il est appelé par la commande **Rsh** ou **bsh –r**. Le comportement de ces commandes est identique à celui de la commande **bsh**, à ceci près qu'elles n'autorisent pas à :

- changer de répertoire (commande **cd**),
- définir la valeur des variables **PATH** ou **SHELL,**
- spécifier des chemins ou des noms de commandes comportant une barre oblique (/),
- réacheminer les sorties.

Si le shell restreint détermine qu'une commande est une procédure shell, il appelle le shell Bourne pour l'exécuter. L'utilisateur peut ainsi accéder à toutes les fonctionnalités du shell Bourne par le biais d'un menu de commandes limité. Ce schéma suppose que l'utilisateur n'a pas accès en écriture et en exécution au même répertoire.

Si le paramètre *Fichier* [*Paramètre*] est spécifié au lancement du shell Bourne, le shell exécute le fichier script identifié par *Fichier*, avec ses paramètres éventuels. Le fichier script doit être accessible en lecture. Les éventuels paramètres **setuid** et **setgid** sont ignorés. Le shell lit ensuite les commandes. Si l'indicateur **–c** ou **–s** est utilisé, ne spécifiez pas ce script.

Lancé via la commande **Rsh,** le shell renforce les restrictions après interprétation des fichiers **.profile** et **/etc/environment**. C'est donc à celui qui écrit le fichier **.profile** qu'incombe le contrôle des actions autorisées à l'utilisateur et du répertoire auquel il accède (de préférence pas le répertoire de connexion). Un administrateur peut créer un répertoire de commandes dans le répertoire **/usr/rbin,** utilisable par la commande **Rsh,** en modifiant en conséquence la variable PATH. Lancé via la commande **bsh –r**, le shell applique les restrictions au moment où il interprète les fichiers *.profile*.

Appelé par **–Rsh**, le shell restreint lit le fichier **.profile** de l'utilisateur (**$HOME/.profile**). Il agit comme le shell Bourne normal, à ceci près que toute interruption provoque la sortie immédiate et non un retour au niveau commande.

Shell Korn restreint

Le shell Korn restreint permet de définir des noms de connexion et des environnements d'exécution dont les fonctionnalités sont plus contrôlées que celles du shell Bourne normal. La commande **rksh** ou **ksh –r** ouvre le shell Korn restreint. Le comportement de ces commandes est identique à celui de la commande **ksh**, à ceci près qu'elles n'autorisent pas les actions suivantes :

- Changer le répertoire de travail courant
- Définir la valeur des variables **SHELL**, **ENV** ou **PATH**
- Spécifier le nom de chemin d'une commande contenant une barre oblique (**/**)
- Réacheminer la sortie d'une commande avec > (caret droit), >| (caret droit, symbole tube), (caret gauche, caret droit) ou >> (deux carets droits).

Si le shell Korn restreint détermine qu'une commande est une procédure shell, il appelle le shell Korn pour l'exécuter. L'utilisateur peut ainsi accéder à toutes les fonctionnalités du shell Korn par le biais d'un menu de commandes limité. Ce schéma suppose que l'utilisateur ne dispose pas de droits d'écriture et d'exécution sur le même répertoire.

Si le paramètre **Fichier** [*Paramètre*] est spécifié au lancement du shell Korn, le shell exécute le fichier script identifié par **Fichier**, avec ses paramètres éventuels. Le fichier script doit être accessible en lecture. Les éventuels paramètres **setuid** et **setgid** sont ignorés. Le shell lit ensuite les commandes. Si l'indicateur **–c** ou **–s** est utilisé, ne spécifiez pas ce script.

Lancé via la commande **rksh**, le shell renforce les restrictions après interprétation des fichiers **.profile** et **/etc/environment**. C'est donc à celui qui écrit le fichier **.profile** qu'incombe le contrôle des actions autorisées à l'utilisateur et du répertoire auquel il accède (de préférence pas le répertoire de connexion). Un administrateur peut créer un répertoire de commandes dans le répertoire **/usr/rbin,** utilisable par la commande **rksh** en modifiant en conséquence la variable **PATH**. Lancé via la commande **ksh –r**, le shell applique les restrictions au moment où il interprète les fichiers **.profile.**

Appelé par la commande **rksh**, le shell Korn restreint lit le fichier **.profile** de l'utilisateur (**$HOME/.profile**). Il agit comme le shell Korn normal, à ceci près que toute interruption provoque la sortie immédiate et non un retour au niveau commande.

Commandes du shell Bourne

Lorsque vous lancez une commande dans le shell Bourne, il commence par l'analyser et effectuer toutes les substitutions indiquées. Puis il exécute la commande, à condition que :

- le nom de commande corresponde à une commande spéciale du shell Bourne.

OU

- le nom de commande corresponde au nom d'une fonction définie. Dans ce cas, le shell définit les paramètres positionnels par ceux de la fonction.

Si le nom de commande ne correspond ni à une commande intégrée ni à une fonction définie et que la commande nomme un fichier exécutable qui est un programme (binaire) compilé, le shell (en tant que *parent*) génère dynamiquement un nouveau process (*enfant*) qui exécute immédiatement le programme. Si le fichier est déclaré exécutable mais n'est pas un programme compilé, le shell suppose qu'il s'agit d'une procédure shell. Dans ce cas, le shell génère dynamiquement une autre instance de lui-même (un *sous-shell*) qui lit le fichier et en exécute les commandes. Le shell exécute également les commandes mises entre parenthèses dans le sous–shell. Pour l'utilisateur, l'exécution d'un programme compilé s'effectue exactement de la même manière qu'une procédure shell. Le shell recherche normalement les commandes dans les répertoires du système de fichiers, dans l'ordre suivant :

1. **/usr/bin**
2. **/etc**
3. **/usr/sbin**
4. **/usr/ucb**
5. **$HOME/bin**
6. **/usr/bin/X11**
7. **/sbin**
8. Répertoire courant.

Le shell explore successivement chaque répertoire, passant au suivant lorsqu'il ne parvient pas à trouver la commande.

> **Remarque :** La variable **PATH** détermine l'ordre dans lequel le shell effectue la recherche dans les répertoires. Vous pouvez modifier la séquence donnée des répertoires cible en redéfinissant la variable **PATH**.

Si vous indiquez un chemin d'accès en lançant une commande (par exemple, **/usr/bin/sort**), le shell n'explore que le répertoire correspondant. Si le nom de commande contient une barre oblique (`/`), le shell ne tient pas compte du chemin d'accès.

Vous pouvez spécifier un chemin d'accès complet commençant par le répertoire racine (par exemple **/usr/bin/sort**). Vous pouvez également indiquer un chemin d'accès relatif au répertoire courant. Si vous spécifiez :

```
bin/monfichier
```

le shell cherche le répertoire `bin` dans le répertoire courant, puis, dans ce répertoire, le fichier `monfichier`.

> **Remarque :** Le shell restreint ne traite pas les commandes contenant une barre oblique (/).

Le shell mémorise le chemin d'accès de chaque commande exécutée (pour éviter des recours répétés à la commande **exec**). Si la commande se trouve dans un répertoire relatif (ne commençant pas par /), le shell doit la rechercher à nouveau à chaque changement de répertoire courant. Si vous modifiez la valeur de la variable **PATH** ou que vous exécutez la commande **hash –r**, le shell efface tous les chemins mémorisés.

Ce chapitre traite des points suivants :

- Déclaration de caractères, page 12-74
- Traitement des signaux, page12-74.
- Commandes intégrées (shell Bourne), page 12-76
- Substitution de commandes (shell Bourne), page 12-82

Déclaration de caractères

De nombreux caractères ont une spécification spéciale pour le shell. Vous souhaiterez quelquefois passer outre. Pour cela, vous n'avez qu'à encadrer les caractères concernés par des apostrophes (') et des guillemets (”), ou en les faisant précéder d'une barre oblique inverse (\).

Tous ces caractères, à l'exception de l'apostrophe, sont alors interprétés littéralement. Ainsi, la commande :

```
stuff='echo $? $*; ls * | wc'
```

attribue la chaîne littérale `echo $? $*; ls * | wc` à la variable `stuff`. Le shell n'exécute pas les commandes **echo**, **ls** et **wc** et ne développe ni les variables $? et $*, ni le caractère astérisque (*).

Placés entre guillemets, les caractères $ (dollar), ` (apostrophe inverse) et ” (guillemets) conservent leur signification spéciale, tous les autres caractères étant interprétés littéralement. La substitution des variables et des commandes encadrées par des guillemets a donc lieu. En outre, les guillemets n'affectent pas les commandes dans une substitution de commande faisant partie de la chaîne déclarée, aussi les caractères conservent-ils dans ce cas leur signification spéciale.

Soit la séquence :

```
ls *
  file1 file2 file3
  message="This directory contains `ls * ` "
  echo $message
  This directory contains file1 file2 file3
```

Le caractère spécial (*) figurant dans la chaîne à substituer a été développé.

Pour annuler la signification spéciale des caractères dollar ($), apostrophe inverse (`) et double guillemet (”), placés entre guillemets, faites–les précéder d'une barre oblique inverse (\). Dans une expression non encadrée de guillemets, faire précéder un caractère d'une barre oblique inverse équivaut à le placer entre apostrophes. Par conséquent, une barre oblique inverse précédant immédiatement un caractère nouvelle ligne (c'est–à–dire une barre oblique inverse en fin de ligne) annule le caractère nouvelle ligne, ce qui vous permet de continuer la ligne de commande sur la ligne suivante (physiquement).

Traitement des signaux

Les signaux **INTERRUPT** et **QUIT** sont ignorés lorsque la commande est suivie d'un & (perluète) ; c'est–à–dire exécutée en arrière-plan. Sinon, les valeurs des signaux sont celles transmises par le shell parent, à l'exception du signal VIOLATION DE SEGMENTATION. Pour en savoir plus, reportez–vous à la commande intégrée **trap** du shell Bourne, page 12-80.

Commandes composées (shell Bourne)

Une commande composée est l'un des éléments suivants :

- pipeline (séquence de commandes séparées par le symbole |),
- liste de commandes simples,
- commande commençant par un mot réservé,
- commande commençant par l'opérateur ((parenthèse ouvrante).

Sauf indication contraire, la valeur renvoyée par une commande composée est celle de la dernière commande simple exécutée.

Mots réservés

Les mots réservés ne sont identifiés que lorsqu'ils apparaissent sans guillemets et comme premier mot d'une commande.

```
for                 do              done
   case           esac
   if             then            fi
   elif           else
   while          until
   {              }
   (              )
```

for *Identificateur* [**in** *Mot* . . .] **do** *Liste* **done**	Affecte à *Identificateur* la valeur du (des) paramètre(s) *Mot* (un à la fois) et exécute les commandes spécifiées par *Liste*. Si **in** *Mot*. . . est omis, la commande **for** exécute le contenu du paramètre *Liste* pour chaque paramètre positionnel défini, la commande s'achevant lorsque tous ces paramètres ont été traités.
case *Mot* **in** *Trame* [| *Trame*] . . . **)** *Liste***;;** [*Trame* [| *Trame*] . . . **)** *Liste***;;**]**.. . esac**	Exécute les commandes spécifiées par *Liste*, associées à la première *Trame* correspondant à la valeur de *Mot*. Les conventions de notation sont les mêmes que celles utilisées pour la substitution de noms de fichiers, à ceci près qu'une barre oblique (/), un point en tête (.) ou une séquence barre oblique/point ne doivent pas obligatoirement correspondre explicitement.
if *Liste* **then** *Liste* [**elif** *Liste* **then** *Liste*] . . . [**else** *Liste*] **fi**	Exécute les commandes spécifiées par le paramètre *Liste* qui suit la commande **if**. Si la commande renvoie une valeur nulle, le shell exécute la commande *Liste* par la première commande **then**. Sinon, il exécute le paramètre *Liste* définie par la commande *elif* (si elle est précisée). Si la valeur de sortie est encore nulle, le shell passe aux commandes définies par le **then** suivant. Si la valeur renvoyée est non nulle, le shell exécute les commandes définies par le paramètre *Liste* suivant la commande **else** (si elle est précisée). Si aucune commande **else** *Liste* ou **then** *Liste* n'est exécutée, la commande **if** renvoie une valeur nulle.

while *Liste* **do** *Liste* **done**	Exécute les commandes spécifiées par le paramètre *Liste* qui suit la commande **while**. Si la commande renvoie une valeur nulle, le shell exécute le paramètre *Liste* suivant la commande **do**. Il boucle ensuite sur les listes jusqu'à obtenir une valeur de sortie non nulle. Si aucune des commandes de **do** *Liste* n'est exécutée, la commande **while** renvoie une valeur nulle.
until *Liste* **do** *Liste* **done**	Exécute les commandes spécifiées par le paramètre *Liste* qui suit la commande **until**. Si la commande renvoie une valeur nulle, le shell exécute le paramètre *Liste* suivant la commande **do**. Il boucle ensuite sur les listes jusqu'à obtenir une valeur de sortie nulle. Si aucune des commandes de **do** *Liste* n'est exécutée, la commande **until** renvoie une valeur nulle.
(*Liste* **)**	Exécute, dans un sous-shell, les commandes spécifiées par le paramètre *Liste*.
{ *Liste***; }**	Exécute, dans le shell courant, les commandes spécifiées par le paramètre *Liste*, sans créer de sous-shell.
Nom **() {** *Liste* **}**	Définit une fonction référencée par *Nom*. Le corps de cette fonction est constitué des commandes entre accolades spécifiées par *Liste*.

Commandes intégrées (shell Bourne)

Les commandes spéciales, intégrées au shell Bourne, sont exécutées dans le processus shell. Sauf indication contraire, la sortie est envoyée au descripteur de fichier1 (sortie standard) avec un état de sortie nul lorsque la commande est exempte d'erreur de syntaxe. Le réacheminement des entrées/sorties est autorisé.

Reportez–vous à la section Liste des commandes intégrées (shell Bourne), page 12-91 pour obtenir un classement alphabétique de ces commandes.

Les commandes ci-dessous sont traitées quelque peu différemment des autres commandes intégrées :

```
:  (colon)      exec        shift
 .  (dot)        exit        times
  break        export      trap
  continue     readonly    wait
  eval         return
```

Le shell Bourne traite ces commandes comme suit :

- Les listes d'affectation de paramètres précédant la commande restent en vigueur après l'exécution de la commande.
- Les réacheminements d'E/S sont traités après les affectations de paramètres.
- Toute erreur détectée dans un script shell provoque l'arrêt du script.

Description des commandes spéciales

Voici les commandes intégrées spéciales du shell Bourne :

Commandes intégrées	
:	Renvoie un état de sortie nul.
. *Fichier*	Lit et exécute les commandes de *Fichier* et revient au shell. Ne lance pas de sous-shell. Le shell recherche le fichier dans le répertoire défini par la variable **PATH**.
break [*n*]	Sort, le cas échéant, de la boucle **for**, **while** ou **until**. Si *n* est spécifié, la commande **break** sort à la nième boucle.
continue [*n*]	Reprend à l'itération suivante de la boucle **for**, **while** ou **until**. Si vous spécifiez *n*, la commande reprend à la *n*ième boucle.
cd *Répertoire*]	Passe au répertoire défini par *Répertoire*. En l'absence de spécification de *Répertoire*, c'est la valeur de la variable shell **HOME** qui est utilisée. La variable shell **CDPATH** définit le chemin d'accès pour *Répertoire*. **CDPATH** est constitué d'une liste de répertoires séparés par des deux–points. Un chemin d'accès vide (qui est le chemin par défaut) désigne le répertoire courant. Ce répertoire est affiché immédiatement après le signe égal ou entre les deux-points dans la liste des chemins. Si *Répertoire* commence par une barre oblique (/), le shell ne tient pas compte du chemin d'accès. Sinon, tous les répertoires de la variable shell **CDPATH** sont explorés. **Remarque :** Le shell restreint ne traite pas la commande **cd**.
echo *Chaîne* . . .]	Ecrit les chaînes de caractères sur la sortie standard. Pour en savoir plus, reportez–vous à la commande **echo**. L'indicateur **–n**, page 12-79 n'est pas pris en charge.
eval [*Argument* . . .]	Lit les arguments comme entrée du shell et exécute le(s) commande(s) résultante(s).
exec [*Argument* . . .]	Exécute, à la place du shell, la commande spécifiée par *Argument*, sans créer de nouveau process. Des arguments d'entrée/sortie peuvent apparaître et, si d'autres arguments n'apparaissent pas, engendrer des modifications de l'entrée ou de la sortie shell. Ceci est fortement déconseillé s'il s'agit de votre shell de connexion.
exit [*n*]	Quitte le shell dont l'état de sortie est spécifié par le paramètre *n*. Si n est omis, la valeur de sortie est celle de la dernière commande exécutée (CtrlD provoque également la sortie du shell). La valeur du paramètre *n* est un entier compris entre 0 et 255.
export [*Nom* . . .]	Marque les noms spécifiés, pour exportation automatique vers l'environnement des commande ultérieures. Si *Nom* est omis, la commande **export** affiche la liste des noms exportés dans ce shell. Vous ne pouvez exporter des noms de fonctions.

hash [**–r**][*Commande* . . .]	Recherche et mémorise l'emplacement du chemin d'accès à chaque *Commande* spécifiée. A l'inverse, l'indicateur **–r** efface tous les chemins mémorisés. En l'absence de cet indicateur ou de commandes, le shell affiche des informations sur les commandes mémorisées, au format : `Nombre Coût Commande`
	`Nombre` indique le nombre de fois qu'une commande a été exécutée par le processus shell. `Coût` est une mesure du travail requis pour rechercher une commande dans un chemin. `Commande` affiche les noms des chemins correspondant à chaque commande spécifiée. Dans certains cas, l'emplacement mémorisé doit être recalculé – s'il s'agit d'un chemin relatif et que vous changez de répertoire courant, par exemple. Les commandes concernées sont indiquées par un astérisque (*), affiché en regard de `Nombre`. `Coût` est incrémenté en conséquence.
pwd	Affiche le répertoire courant. Reportez–vous à la description de la commande **pwd**.
read [*Nom* . . .]	Lit une ligne de l'entrée standard. Affecte le premier mot de la ligne au premier paramètre *Nom*, le deuxième mot au deuxième paramètre, etc., les mots résiduels étant affectés au dernier paramètre *Nom*. Cette commande renvoie une valeur nulle, sauf si elle détecte un caractère fin de fichier.
readonly [*Nom* . . .]	Marque en lecture seule le *Nom* spécifié. La valeur de ce nom ne peut être redéfinie. Si *Nom* est omis, la commande **readonly** affiche la liste des noms en lecture seule.
return [*n*]	Provoque la fin d'une fonction, avec la valeur de retour *n*. Si *n* est omis, la fonction renvoie l'état de la dernière commande exécutée de cette fonction. Cette commande n'est valide que lorsqu'exécutée sous la fonction shell.

set [*Flag* [*Argument*] . . .]	Définit un ou plusieurs des indicateurs suivants : **–a** Marque pour l'export toutes les variables auxquelles une affectation est effectuée. Si l'affectation précède un nom de commande, l'attribut export n'est efficace que pour l'environnement d'exécution de cette commande, sauf lorsque l'affectation précède une des commandes intégrées spéciales. Dans ce cas, l'attribut export reste après l'arrêt de la commande intégrée. Si l'affectation ne précède pas un nom de commande ou si l'affectation est un résultat de l'opération des commandes **getopts** ou **read**, l'attribut export reste jusqu'à ce que la variable soit retirée. **–e** Provoque la sortie immédiate d'une commande si toutes les conditions suivantes sont réunies : . elle renvoie une valeur supérieure à zéro, . elle n'est pas un élément de la liste d'une commande **while**, **until** ou **if**, . elle n'est pas en cours de test via des listes AND ou OR, . elle n'est pas un pipeline précédé du mot réservé ! (point d'exclamation). **–f** Désactive la substitution de noms de fichiers. **–h** Localise et mémorise les commandes appelées par les fonctions au moment de leur définition. (Normalement, ces commandes sont localisées au moment de l'exécution de la fonction ; reportez–vous à la description de la commande **hash**, page 12-78). **–k** Place tous les mots–clés dans l'environnement d'une commande (et pas seulement ceux qui précèdent le nom de commande). **–n** Lit les commandes sans les exécuter. Pour contrôler les erreurs de syntaxe de script shell, utilisez l'indicateur **–n**. **–t** Quitte après lecture et exécution d'une seule commande. **–u** Traite toute variable non définie comme une erreur et sort immédiatement après avoir opéré la substitution de variable. Un shell interactif ne permet pas de sortie. **–v** Affiche les lignes d'entrée au fur et à mesure de leur lecture. **–x** Affiche les commandes et leurs arguments avant leur exécution. **––** Ne modifie pas les indicateurs. Utile pour définir le paramètre positionnel **$1** par une chaîne commençant par un tiret (–).

	Un signe + (plus) à la place d'un signe – (moins) désactive les indicateurs. Vous pouvez aussi spécifier ces indicateurs sur la ligne de commande du shell. La variable spéciale **$–** contient l'ensemble des indicateurs courants. Tout *Argument* se rapportant à la commande **set** devient un paramètre positionnel et est attribué, en ordre, à `$1`, `$2`, `...`, etc. Si vous ne spécifiez pas d' *indicateur* ou d' *argument*, la commande **set** affiche tous les noms et valeurs des variables shell actuelles.
shift [*n*]	Décale les arguments de la ligne de commande vers la gauche ; autrement dit, réaffecte les paramètres positionnels en rejetant la valeur courante de **$1** et en réaffectant la valeur de **$2** à **$1**, de **$3** à **$2**, etc. S'il y a plus de 9 arguments, le 10ème est affecté à **$9** et s'ils en restent d'autres, ils ne sont pas affectés (jusqu'au prochain **shift**). S'il y a 9 arguments (ou moins), la commande **shift** annule l'affectation du paramètre positionnel doté du numéro le plus élevé. Le paramètre positionnel **$0** n'est jamais décalé. La commande **shift** *n* est une notation abrégée qui permet de spécifier *n* décalages consécutifs. La valeur par défaut de *n* est 1.
test *Expression* | [*Expression*]	Evalue des expressions conditionnelles. Reportez–vous à la description de la commande **test**. L'indicateur **–h** n'est pas accepté par la commande de test intégrée dans **bsh**.
times	Affiche les temps cumulés utilisateur et système, pour le shell et les process exécutés à partir de celui–ci.
trap [*Commande*] [*n*] . . .	Exécute la commande spécifiée par le paramètre correspondant lorsque le shell reçoit le ou les signaux déterminés par *n*. Les commandes **trap** sont exécutées dans l'ordre des numéros des signaux. Toute tentative de définir un traitement d'interruption sur un signal qui a été ignoré en entrée du shell courant est sans effet. **Remarque :** Le shell balaye le paramètre *Commande* une première fois lorsque le traitement d'interruption est défini, puis de nouveau à l'exécution du traitement. Si vous ne spécifiez pas de commande, tous les traitements d'interruption spécifiés par *n* sont restaurés à leurs valeurs courantes. Si la chaîne spécifiée est vide, le shell et les commandes appelées ignorent le signal. Si *n* vaut 0 (zéro), la commande est exécutée lorsque vous quittez le shell. Si vous n'avez spécifié ni commande ni signal, la commande **trap** affiche la liste des commandes associées à un numéro de signal.
type [*Nom* . . .]	Indique l'interprétation à donner à chaque *Nom* spécifié, lorsqu'utilisé comme nom de commande.

ulimit [**–HS**] [– **c** | **–d** | **–f** | **–m** | **–s** | **–t**] [*limite*]	Affiche ou ajuste les ressources shell affectées. Les paramètres des ressources shell peuvent être affichés individuellement ou en groupe. Par défaut, les ressources sont affichées avec leur limite logicielle, ou leur borne inférieure, en tant que groupe. L'affectation des ressources shell dépend de l'ID utilisateur effectif du shell courant. Le niveau matériel d'une ressource ne peut être défini si cet ID n'est pas celui de l'utilisateur root. Une erreur est générée si vous n'êtes pas un utilisateur root ou si vous essayez de définir le niveau matériel de ressource. Par défaut, l'utilisateur root définit les limites à la fois logicielles et matérielles des ressources. Il doit donc être particulièrement vigilant lors de l'emploi des indicateurs **–S**, **–H** ou de l'indicateur par défaut. A moins que vous ne soyez un utilisateur root, vous ne pouvez définir que la limite logicielle d'une ressource. Si vous modifiez une limite à la baisse, vous ne pourrez plus l'augmenter à nouveau, même pour la ramener à son niveau antérieur. Pour définir les limites d'une ressource, sélectionnez l'indicateur ad hoc et la valeur souhaitée (obligatoirement un entier). Vous ne pouvez définir qu'une ressource à la fois : si vous spécifiez plusieurs indicateurs de ressources, vous risquez d'obtenir des résultats indéfinis. Par défaut, **ulimit** suivi d'une simple valeur définit la taille des fichiers du shell. L'indicateur **–f** est facultatif. La commande *ulimit* accepte les indicateurs suivants : **–c** Définit ou affiche un segment central du shell. **–d** Définit ou affiche un segment de données du shell. **–f** Définit ou affiche la taille des fichiers shell. **–H** Définit ou affiche la limite matérielle d'une ressource (utilisateur racine exclusivement). **–m** Définit ou affiche la mémoire shell. **–s** Définit ou affiche un segment de pile du shell. **–S** Définit ou affiche la limite logicielle d'une ressource. **–t** Définit ou affiche le temps CPU maximal pour le shell.
umask [*nnn*]	Détermine les droits d'accès au fichier. Cette valeur, associée aux droits du process créateur, définit les droits sur un fichier, à sa création (valeur par défaut 022). La valeur par défaut est 022. Si aucune valeur n'est entrée, umask affiche la valeur courante.

unset [*Nom* . . .]	Supprime la variable ou la fonction correspondant à chaque *Nom* spécifié. La définition des variables **PATH**, **PS1**, **PS2**, **MAILCHECK** et **IFS** ne peut être supprimée.
wait [*n*]	Attend la fin de l'exécution du process enfant spécifié par *n* pour sortir, puis renvoie l'état de sortie de ce process. Si *n* est omis, le shell attend la fin de tous les process enfants actifs et renvoie la valeur 0.

Substitution de commandes (shell Bourne).

La substitution de commandes permet de définir comme argument d'une commande le résultat d'une autre commande. Si vous encadrez une ligne de commande d'apostrophes inverses ``, le shell exécute d'abord la (les) commande(s), puis remplace l'intégralité de l'expression (apostrophes inverses comprises) par le résultat. Cette fonction est souvent utilisée pour affecter des variables shell. Par exemple, l'instruction :

```
today=`date`
```

affecte à la variable `today` la chaîne représentant la date courante. L'affectation suivante enregistre, dans la variable `files`, le nombre de fichiers du répertoire courant:

```
files=`ls | wc -l`
```

La substitution de commande peut être exécutée sur n'importe quelle commande dont le résultat est envoyé en sortie standard.

Pour imbriquer des substitutions, faites précéder chaque apostrophe inverse imbriquée d'une barre oblique inverse (\), comme suit :

```
logmsg=`echo Your login directory is \`pwd\``
```

Vous pouvez également attribuer indirectement des valeurs aux variables shell via la commande intégrée **read**, page 12-78. Cette commande lit une ligne à partir de l'entrée standard (le clavier généralement) et affecte successivement les mots de cette ligne aux variables nommées. Par exemple :

```
read first init last
```

lit une ligne de la forme :

```
J. Q. Public
```

avec le même effet que si vous aviez entré :

```
first=J. init=Q. last=Public
```

La commande spéciale **read** affecte les mots résiduels à la dernière variable.

Substitution de variables et de noms de fichiers (shell Bourne)

Le shell Bourne autorise la substitution de variables et de noms de fichiers.

Les sections suivantes traitent de la création et de la substitution de variables dans le shell Bourne :

Substitution de variables (shell Bourne)

Différents mécanismes sont disponibles pour créer des variables (affecter une valeur de chaîne à un nom). Certaines variables, certains paramètres positionnels et certains mots–clés ne peuvent normalement être spécifiés qu'à partir d'une ligne de commande. D'autres variables sont des noms auxquels peuvent être attribuées (par vous ou par le shell) des valeurs de chaîne.

Variables définies par l'utilisateur

Le shell identifie les variables alphanumériques auxquelles des valeurs de chaîne peuvent être affectées. Pour affecter une valeur de chaîne à un nom, tapez :

```
Name=Chaîne
```

Un nom est une série de lettres, de chiffres et de traits de soulignement commençant par un trait de soulignement ou une lettre. Pour utiliser la valeur assignée à une variable, faites précéder le nom d'un signe dollar (`$`). La variable *$Name,* par exemple, utilise la valeur spécifiée par la variable *Chaîne*. Notez l'absence d'espaces de part et d'autre du signe égal (`=`) dans une instruction d'affectation. Les paramètres positionnels ne peuvent apparaître dans une instruction d'affectation : ils ne peuvent être définis que comme décrit à la section Paramètres positionnels, page 12-88. Une ligne de commande peut comporter plusieurs instructions d'affectation, mais n'oubliez pas que le shell les exécute de la droite vers la gauche.

Si vous encadrez la variable *Chaîne* de guillemets ou d'apostrophes (`"` ou `'`), le shell n'interprète pas les caractères blancs, tabulations, deux–points et nouvelle ligne de la chaîne comme des délimiteurs, mais les imbrique littéralement dans la chaîne.

Si vous encadrez la variable *Chaîne* de guillemets (`"`), le shell reconnaît les variables de la chaîne et effectue les substitutions correspondantes, c'est–à–dire, remplace les références à des paramètres positionnels et autres noms de variable précédés du signe dollar (`$`) par leurs valeurs correspondantes. Le shell effectue également la substitution de commande dans les chaînes entre guillemets.

Si vous encadrez la variable *Chaîne* d'apostrophes (`'`), le shell n'effectue pas de substitution de variable ou de commande dans la chaîne. L'exemple suivant en est une illustration :

```
You:            num=875
                 number1="Add $num"
                 number2='Add $num'
                 echo $number1
System:         Add 875
You:            echo $number2
System:         Add $num
```

Le shell ne réinterprète pas les blancs dans les affectations après la substitution de variable. `$first` et `$second` ont donc la même valeur :

```
first='a string with embedded blanks'
second=$first
```

Pour référencer une variable, vous pouvez placer son nom (ou le chiffre désignant le paramètre positionnel) entre accolades `{ }` pour le séparer de la chaîne suivante. En particulier, lorsque le caractère qui suit immédiatement le nom est une lettre, un chiffre ou un trait de soulignement, et que la variable n'est pas un paramètre positionnel, les accolades sont obligatoires :

```
You:            a='This is a'
                echo "${a}n example"
System:         Par exemple :
You:            echo "$a test"
System:         This is a test
```

Pour plus d'informations sur l'utilisation des accolades dans les substitutions de variables, reportez–vous à la section Substitution conditionnelle, page 12-87.

Variables utilisées par le shell

Le shell utilise les variables suivantes. Certaines sont définies par le shell, mais pouvez toutes les définir ou les redéfinir.

CDPATH	Chemin de recherche utilisé par la commande **cd** (changement de répertoire).
HOME	Indique le nom de votre *répertoire de connexion*, qui est le répertoire courant dès que la connexion est établie. Cette variable est initialisée par le programme **login**. La commande **cd** utilise par défaut la valeur de la variable **$HOME**. Utiliser cette variable plutôt qu'un nom de chemin explicite dans une procédure shell permet d'exécuter la procédure à partir d'un répertoire différent sans modification.
IFS	Séparateurs de zones internes, utilisés par le shell pour l'interprétation des blancs. Voir Interprétation des blancs, page 12-87. La variable **IFS** comprend initialement les caractères blanc, tabulation et nouvelle ligne.
LANG	Environnement local à utiliser pour les catégories locales, lorsque ni la variable **LC_ALL,** ni la variable d'environnement correspondante (commençant par **LC**_) n'en spécifient. Pour plus d'informations sur les paramètres régionaux, voir la section relative à la présentation des environnements locaux du manuel *AIX 5L Version 5.3 National Language Support Guide and Reference*.
LC_ALL	Environnement local à utiliser pour remplacer les valeurs des catégories locales spécifiées par la variable **LANG** ou par d'autres variables d'environnement commençant par **LC**_.
LC_COLLATE	Ordre pour le tri des noms et pour la définition d'intervalles de caractères dans les trames.
LC_CTYPE	Environnement local pour interpréter des séquences d'octets de données comme des caractères (c'est–à–dire, différencier les caractères mono–octet des caractères multi–octets dans les arguments et les fichiers entrée), ces caractères étant définis comme des lettres (classe **alpha**), ainsi que le comportement des classes de caractères dans les recherches génériques.
LC_MESSAGES	Langue dans laquelle sont écrits les messages.
LIBPATH	Chemin d'accès aux bibliothèques partagées.
LOGNAME	Nom de connexion, marqué **readonly** dans le fichier **etc/profile**.

MAIL	Chemin d'accès au fichier utilisé par la messagerie pour détecter l'arrivée de courrier. Si cette variable est définie, le shell vérifie périodiquement l'heure à laquelle le fichier a été modifié, et affiche la valeur de **$MAILMSG** si cette heure a changé et que la longueur du fichier est positive. Déclarez la variable **MAIL** dans le fichier **.profile**. Les utilisateurs de la commande **mail** lui affectent généralement la valeur **/usr/spool/mail/$LOGNAME**.
MAILCHECK	Délai (en secondes) entre deux "levées" de courrier par le shell dans les fichiers spécifiés par les variables **MAILPATH** ou **MAIL**. La valeur par défaut est 600 secondes (10 minutes). Si vous attribuez à la variable **MAILCHECK** la valeur 0, le shell effectue une vérification avant chaque invite.
MAILMSG	Message de notification de courrier. Si vous définissez explicitement la variable **MAILMSG** par une chaîne vide (`MAILMSG=""`), aucun message ne s'affiche.
MAILPATH	Liste de noms de fichier, séparés par deux points (: Si cette variable est définie, le shell vous informe de l'arrivée de courrier dans l'un des fichiers de la liste. Vous pouvez faire suivre chaque nom de fichier d'un signe `%` et d'un message qui s'affichera à l'arrivée de courrier. Sinon, le shell utilise la valeur de la variable **MAILMSG** ou, par défaut, le message `[YOU HAVE NEW MAIL]`. **Remarque :** Lorsque la variable **MAILPATH** est définie, ce sont les fichiers correspondants qui sont vérifiés et non le fichier défini par la variable **MAIL**. Pour contrôler à la fois les fichiers définis par la variable **MAILPATH** et par la variable **MAIL**, incluez le fichier **MAIL** dans la liste de fichiers **MAILPATH**.
CHEMIN	Chemin d'accès aux commandes, sous forme de liste de chemins à des répertoires, séparés par des deux points. Lorsqu'il recherche une commande, le shell explore ces répertoires dans l'ordre spécifié. Le répertoire courant est représenté par une chaîne nulle. La variable **PATH** est habituellement initialisée dans le fichier **/etc/environment**, en général par **/usr/bin:/etc:/usr/sbin:/usr/ucb: /usr/bin/X11:/sbin**. Vous pouvez redéfinir cette variable en fonction de vos besoins. La variable **PATH** de votre fichier **.profile** comporte également **$HOME/bin** et votre répertoire courant. Si vous avez défini un répertoire de commandes spécial pour vos projets, **/projet/bin** par exemple, que vous souhaitez explorer en premier, définissez la variable **PATH** comme suit : `PATH=/project/bin:$PATH` Pour affecter à la variable **PATH** une autre valeur, insérez–la de préférence dans le fichier **$HOME/.profile**. Vous ne pouvez pas redéfinir la variable **PATH** si vous exécutez des commandes dans le shell restreint.
PS1	Chaîne constituant l'invite système principale. Cette chaîne est affichée par un shell interactif pour signifier qu'il attend une entrée. La valeur par défaut de **PS1** est `$` suivi d'un espace, pour les utilisateurs non racine.
PS2	Chaîne constituant l'invite secondaire. La variable **PS2** est affichée par le shell pour signifier qu'il attend une suite lorsqu'il rencontre un caractère nouvelle ligne. La valeur par défaut de la variable **PS2** est `>.` suivi d'un blanc.

SHACCT	Nom d'un de vos fichiers. Si cette variable est définie, le shell inscrit un enregistrement comptable à chaque exécution d'un script shell dans ce fichier. Vous disposez ensuite de programmes de comptabilité, tels que **acctcom** et **acctcms** pour analyser les données collectées.
SHELL	Chemin d'accès au shell, enregistré avec l'environnement. Cette variable doit être définie et exportée par le fichier **$HOME/.profile** de chaque connexion restreinte.
TIMEOUT	Délai d'inactivité (en minutes) au bout duquel le shell quitte. Si cette variable est affectée d'une valeur supérieure à zéro (0), le shell quitte si aucune commande n'est lancée dans le délai prescrit, après l'affichage de l'invite **PS1**. (Notez que le shell peut être compilé avec l'indication d'une limite maximale pour le délai.) Les paramètres sont séparés par des espaces.

Variables spéciales prédéfinies

Plusieurs variables ont des significations spéciales. Voici les variables qui ne peuvent être définies que par le shell :

$@	Développe les paramètres positionnels, en commençant par **$1**. Les paramètres sont séparés par des espaces. Si vous encadrez **$@** de guillemets (″), le shell interprète chaque paramètre positionnel comme une chaîne distincte. En l'absence de paramètre positionnel, le shell développe l'instruction en chaîne nulle non déclarée.
$*	Développe les paramètres positionnels, en commençant par **$1**. Le shell sépare chaque paramètre par le premier caractère de la variable **IFS**. Si vous encadrez **$*** de guillemets (″), le shell encadre de guillemets les valeurs du paramètre positionnel. Les valeurs sont séparées par le premier caractère de la valeur de la variable **IFS**.
$#	Spécifie le nombre (décimal) de paramètres positionnels transmis au shell, nom de la procédure shell elle–même exclu. Le paramètre **$#** génère donc le paramètre positionnel ayant la valeur la plus élevée. Ce paramètre sert principalement à vérifier que le nombre d'arguments requis est présent. Seuls les paramètres positionnels **$0** à **$9** sont accessibles via le shell. Pour en savoir plus, reportez–vous à Paramètres positionnels, page 12-88.
$?	Renvoie la valeur de la dernière commande exécutée. Il s'agit d'une chaîne décimale. La plupart des commandes renvoient 0 pour indiquer qu'elles ont été correctement exécutées. Le shell renvoie la valeur courante de la variable**$?** comme valeur de sortie.

$$	Numéro de process du process courant. Un numéro de process étant unique, cette chaîne est souvent utilisée pour générer des noms uniques pour les fichiers temporaires.
	L'exemple suivant illustre comment créer des fichiers temporaires dans un répertoire réservé à cet effet :
	`temp=/tmp/$$`
	`ls >$temp`
	`.`
	`.`
	`.`
	`rm $temp`

$!	Numéro de process de la dernière commande d'arrière–plan invoquée via la terminaison **&**.
$–	Chaîne constituée des noms des indicateurs d'exécution, actuellement définis dans le shell.

Interprétation des blancs

Après exécution des substitutions de variables et de commandes, le shell balaye les résultats, à la recherche des séparateurs de zones (définis dans la variable **IFS**). A chaque occurrence d'un séparateur, il partage la ligne en mots distincts, séparés par des espaces. Il conserve les arguments nuls explicites (`""` ou `''`), mais élimine les implicites (résultant de paramètres non renseignés).

Substitution conditionnelle

En général, le shell remplace l'expression `$`*Variable* par la chaîne affectée, le cas échéant à la variable *Variable*. Une notation spéciale permet cependant d'exécuter une *substitution conditionnelle*, selon que la variable est spécifiée ou non nulle, ou les deux à la fois. Par définition, une variable est définie lorsqu'une valeur lui a été affectée. La valeur d'une variable peut être la chaîne nulle, que vous pouvez affecter d'une des façons suivantes :

`A=`

`bcd=""`

`Efg=''` Affecte la chaîne vide à `A`, `bcd` et `Efg`.

`set '' ""` Affecte la chaîne nulle aux premier et deuxième paramètres positionnels, et annule la définition des autres.

Voici la liste des expressions utilisables dans une substitution conditionnelle.

${ *Variable* **–** *Chaîne* **}**	Si la variable est définie, remplace l'expression par la valeur *Variable*. Sinon, la remplace par la valeur *Chaîne*.
${ *Variable***:–** *Chaîne* **}**	Si la variable est définie et non nulle, remplace l'expression par la valeur *Variable*. Sinon, la remplace par la valeur *Chaîne*.
${ *Variable* **=** *Chaîne* **}**	Si la variable est définie, remplace l'expression par la valeur *Variable*. Sinon, remplace la valeur *Variable* par la valeur *Chaîne*, puis l'expression par la valeur *Variable*. Vous ne pouvez pas affecter ainsi des valeurs aux paramètres positionnels.
${ *Variable***:=** *Chaîne* **}**	Si la variable est définie et non nulle, remplace l'expression par la valeur *Variable*. Sinon, remplace la valeur *Variable* par la valeur *Chaîne*, puis l'expression par la valeur *Variable*. Vous ne pouvez pas affecter ainsi des valeurs aux paramètres positionnels.

${ *Variable* **?** *Chaîne* **}**	Si la variable est définie, remplace l'expression par la valeur *Variable*. Sinon, affiche un message de la forme suivante :
	`Variable : Chaîne`
	et sort du shell courant (sauf s'il s'agit du shell de connexion). Si *Chaîne* est omis, le shell affiche le message suivant :
	`Variable : paramètre nul ou non défini`

${ *Variable***:?** *Chaîne* **}**	Si la variable est définie et non nulle, remplace l'expression par la valeur *Variable*. Sinon, affiche un message de la forme suivante :
	`Variable : Chaîne`
	et sort du shell courant (sauf s'il s'agit du shell de connexion). Si *Chaîne* est omis, le shell affiche le message suivant :
	`Variable : paramètre nul ou non défini`

${ *Variable* **+** *Chaîne* **}**	Si la variable est définie, remplace l'expression par la valeur de *Chaîne*. Sinon, lui substitue la chaîne nulle.
${ *Variable***:+** *Chaîne* **}**	Si la variable est définie et non nulle, remplace l'expression par la valeur de *Chaîne*. Sinon, lui substitue la chaîne nulle.

Lorsqu'il exécute une substitution conditionnelle, le shell n'évalue la variable *Chaîne* qu'au moment où il l'utilise comme chaîne de substitution. Ainsi, dans l'exemple suivant, le shell n'exécute la commande **pwd** que lorsque `d` est nul ou non spécifié :

```
echo ${d:-`pwd`}
```

Paramètres positionnels

Lorsque vous exécutez une procédure shell, le shell crée implicitement des paramètres positionnels qui référencent les mots de la ligne de commande par leur position sur la ligne. Le mot en position 0 (nom de la procédure) est appelé **$0**, le mot suivant (premier paramètre) **$1**, etc, jusqu'à **$9**. Pour référencer des paramètres au–delà de 9, passez par la commande intégrée **shift**, page 12-80.

Pour redéfinir explicitement des paramètres positionnels, vous disposez de la commande intégrée **set**, page 12-79.

> **Remarque :** Lorsqu'une position n'est pas spécifiée, son paramètre positionnel est nul. Les paramètres positionnels sont globaux et peuvent donc être transmis aux procédures shell imbriquées.

Substitution de noms de fichiers (shell Bourne)

Les paramètres de commande sont souvent des noms de fichiers. Vous pouvez générer automatiquement une liste de noms de fichier comme paramètres d'une ligne de commande. Pour ce faire, spécifiez un caractère que le shell identifie comme métacaractère. Si une commande contient ce caractère, le shell remplace le caractère par le nom de fichier enregistré dans un répertoire.

> **Remarque :** Le shell Bourne n'autorise pas le développement de nom de fichier basé sur la classification des caractères par équivalence.

En général, un métacaractère correspond à une lettre alphabétique, mais vous pouvez utiliser des métacaractères spéciaux dans votre trame. Il s'agit des caractères suivants :

*****	Correspond à n'importe quelle chaîne, chaîne nulle comprise
?	Correspond à un seul caractère (quelconque)
[...]	Correspond à n'importe quel caractère entre crochets
[! . . .]	Correspond à n'importe quel caractère entre crochets *autre que* ceux spécifiés à la suite du point d'exclamation

Mise entre crochets, une paire de caractères séparés par un tiret (`-`), correspond à n'importe quel jeu de caractères compris lexicalement entre ces deux valeurs (selon l'ordre de classement binaire des caractères).

L'utilisation de trames souffre quelques restrictions. Si le premier caractère d'un nom de fichier est un point (`.`), il ne peut être mis en correspondance qu'avec une trame commençant aussi par un point. Par exemple, `*` (astérisque) correspond aux fichiers **monfichier** et **votrefichier**, mais pas aux fichiers**.monfichier** et **.votrefichier**. Pour ces fichiers, utilisez, par exemple :

```
.*file
```

Si la recherche n'aboutit pas, c'est la trame elle–même qui est renvoyée en sortie.

Excluez des noms de fichiers et de répertoires les caractères `*`, `?`, `[` et `]` : ils risquent de provoquer une boucle infinie lors des recherches sur la base de trames.

Classes de caractères

Pour faire correspondre des noms de fichiers, vous disposez également des classes de caractères :

```
[[:   charclass :]]
```

Avec ce format, le système établit une correspondance avec chacun des caractères appartenant à la classe spécifiée. Les classes définies correspondent aux sous–routines **ctype**, comme suit :

Classe de caractères	Définition
alnum	Caractères alphanumériques
alpha	Majuscules et minuscules
blank	Espace ou tabulation horizontale
cntrl	Caractères de contrôle
digit	Chiffres
graph	Caractères graphiques
lower	Minuscules
print	Caractères imprimables
punct	Caractères de ponctuation
space	Espace, tabulation horizontale, retour chariot, nouvelle ligne, tabulation verticale ou page suivante
upper	Lettres majuscules
xdigit	Chiffres hexadécimaux

Réacheminement des entrées/sorties (shell Bourne)

La plupart des commandes ignorent l'origine de leurs entrées et la destination de leurs sorties (clavier, écran ou fichier). Une commande peut donc être lancée aussi bien via le clavier que via un pipeline.

Les options ci–après peuvent apparaître n'importe où dans une commande simple. Elles peuvent aussi la précéder ou la suivre, mais ne peuvent lui être transmises.

< *Fichier*	Ouvre le fichier comme entrée standard.
> *Fichier*	Ouvre le fichier comme sortie standard. Crée le fichier s'il n'existe pas, le tronque à une longueur nulle sinon.
> > *Fichier*	Ouvre le fichier comme sortie standard. Créer le fichier s'il n'existe pas, y ajoute les sorties sinon.
<< [**–**] *eofstr*	Lit toutes les lignes à partir de la variable *eofstr* jusqu'à la ligne ne contenant que *eofstr* ou un caractère de fin de fichier. Si un des caractères de *eofstr* est déclaré, le shell n'interprète ni ne développe aucun caractère des lignes d'entrée. Sinon, il exécute la substitution de commande et de variable et ignore le caractère nouvelle ligne déclaré (**\newline**). Pour déclarer des caractères de la variable *eofstr* ou dans les lignes d'entrée, faites–les précéder d'une barre oblique inverse (\). Si vous ajoutez un tiret (–) à l'<< option de réacheminement, toutes les tabulations en tête de la variable *eofstr* et des lignes d'entrée sont supprimées.
<& *Chiffre*	Associe l'entrée standard au descripteur de fichier spécifié par *Chiffre.*
>& *Chiffre*	Associe l'entrée standard au descripteur de fichier spécifié par *Chiffre.*
<&–	Ferme l'entrée standard.
>&–	Ferme la sortie standard.

Remarque : Le shell restreint n'autorise pas le réacheminement des sorties.

Pour en savoir plus sur le réacheminement, voir Réacheminement des entrées/sorties, page 5-1.

Liste des commandes intégrées (shell Bourne)

:, page 12-77	Renvoie un état de sortie nul.
., page 12-77	Lit et exécute les commandes d'un fichier et revient au shell.
break, page 12-77	Sort, le cas échéant, de la boucle **for**, **while** ou **until**.
cd, page 12-77	Passe au répertoire spécifié.
continue, page 12-77	Reprend à l'itération suivante de la boucle **for**, **while** ou **until**.
echo, page 12-77	Ecrit les chaînes de caractères sur la sortie standard.
eval, page 12-77	Lit les arguments comme entrée et exécute la ou les commandes qui en résultent.
exec, page 12-77	Exécute, à la place du shell, la commande spécifiée par *Argument,* sans créer de nouveau process.
exit, page 12-77	Quitte le shell dont l'état de sortie est spécifié par le paramètre *n*.
export, page 12-77	Marque les noms spécifiés, pour exportation automatique vers l'environnement des commandes ultérieures.
hash, page 12-78	Recherche et mémorise l'emplacement du chemin d'accès aux commandes spécifiées.
pwd, page 12-78	Affiche le répertoire courant.
read, page 12-78	Lit une ligne de l'entrée standard.
readonly, page 12-78	Marque en lecture seule le *Nom* spécifié.
return, page 12-78	Provoque la fin d'une fonction, avec une valeur de retour donnée.
set, page 12-79	Contrôle l'affichage de divers paramètres sur la sortie standard.
shift, page 12-80	Décale vers la gauche les arguments de la ligne de commande.
test, page 12-80	Evalue des expressions conditionnelles.
times, page 12-80	Affiche les temps cumulés utilisateur et système, pour le shell et les process exécutés à partir de celui–ci.
trap, page 12-80	Exécute une commande donnée lorsque le shell reçoit un ou plusieurs signaux déterminés.
type, page 12-80	Indique l'interprétation à donner à chaque nom spécifié, lorsqu'utilisé comme nom de commande.
ulimit, page 12-81	Affiche ou ajuste les ressources shell affectées.
umask, page 12-81	Détermine les droits d'accès au fichier.
unset, page 12-82	Supprime la variable ou la fonction correspondant au nom spécifié.
wait, page 12-82	Attend la fin de l'exécution du process enfant et indique son état à la sortie.

Shell C : généralités

Le shell C est un interpréteur de commandes interactif et un langage de programmation. Il utilise une syntaxe qui est similaire au langage de programmation C. La commande **csh** lance le shell C.

Lorsque vous vous connectez, la commande **csh** recherche tout d'abord le fichier de lancement **/etc/csh.cshrc.** propre à l'ensemble du système. Si ce fichier existe, le shell C exécute la commande stockée dans ce fichier. Il exécute ensuite le fichier **/etc/csh.login** (si celui–ci est disponible). Il explore enfin votre répertoire personnel à la recherche des fichiers **.cshrc** et **.login** : ces fichiers (s'ils existent) contiennent des informations personnalisées sur l'exécution du shell C. Toutes les variables éventuellement définies dans les fichiers **.cshrc** et **.login** (de votre répertoire **$HOME**) priment sur celles des fichiers **/etc/csh.cshrc** et **/etc/csh.login**. Seul un utilisateur racine peut modifier les fichiers **/etc/csh.cshrc** et **/etc/csh.login**.

Les fichiers **/etc/csh.cshrc** et **$HOME/.cshrc** sont exécutés lors de la connexion et à chaque fois que la commande **csh** ou un shell C est appelé. Ils sont généralement utilisés pour définir les caractéristiques du shell C, comme les alias et les variables de shell C (par exemple, **history**, **noclobber** ou **ignoreeof**). Il est recommandé de n'utiliser que les commandes intégrées du shell C (reportez–vous à la section Commandes intégrées (shell C), page 12-94) dans les fichiers **/etc/csh.cshrc** et **$HOME/.cshrc** car l'utilisation d'autres commandes peut allonger le temps de démarrage des scripts shell.

Cette section traite des points suivants :

Règles d'utilisation

Voici les règles applicables au shell C :

- Les mots ne peuvent dépasser 1024 octets.
- Les listes d'arguments sont limitées à ARG_MAX octets. La variable ARG_MAX est définie dans le fichier **/usr/include/sys/limits.h**.
- Le nombre d'arguments d'une commande impliquant le développement de noms de fichiers est limité à 1/6ème du nombre de caractères autorisés dans une liste d'arguments.
- Les substitutions de commandes ne peuvent porter sur un nombre de caractères supérieur à celui autorisé dans une liste d'arguments.
- Pour détecter les boucles, le shell limite à 20 le nombre de substitutions d'alias sur une seule ligne.
- La commande **cshcsh** ne prend pas en charge le développement de noms de fichiers basés sur la classification des caractères par équivalence.
- Les descripteurs de fichiers (autres que les entrées et sorties standard et les erreurs standard) ouverts avant que la commande **csh** n'exécute d'application ne sont pas disponibles pour cette application.

Traitement des signaux

Le shell C ignore normalement les signaux quit. Les travaux exécutés distinctement ne sont pas affectés par les signaux générés par le clavier (**INTERRUPT**, **QUIT** et **HANGUP**). Les autres signaux ont la valeur que le shell hérite de son parent. Vous pouvez contrôler le mode de gestion des signaux d'INTERRUPTION et de TERMINAISON dans les procédures shell au moyen de **onintr**. Les shells de connexion interceptent ou ignorent les signaux **TERMINATE** suivant leur configuration. Les autres shells les passent aux process enfants. En aucun cas, les interruptions (signaux **INTERRUPT**) ne sont autorisées lorsqu'un shell de connexion est en train de lire le fichier **.logout**.

Commandes du shell C

Une commande simple est une séquence de mots, séparés par des espaces ou des tabulations.

Un *mot* est une séquence de caractères et/ou de chiffres, sans espaces non délimités. Sont également considérés comme des mots les caractères suivants, lorsqu'ils sont utilisés comme séparateurs de commandes ou caractères de terminaison :

```
&        |       ;
&&       ||     <<          > >
<        >      (            )
```

Ces caractères peuvent faire partie d'un autre mot. Toutefois, les faire précéder d'une barre oblique inverse (\), empêche le shell de les interpréter comme des caractères spéciaux. Les chaînes entourées par ' ' ou " " (guillemets) ou par des apostrophes inverses peuvent également former des parties de mots. Les espaces, tabulations et caractères spéciaux ne constituent pas des mots distincts lorsqu'ils sont encadrés de ces signes. Vous pouvez également encadrer le caractère nouvelle ligne sous réserve de le faire précéder d'une barre oblique inverse (\).

Le premier mot d'une commande simple (numéroté 0) précise généralement le nom de la commande. Les autres mots, à quelques exceptions près, sont passés à cette commande. Si la commande spécifie un fichier exécutable qui est un programme compilé, le shell exécute immédiatement le programme. Si le fichier exécutable n'est pas un programme compilé, le shell suppose qu'il s'agit d'une procédure shell. Le shell lance alors une autre instance de lui–même (un sous–shell) pour lire le fichier et exécuter les commandes qu'il contient.

Cette section traite des points suivants :

Commandes intégrées (shell C)

Les commandes intégrées sont exécutées à l'intérieur du shell. Si une commande de ce type constitue un élément d'un pipeline (excepté le dernier), elle est exécutée dans un sous-shell.

> **Remarque :** Si vous entrez une commande à l'invite du shell C, le système recherche d'abord une commande intégrée. S'il n'en trouve pas, il explore les répertoires spécifiés par la variable shell **path**, à la recherche d'une commande système. Notez que certaines commandes shell C intégrées portent le même nom que des commandes système : ces commandes n'ont toutefois pas nécessairement la même fonction. Pour plus d'informations sur le fonctionnement de la commande, reportez–vous à la description de la commande concernée.

Si vous lancez, à partir du shell, une procédure dont les premiers caractères sont `#!/ShellPathname`, le shell C lance le shell spécifié dans le commentaire pour exécuter la procédure. Sinon, il exécute le shell par défaut (le shell lié à **/usr/bin/sh**). Dans ce cas, les commandes intégrées du shell C ne sont pas reconnues. Pour exécuter les commandes du shell C, commencez la procédure par la ligne `#!/usr/bin/csh`.

Reportez–vous à la section Liste des commandes intégrées (shell C), page 12-119 pour obtenir un classement alphabétique de ces commandes.

Description des commandes

Voici la liste des commandes intégrées du shell C :

alias [*Nom* [*ListeMots*]]	Affiche la liste de tous les alias si aucun paramètre n'est spécifié. Sinon, affiche l'alias du nom indiqué. Si *ListeMots* est une liste de mots, elle est affectée comme alias de la variable *Nom*. Ce nom ne peut être **alias** ou **unalias**.
bg [**%** *Travail*...]	Place le *Travail* spécifié ou le travail courant en arrière-plan, et poursuit, le cas échéant, son exécution.
break	Reprend l'exécution après la **fin** de la commande **foreach** ou **while** la plus proche.
breaksw	Effectue une interruption (break) à partir d'une commande **switch**. Reprend après la commande **endsw** .
case *Label***:**	Définit un *Label* dans une commande **switch**.
cd [*Nom*]	Equivalent à la commande **chdir** (voir description ci–dessous).
chdir [*Nom*]	Change le répertoire en cours par celui spécifié par la variable *Nom*. Si vous ne spécifiez pas la variable *Nom*, la commande change vers votre répertoire local. Si la valeur de la variable *Nom* n'est pas un sous–répertoire du répertoire actuel et ne commence pas par `/`, `./`, or `../`, le shell vérifie chaque élément de la variable shell **cdpath** pour déterminer s'il contient un sous–répertoire correspondant à la variable *Nom*. Si la variable *Nom* est une variable de shell avec une valeur commençant par une barre oblique (`/`), le shell tente cette action dans le but de déterminer s'il s'agit ou non d'un répertoire. La commande **chdir** et la commande **cd** sont équivalentes.
continue	Reprend l'exécution à l'instruction **end** de la commande **foreach** ou **while** la plus proche.
default:	Libelle le cas **default** d'une instruction **switch**. Le cas **default** doit suivre tous les autres labels **case.**
dirs	Affiche la pile de répertoires.
echo	Ecrit les chaînes de caractères sur la sortie standard du shell.
else	Exécute les commandes qui suivent la seconde **else** dans une séquence de commande **if (***Expression***) then**... **else if (***Expression2***) then**... **else**... **endif**.
end	Donne successivement à *Nom* la valeur de chaque élément de la *Liste* et exécute la séquence de commandes entre les instructions **foreach** et **end** correspondantes. Les instructions **foreach** et **end** doivent se trouver seules sur des lignes distinctes. L'instruction **continue** permet de reprendre la boucle, **break** de l'interrompre. Lorsque la commande **foreach** est lue à partir du terminal, un `?` s'affiche pour vous inviter à entrer les *commandes*. Les commandes à l'intérieur des boucles, entrées à l'invite `?`, ne sont pas placées dans la liste d'historique.

endif	Si *Expression* est vraie, exécute les *commandes* qui suivent le premier **then**. Si **else if** *Expression2* est vraie, exécute les *commandes* qui suivent le deuxième **then**. Si **else if** *Expression2* est fausse, exécute les *Commandes* qui suivent le **else**. Vous pouvez spécifier autant de paires **else–if** que vous le souhaitez. Une seule instruction **endif** est requise. Le segment **else** est facultatif. Les mots **else** et **endif** doivent se trouver en tête de ligne. Le segment **if** doit se trouver seul sur une ligne ou suivre une commande **else**.
endsw	Compare les labels **case** avec les valeurs de *Chaîne*. *Chaîne* est d'abord développée (commandes et noms de fichiers). Utilisez les caractères joker *, ?, et [. . .] dans les labels **case**, dont les variables sont développées. Si aucun des labels ne correspond, l'exécution démarre après le label **default**. Les labels **case** et **default** doivent apparaître en début de ligne. La commande **breaksw** entraîne la poursuite de l'exécution après la commande **endsw** . Sinon, le contrôle peut passer aux labels **case** et **default**, comme dans le langage C. Si aucun label ne comprend et qu'il n'existe pas de **default**, l'exécution se poursuit après la commande **endsw**.
eval *Paramètre* . . .	Lit la valeur de la variable *Paramètre* comme entrée du shell et exécute la commande résultante dans le contexte shell courant. Cette commande permet d'exécuter des commandes générées par le résultat de substitution de commandes ou de variables, dans la mesure où l'analyse intervient avant ces substitutions.
exec *Commande*	Exécute la *Commande* spécifiée, à la place du shell courant.
exit [**(** *Expression* **)**	Quitte le shell avec la valeur de la variable shell **status** (si aucune *Expression* n'est spécifiée) ou avec la valeur *Expression* spécifiée.
fg [% *Travail*...]	Amène au premier plan le travail courant ou le *travail* spécifié et, le cas échéant, poursuit son exécution.
foreach *Nom* (*Liste*) *Commande*. . .	Définit tour à tour une variable *Nom* pour chaque membre spécifié par la variable *Liste* et une séquence de commandes, jusqu'à atteindre une commande **end.**
glob *Liste*	Affiche une *liste* avec développement de l'historique, des variables et des noms de fichiers. Les mots sont séparés par un caractère nul et aucun retour chariot n'est inséré à la fin.
goto *Mot*	Poursuit l'exécution après la ligne spécifiée par *Mot*. *Mot* est développé (nom de fichier et commande) pour devenir une chaîne de la forme définie par la variable *Label* . Le shell repasse sur l'entrée autant que possible à la recherche d'une ligne de la forme *Label***:**, éventuellement précédée d'espaces ou de tabulations.
hashstat	Affiche des statistiques indiquant le pourcentage de recherches abouties de commandes, par la table de hachage.
history [**–r** | **–h**] [*n*]	Affiche les listes des événements de l'historique. Cet affichage s'effectue par ordre chronologique. Si *n* est spécifié, seuls les n événements les plus récents sont affichés. L'indicateur **–r** inverse l'ordre d'affichage (événement le plus récent le premier). L'indicateur **–h** affiche la liste sans chiffres en tête. Servez–vous en pour générer des fichiers exploitables avec l'indicateur **–h** de la commande **source**.

if (*Expression*) *Commande*	Exécute la *Commande* spécifiée (arguments compris) si *Expression* est vraie. Les substitutions sur la variable *Commande* ont lieu plus tôt, en même temps que pour le reste de l'instruction **if**. La *commande* spécifiée doit être une commande simple (et non un pipeline, une liste de commandes ou une liste de commandes entre parenthèses). **Remarque :** Les réacheminements d'entrée/sortie interviennent même si *Expression* est fausse et que *Commande* n'est pas exécutée.
jobs [**–l**]	Liste les travaux actifs. Avec **–l** (*l* minuscule), indique, outre les numéros et les ID de **travaux**, les ID processus.
kill –l | [[– *Signal*] % *Travail*...| *PID*...]	Envoie le signal **TERM** (terminate), ou celui spécifié par *Signal* au *Travail* ou au *PID* (process) spécifié. Ces signaux peuvent être un nom ou un nombre, définis dans le fichier **/usr/include/sys/signal.h** (amputés de leur préfixe **SIG**). L'indicateur **–l** (*L* minuscule) liste les noms de signaux.
limit [**–h**] [*Ressource* [*Usage–max*]]	Limite l'usage de la ressource par le process courant et les process qu'il crée. Ces limites sont définies dans le fichier **/etc/security/limits**. Les ressources contrôlables sont : le temps d'utilisation CPU, la taille des fichiers, la taille des données, la taille du vidage de la mémoire centrale et l'utilisation de la mémoire. Les valeurs maximales pour l'affectation de ces ressources sont définies via la commande **mkuser**, au moment de l'ajout de l'utilisateur au système. Elles peuvent être modifiées à l'aide de la commande **chuser**. Les limites sont d'ordre logiciel ou matériel. Les utilisateurs peuvent définir des limites logicielles, jusqu'aux plafonds imposés par le matériel. Cependant, seul un utilisateur root est habilité à modifier les limites matérielles. L'indicateur **–h** affiche les limites matérielles. Si *Usage–Max* n'est pas spécifié, la commande **limit** affiche la limite courante de la ressource spécifiée. Si *Ressource* n'est pas spécifiée, elle affiche les limites courantes de toutes les ressources. Pour en savoir plus sur les ressources contrôlées par la commande **limit**, reportez-vous aux sous-routines **getrlimit, setrlimit** et **vlimit** dans le manuel *AIX Technical Reference, Volume 1: Base Operating System and Extensions*: Base Operating System and *Extensions Volume 1*. Le paramètre *Usage–Max* relatif au temps d'utilisation CPU est au format hh:mm:ss. Pour les autres ressources, il est spécifié par un nombre en virgule flottante ou par un entier éventuellement suivi d'une unité de mesure. Cette unité de mesure peut être : k ou kilo-octets (1 024 octets), m ou mega-octets, ou b ou blocs (unités utilisées par la sous-routine **ulimit** comme expliqué dans le manuel *AIX 5L Version 5.3 Technical Reference: Base Operating System and Extensions Volume 2*). Si vous ne spécifiez pas d'unité de mesure, k est utilisé pour toutes les ressources. Pour les noms de ressources et les unités de mesure, des préfixes non ambigus suffisent. **Remarque :** Cette commande ne limite la mémoire physique disponible pour un process que si d'autres process actifs ont besoin de mémoire système.

connexion	Met fin à un shell de connexion et le remplace par une occurrence de la commande **/usr/bin/login.** C'est un moyen de se déconnecter (cette commande a été incluse pour des raisons de compatibilité avec les commandes **ksh** et **bsh**).
logout	Termine un shell de connexion. Particulièrement utile si l'option **ignoreeof** est active.
nice [**+** *n*] [*Commande*]	Si n n'est pas spécifié, définit à 24 la priorité d'exécution des commandes dans le shell. Si **+***n* est spécifié, définit la priorité au nombre spécifié. Si l'indicateur +n et la *Commande* sont spécifiés, exécute la *Commande* à 24 plus le nombre spécifié. Si vous êtes utilisateur root, vous pouvez exécuter l'instruction **nice** avec un nombre négatif. *Commande* est toujours exécutée dans un sous–shell, et les restrictions applicables aux commandes d'une instruction **if** simple valent pour cette commande.
nohup [*Commande*]	Si *Commande* n'est pas spécifiée, les arrêts (**hangups**) sont ignorés pour toute la suite du script. Si *Commande* est spécifiée, ils ne sont ignorés que pour cette *Commande*. Pour exécuter un pipeline ou une liste de commandes, placez–le (–la) dans un script shell, accordez à celui–ci des droits d'exécution et définissez le shell comme valeur de *Commande*. Tous les process exécutés en arrière–plan avec une perluète (&) sont protégés de manière effective: ils ne recevront pas de signal **hangup** lorsque vous vous déconnecterez. Ils restent néanmoins susceptibles de recevoir des signaux **hangup** explicites, sauf si vous utilisez l'instruction **nohup**.
notify [**%** *Travail*...]	Indique au shell de vous prévenir (de façon asynchrone) lorsque l'état du travail courant ou du *travail* spécifié change. Normalement, le shell émet un avertissement juste avant d'afficher l'invite. Cette fonction est automatiquement activée si la variable shell **notify** est spécifiée.
onintr [**–** | *Label*]	Contrôle l'action du shell au niveau des interruptions. En l'absence d'argument, restaure l'action par défaut (arrêt du script ou retour au niveau entrée des commandes). Avec l'indicateur **–**, ignore toutes les interruptions. Si *Label* est spécifié, le shell exécute l'instruction **goto** *Label* à réception d'une interruption ou lorsqu'un process enfant se termine suite à une interruption. Dans tous les cas, si le shell est exécuté séparément et que les interruptions sont ignorées, l'instruction **onintr** n'a pas de sens : les interruptions sont toujours ignorées du shell et de toutes les commandes appelées.
popd [**+** *n*]	Détruit la pile de répertoires et revient au nouveau premier répertoire. Si vous spécifiez +*n*, la commande rejette la *n*ième entrée de la pile. Les éléments de la pile sont numérotés de haut en bas, à partir de 0.
pushd [**+** *n* | *Nom*]	Sans arguments, intervertit les deux premiers éléments de la pile des répertoires. Avec la variable *Nom*, passe au nouveau répertoire en décalant l'ancien répertoire courant (indiqué dans la variable shell **cwd**) dans la pile. Avec la variable +*n*, place le *n*ième élément de la pile en tête, et en fait le répertoire courant. Les éléments de la pile sont numérotés de haut en bas, à partir de 0.

rehash	Recalcule la table d'adressage du contenu des répertoires dans la variable shell **path**. Cette action est requise si de nouvelles commandes sont ajoutées aux répertoires dans la variable **path** pendant que vous êtes connecté. La commande **rehash** n'est obligatoire que si des commandes sont ajoutées dans un répertoire utilisateur ou si un changement intervient au niveau du contenu d'un répertoire système.
repeat *Compte Commande*	Exécute la *Commande* spécifiée, soumise aux mêmes restrictions que l'instruction **if**, autant de fois que spécifié par *Compte*. **Remarque :** Les réacheminements d'E–S interviennent une et une seule fois, même si *Compte* vaut 0.
set [[*Nom* [n]] [**=** *Mot*]] | [*Nom* = (*Liste*)]	Sans argument, affiche la valeur de toutes les variables shell. Les variables don't la valeur contient plus d'un mot s'affichent sous la forme d'une liste de mots entre parenthèses. Si *Nom* seul est spécifié, affecte à *Nom* la chaîne nulle. Sinon, affecte à *Nom* la valeur de *Mot* ou de la liste de mots spécifiés dans *Liste*. Si *n* est précisé, affecte au *n*ième élément de *Nom* (s'il existe) la valeur de *Mot*. Dans tous les cas, la valeur est développée (nom de fichier et commande). Les arguments peuvent être répétés pour affecter plusieurs valeurs via une seule commande **set**. Le développement de variables intervient pour tous les arguments, avant toute affectation.
setenv *Nom Valeur*	Affecte à la variable d'environnement spécifiée par *Nom* la valeur *Valeur* (chaîne unique). Les variables d'environnement les plus courantes (**USER**, **TERM**, **HOME** et **PATH**) sont automatiquement importées et exportées à partir des variables shell C **user**, **term**, **home** et **path**. Il est inutile d'exécuter l'instruction **setenv** pour ces variables.
shift [*Variable*]	Décale vers la gauche les éléments de la variable shell **argv** ou de la *Variable* spécifiée. Une erreur se produit si la variable shell **argv** n'est pas définie ou que la *variable* spécifiée contient moins d'un mot.
source [**–h**] *Nom*	Lit les commandes spécifiées par la variable *Nom.* Vous pouvez imbriquer des commandes **source**. Cependant, si les imbrications sont trop poussées, vous risquez de déborder des descripteurs de fichiers. Une erreur dans une commande **source** met fin à toutes les autres commandes **source** imbriquées. Normalement, les entrées effectuées au cours de l'exécution de **source** ne sont pas enregistrées dans la liste d'historique. L'indicateur **–h** place la commande dans l'historique sans l'exécuter.
stop [**%** *Travail*...]	Arrête le travail courant ou le *Travail* spécifié, exécuté en arrière-plan.
suspend	Met fin au shell (équivaut à un signal **STOP**).
switch (*chaîne*)	Démarre la séquence de commandes **switch (** *Chaîne* **) case** *Chaîne***:**... **breaksw default:**... **breaksw endsw**. Cette séquence de commande compare successivement chaque étiquette à la valeur de la variable *Chaîne*. Cette séquence compare successivement chaque label "case" à la valeur de la variable Chaîne.

time [*Commande*]	Contrôle le minutage automatique des commandes. Si *Commande* n'est pas spécifiée, affiche un récapitulatif du temps consommé par le shell et ses enfants. Si *Commande* est spécifiée, la commande est minutée. Le shell affiche ensuite un récapitulatif, comme décrit pour la variable shell **time**, page 12-115. Si nécessaire, un shell supplémentaire est créé, pour afficher les statistiques à la fin de la commande.
	Les exemples suivants utilisent **time** avec la commande **sleep**:
	`time sleep`
	La sortie de cette commande est similaire à celle–ci :
	`0.0u 0.0s 0:00 100% 44+4k 0+0io 0pf+0w`
	Les zones de sortie sont :
	Zone
	Description
	premier
	Nombre de secondes du temps CPU consacrées au process utilisateur
	second
	Nombre de secondes de temps CPU utilisées par le noyau pour le compte du process utilisateur
	Troisième
	Temps écoulé (pendule murale) pour la commande
	Quatrième
	Temps utilisateur CPU total plus temps système, comme pourcentage du temps écoulé
	Cinquième
	Somme moyenne de la taille de mémoire partagée et de la quantité moyenne d'espace mémoire non partagé utilisé (en kilo–octets).
	Sixième
	Nombre d'opérations d'entrée/sortie de blocs
	Septième
	Somme des erreurs de pagination et du nombre de permutations (swaps).

umask [*Valeur*]	Détermine les droits d'accès au fichier. Cette *Valeur*, associée aux droits du process créateur, détermine les droits sur un fichier au moment de sa création. La valeur par défaut est 022. A défaut de *Valeur* spécifiée, la valeur en cours est affichée.
unalias *| *Trame*	Désactive les alias dont le nom correspond à la *Trame.* La commande **unalias *** supprime tous les alias. L'absence d'alias ne provoque pas d'erreur.
unhash	Désactive l'usage de la table de hachage pour la localisation des programmes en cours.

unlimit [**–h**][*Ressource*]	Supprime la limitation imposée à la variable *Ressource*. Si *Ressource* n'est pas spécifiée, toutes les limitations sont supprimées. Reportez–vous à la description de la commande **limit** pour avoir la liste des noms *Ressource*. L'indicateur **–h** supprime les limites matérielles correspondantes. Seul un utilisateur root est habilité à effectuer cette opération.
unset *l *Trame*	Supprime toutes les variables dont le nom correspond à la *Trame*. Utilisez **unset *** pour supprimer toutes les variables. L'absence de variables ne provoque pas d'erreur.
unsetenv *Trame*	Supprime de l'environnement toutes les variables dont le nom correspond à la *Trame*. (Reportez–vous à la commande intégrée **setenv**.)
wait	Attend tous les travaux en arrière–plan. Si le shell est interactif, un INTERRUPT (Ctrl–C généralement) met fin à l'attente. Le shell affiche ensuite les noms et numéros des travaux en suspens.
while (*Expression*) *Commande*. . . **end**	Evalue les *commandes* entre les instructions **while** et **end** correspondantes, tant que la variable *Expression* est différente de zéro. L'instruction **break** permet d'interrompre la boucle, **continue** de la poursuivre. Les instructions **while** et **end** doivent se trouver chacune sur une ligne distincte. Si l'entrée provient d'un terminal, des invites s'affichent après while (*Expression*), comme pour l'instruction **foreach**.
@ [*Nom* [*n*] **=** *Expression*]	Sans argument, affiche le valeurs de toutes les variables shell. Sinon, affecte *Nom* à la variable *Expression* . Si l'expression contient un des caractères <, >, & ou |, cette partie de l'expression doit être placée entre parenthèses. Lorsque *n* est spécifié, l'élément *n* @T>ième >de la variable >*Nom* >est définie sur la valeur de la variable>*Expression* >. A la fois la variable >*Nom* >et son élément >*n* >@T>ième >doivent exister. Les opérateurs du langage C, tels que *= et +=, sont disponibles. L'espace séparant la variable *Nom* de l'opérateur d'affectation est facultatif. Les espaces séparant les éléments de *Expression* sont en revanche obligatoires (faute de quoi, il seraient interprétés comme un seul mot). Les suffixes spéciaux, (++) et (—) augmentent et diminuent, respectivement, la valeur de la variable *Nom*.

Expressions et opérateurs du shell C

La commande intégrée **@** et les instructions **exit**, **if**, et **while** comportent des opérateurs semblables à ceux du langage C, avec les mêmes règles de précédence. Voici les opérateurs disponibles :

Opérateur	Signification
`()`	change la précédence
`~`	complément
`!`	négation
`*/ %`	multiplie, divise, modulo
`+ –`	ajoute, soustrait
`<< > >`	décale à gauche, décale à droite
`<= >= < >`	opérateurs relationnels

Opérateur	Signification
== != =~ !~	comparaison de chaîne/correspondance de trames
&	opérateur au niveau bit AND
^	opérateur au niveau bit OR exclus
|	opérateur au niveau bit OR inclus
&&	AND logique
||	OR logique

Les opérateurs sont classés par ordre de précédence décroissante (de haut en bas, et de gauche à droite).

Remarque : Les opérateurs + et – sont associatifs, en partant de la droite. Ainsi, a + b – c est évalué :

```
a + (b - c)
```

et non comme suit :

```
(a + b) - c
```

Les opérateurs ==, !=, =~, et !~ comparent des arguments chaîne. Tous les autres opèrent sur des nombres. Les opérateurs =~ et !~ sont semblables à == et !=, excepté que l'expression de droite est une *trame* à laquelle est comparé l'opérande de gauche. Cela réduit la nécessité d'utiliser l'instruction **switch** dans les procédures shell.

Les opérateurs logiques ou (||) et et (&&) sont également disponibles. Ils permettent de vérifier un intervalle de nombres, comme dans l'exemple suivant :

```
if ($#argv > 2 && $#argv < 7) then
```

Ici, le nombre d'arguments doit être supérieur à 2 et inférieur à 7.

Les chaînes commençant par zéro (0) sont interprétées comme des nombres octaux. Les arguments nuls ou manquants sont interprétés comme 0. Toutes les expressions ont pour résultat des chaînes représentant des nombres décimaux. Notez que les deux éléments d'une expression peuvent se trouver dans un même mot. Excepté lorsqu'ils se trouvent à côté d'éléments syntaxiquement significatifs pour l'analyseur (& | < > ( )), les éléments d'expression doivent être encadrés d'espaces.

On peut trouver dans des expressions, à titre d'opérandes, des exécutions de commandes entre ( ) et des interrogations de fichier de la forme (**–opérateur** *Nomfichier*), où **opérateur** est l'un des caractères suivants :

r	accès en lecture
w	accès en écriture
x	accès en exécution
e	existence
o	propriété
z	taille zéro
f	fichier ordinaire
d	répertoire

Nomfichier est développé (commande et nom de fichier), puis testé pour vérifier s'il a la relation spécifiée à l'utilisateur réel. Si le *nom de fichier* n'existe pas ou est inaccessible, toutes les requêtes retournent la valeur faux (0). Si la commande aboutit, l'interrogation retourne la valeur vrai (1). Sinon, elle renvoie la valeur faux (0). Si vous souhaitez davantage de précisions sur l'état, lancez la commande en dehors d'une expression et examinez la variable shell **status**.

Substitution de commandes (shell C)

Lors d'une *substitution de commandes*, le shell exécute une commande et remplace cette commande par son résultat. Pour exécuter des substitutions de commandes dans le shell C, encadrez la commande ou la chaîne de commandes par des apostrophes inverses (` `). Le shell décompose normalement le résultat de la commande en mots séparés au niveau des espaces, des tabulations et des caractères nouvelle ligne. Puis il remplace la commande d'origine par cette sortie.

Dans l'exemple suivant, les apostrophes inverses (` `) encadrant la commande **date** indiquent que le résultat de la commande sera remplacé :

```
echo The current date and time is: `date`
```

Le résultat de cette commande est du type :

```
The current date and time is: Wed Apr 8 13:52:14 CDT 1992
```

Le shell C exécute sélectivement les substitutions de commandes sur les arguments des commandes intégrées. Ce qui signifie qu'il ne développe pas les parties des expressions qui ne sont pas évaluées. Dans le cas de commandes non intégrées, le shell effectue la substitution du nom de la commande en dehors de la liste d'arguments. La substitution a lieu dans un shell enfant du shell principal, après réacheminement des entrées/sorties.

Si une chaîne de commandes est encadrée de " ", le shell interprète comme séparateurs de mots les seuls caractères nouvelle ligne, préservant les espaces et les tabulations à l'intérieur du mot. Dans tous les cas, l'unique caractère nouvelle ligne final ne génère pas de nouveau mot.

Exécution des commandes Shell C non intégrées

Lorsque le shell C détermine qu'une commande n'est pas une commande intégrée, il tente de l'exécuter via le sous–programme **execv**. Chaque mot de la variable shell **path** correspond à un répertoire à partir duquel le shell tente de lancer la commande. En l'absence des options **–c** et **–t**, le shell ventile les noms de ces répertoires dans une table interne. Le shell ne tente l'exécution de **exec** dans un répertoire que s'il y a une chance que la commande s'y trouve. Si vous désactivez ce mécanisme au moyen de la commande **unhash** ou que vous associez au shell l'option **–c** ou **–t**, le shell est concaténé avec le nom de la commande considérée pour former le chemin d'accès à un fichier. Le shell effectue la même opération pour chaque élément de répertoire de la variable **path** ne commençant pas par /. Il tente ensuite d'exécuter la commande.

Les commandes entre parenthèses sont toujours exécutées dans un sous–shell. Par exemple :

```
(cd ; pwd) ; pwd
```

affiche le répertoire personnel sans modifier l'emplacement du répertoire courant. En revanche, la commande :

```
cd ; pwd
```

définit le répertoire personnel comme répertoire courant. Les commandes entre parenthèses sont le plus souvent utilisées pour empêcher la commande **chdir** d'agir sur le shell courant.

Si le fichier est accessible en exécution, sans être un fichier exécutable binaire, le shell suppose qu'il s'agit d'un fichier de commandes shell et lance un nouveau shell pour le lire.

S'il existe un alias du shell, les mots de l'alias sont insérés en préfixe de la liste d'arguments, pour former la commande shell. Le premier mot de l'alias doit être le chemin d'accès complet au shell.

Substitution d'historique (shell C)

La substitution d'historique permet de modifier des mots d'une commande précédente pour en créer de nouvelles. Cette fonction simplifie la reprise de commandes et d'arguments d'une commande précédente pour les intégrer à la commande courante, et permet également de corriger aisément les fautes de frappe dans une commande.

Les substitutions d'historique commencent par un point d'exclamation (`!`) et peuvent se trouver n'importe où sur la ligne de commande, sachant qu'elles ne peuvent être imbriquées (c'est–à–dire qu'une substitution d'historique ne peut en contenir une autre). Vous pouvez faire précéder le point d'exclamation (`!`) d'une barre oblique inversée (`\`) pour annuler la signification spéciale du point d'exclamation. En outre, si vous placez le point d'exclamation (`!`) avant un espace, une tabulation, un caractère nouvelle ligne, un signe égal (`=`) ou une parenthèse ouvrante (`(`), la substitution d'historique n'a pas lieu.

Une substitution d'historique peut également être générée par un caret (`^`) inséré en début de ligne. Le shell envoie toute ligne contenant des substitutions d'historique à la station de travail avant d'exécuter la ligne.

Cette section traite des points suivants :

Liste d'historique

La liste d'historique sauvegarde les commandes lues par le shell, constituées de un ou de plusieurs mots. La substitution d'historique réintroduit des séquences de mots issues de ces commandes sauvegardées, dans le flux des entrées.

La variable shell **history** contrôle la taille de la liste historique. Vous devez définir cette variable soit dans le fichier **.cshrc**, soit sur la ligne de commande avec la commande intégrée **set**. La commande précédente est retenue quelle que soit la valeur de la variable **history**. Les commandes de l'historique sont numérotées à partir de 1. La commande intégrée **history** génère une sortie du type :

```
9 write michael
10 ed write.c
11 cat oldwrite.c
12 diff *write.c
```

Le shell affiche les chaînes de commandes avec leur numéro d'événement. Le numéro d'événement apparaît à gauche de la commande et est représenté lorsque la commande a été entrée par rapport aux autres commandes de l'historique. Il est généralement inutile de se servir des numéros pour faire référence aux événements, mais vous pouvez intégrer le numéro de l'événement en cours à l'invite système en plaçant un point d'exclamation (`!`) dans la chaîne d'invite affectée à la variable d'environnement **PROMPT**.

Une référence historique complète contient une spécification d'événement, un descripteur de mot, et un ou plusieurs modificateurs, selon le format général suivant :

```
Evénement[.]Mot:Modificateur[:Modificateur] . . .
```

> **Remarque :** Seul un mot peut être modifié. Les chaînes contenant des espaces ne sont pas admises.

Dans l'exemple de sortie précédent, l'événement courant porte le numéro 13. En conservant cet exemple, voici des références à des événements précédents :

`!10`	Evénement numéro 10.
`!-2`	Evénement numéro 11 (événement courant moins 2).
`!d`	Mot commençant par `d` (événement numéro 12).
`!?mic?`	Mot contenant la chaîne `mic` (événement numéro 9).

Ces formats, sans autre modification, réintroduisent simplement les mots des événements spécifiés, séparés par de simples espaces. Cas particuliers : (double point d'exclamation) fait référence à la commande précédente. La commande **!!** seule sur une ligne d'entrée relance la commande précédente.

Spécification d'événement

Pour sélectionner des mots d'un événement, faites suivre la spécification d'événement par deux points (`:`) et l'un des désignateurs de mots suivants (les mots d'une ligne d'entrée sont numérotés de manière séquentielle à partir de 0) :

0	Premier mot (nom de la commande).
n	*n*ième argument.
^	Premier argument.
$	Dernier argument.
%	Mot correspondant à une chaîne ?*Chaîne*? le précédant immédiatement.
x–y	Plage de mots à partir du *x* ième mot jusqu'au *y* @T>ième mot
–y	Intervalle de mots entre le premier (0) et le *y*ième mot.
*****	Premier mot du dernier argument, ou rien s'il y a un seul mot (le nom de la commande) dans l'événement.
x *	*x*ième et le dernier argument.
x **–**	Equivalent à *x* *, avec omission du dernier argument.

Vous pouvez omettre les deux points entre la spécification d'événement et le descripteur de mot si celui–ci commence par ^ (caret), $ (dollar), * (astérisque), – (tiret) ou % (pourcentage). Vous pouvez également placer une séquence de modificateurs parmi les suivants après le descripteur facultatif du mot, en les faisant précéder de deux points :

h	Supprime l'extension finale d'un chemin d'accès, en conservant le début.
r	Supprime un élément final `.xxx`, en conservant le nom racine.
e	Supprime tout sauf le suffixe `.xxx`.
s/ *AncienMot* **/** *NouveauMot* **/**	Affecte à *AncienMot* la valeur de *NouveauMot*.

La partie gauche d'une substitution ne constitue pas une trame, au sens d'une chaîne reconnue par un éditeur. Il s'agit en fait d'un mot, isolé, sans espace. Normalement, une barre oblique (`/`) délimite le mot d'origine (*OldWord*) et le mot de remplacement (*NewWord*). Toutefois, vous pouvez utiliser un caractère quelconque comme délimiteur. Dans l'exemple suivant, l'utilisation du signe `%` comme délimiteur permet d'inclure une barre oblique `/` dans les mots :

```
s%/home/myfile%/home/yourfile%
```

Le shell remplace une perluète (&) par le texte *OldWord* dans la variable *NewWord*. Dans l'exemple suivant, **/home/myfile** devient **/temp/home/myfile**.

```
s%/home/myfile%/temp&%
```

Le shell remplace un mot nul dans une substitution soit par la dernière substitution, soit par la dernière chaîne utilisée dans la chaîne de balayage contextuelle `!? String ?`. Vous pouvez omettre le délimiteur de fin (`/`) si un caractère nouvelle ligne suit immédiatement. Pour délimiter une liste d'historique, vous disposez des modificateurs suivants :

t	Supprime tous les chemins d'accès en tête, en conservant la fin.
&	Répète la substitution précédente.
g	Applique la modification globalement.
p	Affiche la nouvelle commande, sans l'exécuter.
q	Déclare les mots remplacés, prévenant ainsi toute substitution ultérieure.
x	Equivalent du modificateur **q**, mais décompose les mots au niveau des espaces, des tabulations et des caractères nouvelle ligne.

Lorsque vous utilisez les modificateurs précédents, la modification ne s'applique qu'au premier mot modifiable à moins que le modificateur **g** ne précède le modificateur sélectionné.

Si vous indiquez une référence historique sans spécification d'événement (par exemple, `!$`), le shell utilise comme événement la commande précédente. Si une référence historique antérieure intervient sur la même ligne, le shell répète la référence précédente. Ainsi, la séquence suivante génère les premier et dernier arguments de la commande correspondant à `?foo?`.

```
!?foo?^ !$
```

Une abréviation spéciale remplace une référence historique lorsque le premier caractère non blanc d'une ligne d'entrée est un ^ (caret). Ce format équivaut à `!:s^`, et fournit ainsi un raccourci commode pour les substitutions sur le texte de la ligne précédente. La commande `^ lb^ lib` corrige l'orthographe de `lib` dans la commande précédente.

Si nécessaire, vous pouvez encadrer une substitution d'historique par { } (accolades) pour l'isoler des caractères qui la suivent. Par exemple, si vous souhaitez utiliser une référence à la commande :

```
ls -ld ~paul
```

pour exécuter la commande :

```
ls -ld ~paula
```

utilisez la construction :

```
!{l}a
```

Dans cet exemple, `!{l}a` recherche une commande commençant par `l` et lui ajoute un `a` final.

Guillemets et apostrophes

Pour éviter trop d'interprétations pour toutes ou certaines des substitutions, mettez les chaînes entre guillemets et apostrophes. De simples apostrophes (' ') empêchent que la chaîne soit interprétée, tandis que des guillemets (" ") permettent des développements ultérieurs. Dans les deux cas, le texte résultant devient (en totalité ou en partie) un mot simple.

Substitution d'alias (shell C)

Un *alias* est un nom attribué à une commande ou à une chaîne de commande. Le shell C permet d'attribuer des alias et de les utiliser comme des commandes. Le shell maintient une liste des alias définis.

Lorsque le shell a examiné la ligne de commande, il la décompose en commandes distinctes et recherche le premier mot de chaque commande, de gauche à droite, pour déterminer s'il s'agit d'un alias. Dans l'affirmative, le shell remplace, au moyen du mécanisme d'historique, le texte de l'alias par celui de la commande qu'il référence. Les mots résultants remplacent la commande et la liste d'arguments. En l'absence de référence à la liste d'historique, la liste d'arguments reste inchangée.

Pour en savoir plus sur le mécanisme d'historique du shell C, reportez–vous à Substitution d'historique (shell C), page 12-104.

Les commandes intégrées **alias** et **unalias** établissent, affichent et modifient la liste des alias. Voici le format de la commande alias :

```
alias [Nom [ListeMots]]
```

La variable *Nom* (facultative) spécifie l'alias du nom indiqué. Si vous spécifiez une liste de mots avec la variable *ListeMots*, la commande l'affecte comme alias de la variable *Nom*. Si vous lancez la commande **alias** sans variable, elle affiche tous les alias du shell C.

Si l'alias de la commande **ls** est `ls –l`, la commande :

```
ls /usr
```

est remplacée par :

```
ls –l /usr
```

La liste d'arguments reste inchangée car il n'y a pas de référence à la liste d'historique dans la commande avec alias. De même, si l'alias de la commande **lookup** est :

```
grep \!^ /etc/passwd
```

le shell remplace `lookup bill` par :

```
grep bill /etc/passwd
```

Dans cet exemple, `!^` fait référence à la liste d'historique et le shell la remplace par le premier argument de la ligne d'entrée, `bill`.

Vous pouvez utiliser des caractères joker dans un alias. La commande :

```
alias lprint 'pr &bslash2.!* >

> print'
```

crée une commande qui formate ses arguments pour une imprimante ligne. Le caractère `!` est protégé du shell dans l'alias, de sorte qu'il n'est pas développé tant que la commande **pr** n'est pas exécutée.

Si le shell détecte un alias, il effectue la transformation de mot dans le texte d'entrée et relance le processus alias sur la nouvelle ligne. Si le premier mot du texte suivant est le même que le précédent, un indicateur de fin (du processus alias) empêche le bouclage. Les autres boucles ultérieures sont détectées et génèrent une erreur.

Substitution de variables et de noms de fichiers (shell C)

Le shell C autorise la substitution de variables et de noms de fichiers.

Cette section traite des points suivants :

Substitution de variables (shell C)

Le shell C maintient un ensemble de variables, chacune d'elles ayant pour valeur une liste de zéro ou plusieurs mots. Certaines sont définies ou référencées par le shell. Par exemple, la variable **argv** est une image de la liste de variables shell, et les mots comportant la valeur de cette variable sont référencés de façon particulière.

Vous pouvez modifier et afficher la valeur des variables via les commandes **set** et **unset**. Certaines variables sont des "bascules", c'est–à–dire qu'elles activent ou désactivent une fonction : le shell ne fait aucun cas de leur valeur, seul importe le fait qu'elles soient ou non activées. Par exemple, la variable **verbose** provoque l'affichage (écho) des commandes entrées. La définition de cette variable est activée par l'indicateur **–v** sur la ligne de commande.

D'autres opérations exécutent un traitement numérique des variables. La commande **@** effectue des calculs, dont le résultat est affecté à une variable. Les valeurs des variables sont toutefois toujours représentées par des chaînes (zéro ou plusieurs). Pour les opérations numériques, la chaîne nulle est évaluée à zéro, et les mots suivants sont ignorés.

Lorsque vous émettez une commande, le shell analyse la ligne d'entrée et exécute les substitutions d'alias. Ensuite, avant d'exécuter la commande, il exécute les substitutions de variables. Le caractère $ (dollar) code la substitution. Il est néanmoins transmis tel quel s'il est suivi d'un espace, d'une tabulation ou d'un caractère nouvelle ligne. Faire précéder le caractère $ d'une barre oblique inverse (\) empêche ce développement, sauf dans les deux cas suivants :

- La commande est entre guillemets (" "). Le shell exécute alors la substitution.
- La commande est entre apostrophes (') : le shell n'exécute alors jamais la substitution. Les chaînes entre ` ` sont interprétées au niveau de la substitution de commande. (Reportez–vous à Substitution de commande (shell C), page 12-103.

Le shell détecte les réacheminements d'entrées/sorties avant de développer les variables, et les développe séparément. Sinon, le nom de la commande et la liste complète d'arguments sont développés simultanément. Il est ainsi possible que le premier mot (commande) génère plusieurs mots, dont le premier devient le nom de la commande, les suivants étant des paramètres.

Sauf s'il est entre guillemets (" ") ou associé au modificateur **:q**, le résultat d'une substitution de variable peut faire lui–même l'objet d'une substitution de commande et de nom de fichier. Une variable entre guillemets, composée de plusieurs mots, est développée sous la forme d'un seul mot (ou d'une portion de mot), les mots initiaux étant séparés par des espaces. Si vous spécifiez le modificateur **:q**, la variable est développée en plusieurs mots, entre guillemets, séparés par des espaces, pour empêcher toute substitution ultérieure (de commande ou de nom de fichier).

Voici la syntaxe d'entrée des valeurs de variable dans le shell. Sauf indication contraire, référencer une variable non définie par la commande **set** génère une erreur.

Les modificateurs **:gh**, **:gt**, **:gr**, **:h**, **:r**, **:q** et **:x** sont applicables aux substitutions suivantes. Si des accolades ({ }) apparaissent dans le format de la commande, les modificateurs doivent se trouver entre accolades. Il n'est autorisé qu'un seul modificateur **:** (deux points) pour chaque développement de variable.

$ *Nom*	
${ *Nom***}**	Remplacé par les mots affectés à la variable *Nom*, séparés par des espaces. Les accolades isolent la variable *Nom* des caractères suivants (qui lui seraient sinon intégrés). Les noms des variables shell commencent par une lettre et comportent au maximum 20 lettres et chiffres, ou caractères de soulignement (_). Si *Nom* ne spécifie pas une variable shell, mais est défini dans l'environnement, sa valeur est alors retournée. Les modificateurs précédés de deux points, de même que les autres formats ici décrits, ne sont pas disponibles dans ce cas.
$ *Nom* **[***numéro***]**	
${ *Nom* **[** *numéro* **]}**	Ne sélectionne que quelques mots de la variable *Nom*. Le numéro, qui peut faire l'objet de substitution de variable, est soit un chiffre seul, soit deux chiffres séparés par un tiret (–). Le premier mot de la chaîne représentant la valeur de la variable est numéroté 1. Par défaut, le premier chiffre d'un intervalle vaut 1 et le dernier, **$#** *Nom*. Le symbole* (astérisque) sélectionne tous les mots. Si le second argument est omis ou qu'il se trouve dans un intervalle, un premier intervalle vide ne génère pas d'erreur.
$# *Nom*	
${# *Nom***}**	Indique le nombre de mots de la variable. Cette fonction peut être utilisée dans **[***numéro***]** comme indiqué ci–dessus. Par exemple, `$Nom[$#Nom]`.
$0	Remplace le nom du fichier à partir duquel sont lues les entrées de commandes. Si ce nom est inconnu, une erreur est générée.
$ *nombre*	
${ *nombre* **}**	Equivalent à `$argv[ nombre ]`.
$*	Equivalent à `$argv[*]`.

Les modificateurs **:** ne sont pas applicables aux substitutions suivantes :

$? *nom*	
${? *nom***}**	Remplace par la chaîne 1 si *nom* est définie, par 0 (zéro) sinon.
$?0	Remplace par 1 si le nom du fichier d'entrée courant est reconnu, par 0 (zéro) sinon.
$$	Remplace par le numéro (décimal) du process du shell parent.
$<	Remplace par une ligne de l'entrée standard, sans autre interprétation. Cette substitution permet de lire à partir du clavier dans une procédure shell.

Substitution de nom de fichier (shell C)

Le shell C offre plusieurs fonctions de raccourci permettant d'économiser du temps et de réduire le nombre de frappes de touches. Si un mot contient l'un des caractères `*`, `?`, `[ ]` ou `{ }`, ou commence par un (~), ce mot fait l'objet de substitutions de noms de fichiers. Le shell C considère le mot comme un terme générique et le remplace selon une liste alphabétique de noms de fichiers.

La séquence de classement courante est spécifiée par les variables d'environnement **LC_COLLATE** ou **LANG**. Dans une liste de mots spécifiant des substitutions de noms de fichiers, une erreur est générée si aucun terme générique ne correspond à un fichier existant. Il n'est toutefois pas nécessaire que tous les termes génériques aient un correspondant. Seuls les caractères `*`, `?` et `[ ]` indiquent des correspondances ou des développements de noms de fichiers. Le tilde (~) et les accolades `{ }` indiquent une abréviation de nom de fichier.

Développement de nom de fichier

L'astérisque (`*`) remplace n'importe quelle chaîne de caractères, chaîne nulle comprise. Par exemple, dans un répertoire contenant les fichiers :

```
a aa aax alice b bb c cc
```

La commande `echo a*` affiche tous les fichiers commençant par `a` :

```
a aa aax alice
```

Remarque : Lors de la comparaison, les caractères point (`.`) et barre oblique (`/`) doivent également correspondre.

Le caractère ? (point d'interrogation) remplace un seul caractère. La commande :

```
ls a?x
```

affiche la liste des fichiers commençant par a, suivis d'un seul caractère et terminés par x :

```
aax
```

Pour rechercher un caractère ou un ensemble de caractères, encadrez le(s) caractère(s) de crochets ([]). La commande :

```
ls [abc]
```

affiche la liste des fichiers correspondant exactement à l'un des caractères entre crochets :

```
a b c
```

Entre les crochets, un intervalle de caractères est indiqué par `[a-z]`. Les caractères correspondants sont définis par la séquence de classement courante.

Abréviation de nom de fichier

Les caractères ~ (tilde) et { (accolade ouvrante) indiquent une abréviation de nom de fichier. Un ~ en début de fichier représente les répertoires personnels. Isolé, le caractère ~ prend la valeur de votre répertoire personnel, défini par la variable **home**. Par exemple, la commande

```
ls ~
```

affiche tous les fichiers et répertoires de votre répertoire **$HOME**.

Lorsque cette commande est suivie d'un nom composé de lettres, de chiffres et de tirets (–), le shell recherche un utilisateur portant ce nom et remplace le répertoire **$HOME** de l'utilisateur.

> **Remarque :** Si le ~ est suivi d'un caractère autre qu'une lettre ou une barre oblique (/), ou qu'il apparaît ailleurs qu'au début d'un mot, il n'est pas développé.

Pour faire correspondre des caractères dans des noms de fichiers sans taper l'intégralité du nom, encadrez les noms de fichiers d'accolades ({ }). Ainsi, `a{b,c,d}e` est un raccourci de `abe ace ade`. Le shell préserve l'ordre (de gauche à droite) et, à cet effet, enregistre séparément le résultat des correspondances à un niveau inférieur. Cette construction peut être imbriquée. Ainsi, la commande :

```
~source/s1/{oldls,ls}.c
```

devient :

```
/usr/source/s1/oldls.c /usr/source/s1/ls.c
```

si le répertoire personnel **source** est **/usr/source**. De même, la commande :

```
../{memo,*box}
```

peut devenir :

```
../memo ../box ../mbox
```

> **Remarque :** `memo` n'est pas trié avec le résultat de la comparaison `*box`. Dans ce cas particulier, les accolades { (ouvrante), } (fermante) et } (par paire) sont passées sans changement.

Classes de caractères

Pour effectuer des comparaisons, vous pouvez également vous servir des classes de caractères. Avec le format suivant, le système établit une correspondance avec chacun des caractères appartenant à la classe spécifiée :

```
[:   charclass :]
```

Les classes définies correspondent aux sous–routines **ctype**.

Classe de caractères	Définition
alnum	Caractères alphanumériques
alpha	Majuscules et minuscules
cntrl	Caractères de contrôle
digit	Chiffres
graph	Caractères graphiques
lower	Minuscules
print	Caractères imprimables
punct	Caractères de ponctuation
space	Espace, tabulation horizontale, retour chariot, nouvelle ligne, tabulation verticale ou page suivante
upper	Lettres majuscules
xdigit	Chiffres hexadécimaux

Supposons que vous soyez dans un répertoire contenant les fichiers :

```
a aa aax Alice b bb c cc
```

A l'invite du shell C, entrez :

```
ls [:lower:]
```

Le shell C affiche la liste de tous les fichiers dont le nom commence par une minuscule:

```
a aa aax b bb c cc
```

Pour en savoir plus, reportez–vous à la commande **ed**.

Variables d'environnement (shell C)

Certaines variables ont une signification spéciale pour le shell. Par exemple, **argv**, **cwd**, **home**, **path**, **prompt**, **shell** et **status** sont toujours définies par le shell. Sauf pour **cwd** et **status**, cette action n'a lieu qu'au moment de l'initialisation. Ces variables conservent leur valeur tant que vous ne les modifiez pas explicitement.

La commande **csh** copie les variables d'environnement **USER**, **TERM**, **HOME** et **PATH** dans les variables **csh**, **user**, **term**, **home** et **path**, respectivement. Les valeurs sont recopiées dans l'environnement à chaque réinitialisation des variables de shell normales. La variable **path** ne peut être définie que dans le fichier **.cshrc** car les sous–programmes **csh** importent la définition des chemins à partir de l'environnement et la réexportent si elle a été modifiée.

Voici les variables spéciales du shell :

argv	Arguments passés aux scripts shell. Les paramètres positionnels sont remplacés à partir de cette variable.
cdpath	Liste de répertoires, à explorer par la commande **chdir** ou **cd** lors de recherche de sous-répertoires.
cwd	Chemin d'accès complet au répertoire courant.
echo	Activé par l'indicateur **–x** sur la ligne de commande, affiche (écho) chaque commande et ses arguments avant qu'ils soient exécutés. Les commandes non intégrées sont développées avant d'être affichées. Les commandes intégrées le sont avant toute substitution de commande ou de nom de fichier, dans la mesure où ces substitutions sont ensuite effectuées de façon sélective.
histchars	Chaîne modifiant les caractères utilisés dans une substitution d'historique. Le premier caractère devient le caractère de substitution d'historique (remplaçant le caractère par défaut ! (point d'exclamation). Le second remplace le caractère ^ (caret) dans les substitutions rapides. **Remarque :** Affecter à **histchars** un caractère utilisé dans une commande ou dans un nom de fichier peut provoquer des substitutions d'historique non souhaitées.
historique	Valeur numérique contrôlant la taille de la liste d'historique. Toute commande référencée dans les nombres d'événements permis n'est pas prise en compte. Indiquer une valeur trop importante pour la variable **history** peut provoquer un dépassement mémoire. Qu'elle soit définie ou non, elle sauvegarde la dernière commande exécutée sur la liste d'historique.
home	Répertoire personnel, initialisé par l'environnement. Le développement de nom de fichier du caractère ~ (tilde) fait référence à cette variable.
ignoreeof	Spécifie au shell d'ignorer les caractères EOF (fin de fichier) issus des stations de travail. Ceci pour empêcher les shells d'être accidentellement tués à la lecture d'un caractère de fin de fichier (Ctrl–D).
mail	Fichiers à examiner par le shell pour le courrier. Ce que le shell effectue après chaque commande dont le résultat est une invite, à l'expiration du délai imparti. Le shell affiche le message `Mail in file.` si le fichier existe et que l'heure de son dernier accès est antérieure à l'heure de sa dernière modification.

	Si le premier mot de la variable **mail** est numérique, il indique un nouvel intervalle de vérification du courrier (en secondes – valeur par défaut 600, soit : 10 minutes). Si vous spécifiez plusieurs fichiers courrier, le shell affiche le message `New mail in file`, quand du courrier arrive dans le fichier concerné.
noclobber	Règlemente les réacheminements en sortie pour garantir qu'aucun fichier ne sera accidentellement détruit et que les réacheminements effectuent des ajouts aux fichiers antérieurs.
noglob	Désactive le développement de nom de fichier. Cette fonction est très utile pour les procédures ne traitant pas de noms de fichiers, ou lorsqu'une liste de noms de fichiers a été générée et qu'aucun autre développement n'est souhaitable.
nonomatch	Spécifie de ne pas générer d'erreur si le développement d'un nom de fichier ne correspond à aucun fichier existant, mais de renvoyer la trame initiale. Un format incorrect de la trame provoquera toujours une erreur.
notify	Spécifie au shell de vous avertir en mode asynchrone des modifications intervenues au niveau de l'état des travaux. Par défaut, ces modifications sont affichées juste avant l'affichage de l'invite shell.
path	Spécifie les répertoires à explorer à la recherche des commandes à exécuter. Un mot nul spécifie le répertoire courant. Si aucune variable **path** n'est définie, seuls les chemins d'accès complets sont pris en compte. Le chemin par défaut (extrait du fichier **/etc/environment** utilisé pour la connexion) est : `/usr/bin /etc /usr/sbin /usr/ucb /usr/bin/X11 /sbin`
	Un shell non associé à l'indicateur **–c** ni **–t** répartit normalement le contenu des répertoires dans la variable **path**, après lecture du fichier **.cshrc** et chaque fois que **path** est redéfini. Si de nouvelles commandes sont ajoutées à ces répertoires pendant que le shell est actif, vous devez attribuer la commande **rehash**. Sinon, il se peut que les commandes ne soient pas retrouvées.
invite	Chaîne affichée avant chaque lecture de commande à partir d'une station de travail interactive. Si un ! (point d'exclamation) apparaît dans la chaîne, il est remplacé par le numéro de l'événement courant. Si le caractère ! se trouve dans une chaîne entre guillemets ou entre apostrophes, il doit être précédé d'une barre oblique inversée (\). L'invite par défaut est %, pour les utilisateurs non racine. L'invite par défaut est `#`, pour les utilisateurs non root.
savehist	Valeur numérique contrôlant le nombre d'entrées de la liste d'historique, sauvegardées dans le fichier **~/.history** lorsque vous vous déconnectez. Toute commande référencée dans ce nombre d'événements est sauvegardée. Au cours du lancement, le shell lit **~/.history** dans la liste d'historique, assurant la sauvegarde de l'historique à travers les multiples connexions. Donner à la variable **savehist** une valeur trop importante ralentit le lancement du shell.
shell	Indique le fichier dans lequel se trouve le shell C. Cela permet aux shells parallèles d'interpréter les fichiers dotés de bits d'exécution, mais non exécutables par le système. Cette valeur prend initialement la valeur du fichier personnel du shell C.
status	Etat retourné par la dernière commande. Si la commande se termine anormalement, 0200 est ajouté à l'état. Les commandes intégrées qui n'aboutissent pas retournent un état de sortie de 1 ; celles qui aboutissent renvoie un état de 0 (zéro).

time	Contrôle le minutage automatique des commandes. Lorsque cette variable est définie, toute commande dépassant le délai imparti (en nombre de secondes CPU) affichera, en fin d'exécution, une ligne relative aux ressources utilisées. Pour en savoir plus, reportez–vous à la commande intégrée **time**, page 12-100.
verbose	Activée par l'option **–v** sur la ligne de commande, affiche les mots de chaque commande, après exécution des substitutions d'historique.

Réacheminement des entrées/sorties (shell C)

Avant d'exécuter une commande, le shell C balaye la ligne d'entrée à la recherche de caractères de réacheminement. Ces derniers lui indiquent de réacheminer les entrées et les sorties.

Pour réacheminer les entrées et sorties standard d'une commande, vous disposez des instructions suivantes :

< *Fichier*	Ouvre le *Fichier* spécifié (qui est d'abord développé – variable, commande et nom de fichier) comme fichier d'entrée standard.
<< *Mot*	Lit l'entrée shell jusqu'à la ligne correspondant à la valeur de *Mot*. La variable *Mot* ne peut faire l'objet de substitution de variable, de nom de fichier ou de commande. Chaque ligne entrée lui est comparée avant toute substitution sur la ligne. Sauf si un caractère de déclaration (\, ", ' ou `) figure dans la variable *Mot*, le shell exécute les substitutions de variable et de commande sur les lignes concernées, la barre oblique inversée (\) pouvant servir à déclarer les caractères $ (dollar), \ (barre oblique inversée) et ` (apostrophe). Dans les commandes remplacées, les espaces, les tabulations et les caractères ligne suivante sont préservés, sauf le dernier caractère nouvelle ligne, qui n'est pas pris en compte. Le texte résultant est placé dans un fichier temporaire anonyme, qui devient l'entrée standard de la commande.
>*Fichier* **>!** *Fichier* **>&** *Fichier* **>&!** *Fichier*	Ouvre le *Fichier* spécifié comme sortie standard. Si le *fichier* n'existe pas, il est créé. S'il existe, il est tronqué et son contenu antérieur est perdu. Si la variable **noclobber** est définie, le fichier doit ne pas exister ou être un fichier spécial caractère, faute de quoi une erreur est générée, ceci pour contribuer à prévenir toute destruction accidentelle de fichiers. Dans ce cas, recourez au format comportant un ! (point d'exclamation) pour éliminer ce contrôle. *Fichier* est développé de la même façon que pour les <fichiers d'entrée. Le format >& (perluète) achemine la sortie standard et la sortie d'erreur vers *Fichier*. L'exemple suivant illustre comment acheminer séparément la sortie standard vers **/dev/tty** et la sortie d'erreur standard vers **/dev/null**. Les parenthèses sont obligatoires pour la séparation : `% (find / -name vi -print > /dev/tty) >& /dev/null`
> > *Fichier* **> >!** *Fichier* **> >&** *Fichier* **> >&!** *Fichier*	Utilise *Fichier* comme sortie standard (comme >), mais *ajoute* la sortie à la fin du fichier. Si la variable **noclobber** est définie, une erreur se produit si le fichier n'existe pas, sauf si vous employez un format avec ! (point d'exclamation). Sinon, l'instruction est semblable à >.

Une commande reçoit l'environnement dans lequel le shell a été appelé, modifié le cas échéant par les options d'entrée/sortie et la présence de la commande en tant que pipeline. Ainsi, contrairement à certains shells précédents, les commandes exécutées à partir d'un fichier de commandes shell n'ont pas accès au texte des commandes par défaut. A l'inverse, elles reçoivent les entrées standard d'origine, directement du shell. Utilisez le << mécanisme pour présenter les données en ligne, qui permet aux fichiers de commande shell de fonctionner comme des éléments de pipelines et permet aussi au bloc shell de lire son entrée. Notez que l'entrée standard par défaut pour une commande exécutée séparément n'est pas remplacée par le fichier vide **/dev/null**. Au lieu de cela, l'entrée standard demeure l'entrée standard d'origine du shell.

Pour réacheminer les sorties d'erreur via un tube vers la sortie standard, utilisez le format |& (barre verticale, perluète) et non la seule | (barre verticale).

Flux de contrôle

Le shell contient des commandes qui peuvent servir à réguler le flux de contrôle dans les fichiers de commandes (scripts shell) et, de façon limitée mais fort utile, dans les entrées ligne de commande. Ces commandes forcent le shell à relire ou à sauter ses entrées.zl

Les instructions **foreach, switch** et **while** et le format **if–then–else** de l'instruction **if**, requièrent que les mots-clés principaux apparaissent dans une seule commande simple sur une ligne d'entrée.

Si l'entrée shell ne peut être explorée, le shell place dans des tampons les entrées chaque fois qu'une boucle est lue et se sert du tampon interne pour effectuer les relectures requises par la boucle. Des **goto** arrière permettent ensuite de localiser les entrées que vous n'avez pu rechercher.

Contrôle des travaux (shell C)

Le shell associe à chaque process un numéro de travail. Il maintient un tableau des travaux en cours et leur affecte des numéros (petits nombres entiers). Lorsqu'un travail est lancé en arrière–plan avec un caractère `&` (perluète), le shell affiche une ligne du type :

```
[1] 1234
```

Cette ligne indique que le travail porte le numéro `1` et qu'il se compose d'un seul process dont l'ID est `1234`. Pour afficher la table des travaux en cours, lancez la commande intégrée **jobs**, page 12-97

Un travail en arrière–plan doit affronter la concurrence pour obtenir des données en entrée à partir de la station de travail. Il en est de même pour les données en sortie, qui sont imbriquées avec les sorties des autres travaux.

Vous pouvez référencer un travail dans le shell Korn de plusieurs façons : Le signe pourcentage (`%`) introduit un nom de travail. il peut s'agir du numéro de travail ou du nom de la commande qui a lancé le travail s'il s'agit d'un nom unique. Par exemple, si un process **make** est exécuté comme travail 1, vous pouvez le désigner comme suit : `%1`. Vous pouvez également le désigner comme suit : `%make` s'il n'y a qu'un travail suspendu commençant par la chaîne `make`. Vous pouvez également utiliser le suivant :

```
%?Chaîne
```

pour spécifier un travail dont le nom contient la variable `Chaîne`, s'il n'existe qu'un seul travail de ce type.

Le shell détecte immédiatement les changements d'état des process. Si un travail bloque, rendant impossible la poursuite du traitement, le shell envoie un message à la station de travail. Ce message ne s'affiche que lorsque vous appuyez sur Entrée. Si, toutefois, la variable **notify** est définie, le shell émet immédiatement un message indiquant un changement au niveau de l'état des travaux en arrière–plan. Au moyen de la commande intégrée **notify**, page 12-98, marquez les process pour lesquels vous souhaitez que les changements d'état vous soient notifiés. Par défaut, la commande **notify** marque le process courant.

Liste des commandes intégrées du shell C

@ à la page 12-101	Affiche la valeur des variables shell spécifiées.
alias, page 12-95	Affiche les alias spécifiés (tous, par défaut).
bg, page 12-95	Place à l'arrière–plan le travail courant ou les travaux spécifiés.
break, page 12-95	Reprend l'exécution après la fin de la commande **foreach** ou **while** la plus proche.
breaksw, page 12-95	Interrompt une commande **switch.**
case, page 12-95	Définit un label dans une commande **switch**.
cd, page 12-95	Passe au répertoire spécifié.
chdir, page 12-95	Passe au répertoire spécifié.
continue, page 12-95	Poursuit l'exécution de la commande **foreach** ou **while** la plus proche.
default, page 12-95	Etiquette le cas par défaut d'une instruction **switch.**
dirs, page 12-95	Affiche la pile de répertoires.
echo, page 12-95	Ecrit les chaînes de caractères sur la sortie standard du shell.
else, page 12-95	Exécute les commandes qui suivent le deuxième **else** dans une séquence de commandes **if (** *Expression* **) then** ... **else if (** *Expression2***) then** ... **else** ... **endif**.
end, page 12-95	Indique la fin d'une séquence de commandes précédée de la commande **foreach**.
endif, page 12-96	Exécute les commandes qui suivent le deuxième **then** dans une séquence de commandes **if (** *Expression* **) then** ... **else if (** *Expression2* **) then** ... **else** ... **endif**.
endsw, page 12-96	Marque la fin d'une séquence de commandes **switch (** *Chaîne* **) case** *Chaîne***:**... **breaksw default:**... **breaksw endsw**. Cette séquence de commandes compare successivement chaque étiquette à la valeur de la variable *Chaîne*. L'exécution se poursuit après un **endsw** si une commande **breaksw** est exécutée ou qu'aucun label ne correspond et qu'aucune valeur par défaut n'est définie.
eval, page 12-96	Lit les arguments comme entrée pour le shell et exécute le(s) commande(s) résultante(s), dans le contexte du shell courant.
exec, page 12-96	Exécute la commande spécifiée, à la place du shell courant.
exit, page 12-96	Quitte le shell avec la valeur de variable d'état du shell, ou celle de l'expression spécifiée.
fg, page 12-96	Amène à l'avant–plan le travail courant ou les travaux spécifiés, et poursuit leur exécution même s'ils ont été arrêtés.
foreach, page 12-96	Définit tour à tour une variable *Nom* pour chaque membre spécifié par la variable *Liste* et une séquence de commandes, jusqu'à atteindre une commande **end.**
glob, page 12-96	Affiche une liste avec développement de l'historique, des variables et des noms de fichiers.

goto, page 12-96	Poursuit l'exécution à partir d'une ligne déterminée.
hashstat, page 12-96	Affiche des statistiques indiquant le pourcentage de recherches abouties de commandes, par la table de hachage.
history, page 12-96	Affiche les listes des événements de l'historique.
if, page 12-97	Exécute une commande donnée si l'expression spécifiée est vraie.
jobs, page 12-97	Liste les travaux actifs.
kill, page 12-97	Envoie le signal **TERM** (terminate), ou celui spécifié par *Signal*, aux travaux ou aux process spécifiés.
limit, page 12-97	Limite l'usage de la ressource par le process courant et les process qu'il crée.
login, page 12-98	Met fin à un shell de connexion et le remplace par une occurrence de la commande **/usr/sbin/login.**
logout, page 12-98	Termine un shell de connexion.
nice, page 12-98	Définit la priorité des commandes exécutées dans le shell.
nohup, page 12-98	Provoque la non prise en compte des arrêts dans la suite de la procédure.
notify, page 12-98	Le shell vous prévient (de façon asynchrone) lorsque l'état du travail courant ou du travail spécifié change.
onintr, page 12-98	Contrôle l'action du shell au niveau des interruptions.
popd, page 12-98	Détruit la pile de répertoires et revient au nouveau premier répertoire.
pushd, page 12-98	Intervertit des éléments de la pile des répertoires.
rehash, page 12-99	Recalcule, dans la variable shell path, la table d'adressage contenant les répertoires.
repeat, page 12-99	Exécute la commande spécifiée, soumise aux mêmes restrictions que l'instruction **if**, autant de fois que spécifié.
set, page 12-99	Affiche la valeur de toutes les variables shell.
setenv, page 12-99	Modifie la valeur de la variable d'environnement spécifiée.
shift, page 12-99	Décale vers la gauche la variable spécifiée.
source, page 12-99	Lit la commande spécifiée par la variable *Nom*.
stop, page 12-99	Arrête les travaux (courants ou spécifiés) exécutés en arrière–plan.
suspend, page 12-99	Met fin au shell (équivaut à un signal **STOP**).
switch, page 12-99	Marque le début d'une séquence de commandes **switch (*Chaîne*) case *Chaîne*:**... **breaksw default:**... **breaksw endsw**. Cette séquence de commande compare successivement chaque étiquette à la valeur de la variable *Chaîne*. Cette séquence compare successivement chaque label "case" à la valeur de la variable Chaîne.
time, page 12-100	Affiche un récapitulatif des temps consommés par le shell et son process enfant.
umask, page 12-100	Détermine les droits d'accès au fichier.
unalias, page 12-100	Désactive les alias dont le nom correspond à la *Trame*.
unhash, page 12-100	Désactive l'usage de la table de hachage pour la localisation des programmes en cours.
unlimit, page 12-101	Supprime toute limitation de ressources.

unset, page 12-101	Supprime toutes les variables dont le nom correspond à la *Trame*.
unsetenv, page 12-101	Supprime de l'environnement toutes les variables dont le nom correspond à la *Trame*.
wait, page 12-101	Attend tous les travaux en arrière–plan.
while, page 12-101	Evalue les commandes de la séquence qui va du **while** au **end** correspondant, tant que la variable *Expression* a une valeur non nulle.

Voir aussi

Shell Korn

Commandes **ksh** et **stty**,

Commande **alias**, **cd**, page 12-47,

Commande **export**, page 12-40,

Commande **fc**, page 12-48,

Commande **getopts**, page 12-48,

Commande **read**, page 12-49,

Commandes de shell Korn **set**, page 12-41 et **typeset**, page 12-45,

Fichier **/etc/passwd**,

Shell Bourne

Commandes **bsh** ou **Rsh**, commande **login**,

Commande spéciale shell Bourne **read**,

Sous–routines **setuid** et **setgid**,

Fichier spécial **null**,

Format de fichier **environment** et **profile**.

Shell C

La commande **csh**, commande **ed**.

L'**alias**, page 12-95,

unalias, page 12-100,

jobs, page 12-97,

notify, page 12-98 et **et**, page 12-99 Commandes intégrées du shell c.

Chapitre 13. Documentation AIX

La documentation AIX en ligne est disponible sur le CD-ROM suivant :
– 86 X2 32EM : Hypertext Library. Pour AIX 5.3

Les instructions d'installation d'*Hypertext Library* sont contenues dans le livret du CD-ROM et doivent être suivies à la lettre.

Hypertext Library est fourni avec un ensemble d'outils appelé *The Hypertext Library Utilities.* Cet ensemble d'outils comporte une fonction **Search** permettant la recherche d'informations dans toute la bibliothèque, de même qu'une fonction **Multi-Print** avec laquelle vous pouvez imprimer plusieurs documents au moyen d'un simple clic dans la fenêtre Search Results ; il offre également des fonctions permettant la création et la gestion des **remarques** dans l'ensemble de la bibliothèque ainsi que l'**ajout** de documentation **utilisateur** à la bibliothèque (ou le retrait de cette documentation).

Hypertext Library et *Hypertext Library Utilities* possèdent une interface graphique et texte.

Le contenu de la documentation *Hypertext Library* et les outils *Hypertext Library Utilities* sont décrits dans la page d'accueil *Hypertext Library*.

Annexe A. Accessibilité

L'accessibilité permet à des personnes dotées de compétences différentes d'utiliser les moyens informatiques. Cette annexe décrit les caractéristiques d'accessibilité d'AIX 5L™ et leur utilisation.

Le logiciel AIX ® est compatible avec plusieurs normes d'accessibilité disponibles sur le marché. Les utilisateurs dotés de compétences différentes peuvent ainsi accéder à la documentation d'assistance et à l'ensemble des fonctions prises en charge par AIX via ses interfaces. Vous pouvez également personnaliser la souris et le clavier.

Utilisation de la ligne de commande

Toutes les fonctions du logiciel AIX sont disponibles via la ligne de commande de sorte qu'elles peuvent être prises en charge par des technologies assistées compatibles avec l'environnement emacs. L'interface de commande est également disponible à distance à l'aide des technologies assitées telles que JAWS qui prend en charge les environnements telnet dans le système d'exploitation Windows.

Utilisation de Web–based System Manager et de SMIT

Web–based System Manager offre une accessibilité clavier complète et une prise en charge limitée pour le changement de taille de police pour les administrateurs système. Le lecteur d'écran n'est actuellement pas pris en charge. Les menus System Management Interface Tool (SMIT) sont accessibles à l'aide des mêmes technologies assistées qui fournissent une accessiblité via la ligne de commande. La version textuelle de SMIT (qui est activée via la commande **smitty** sur un terminal graphique) doit être utilisée à la place de la version Motif ou GUI activée à l'aide de la commande **smit**.

Utilisation des extensions clavier X pour augmenter l'accessibilité dans l'environnement X WindowSystem

Si vous utilisez telnet ou un programme d'émulation de terminal pour accéder à AIX, les caractéristiques d'accessibilité du clavier sont fournies soit par l'application, soit par votre environnement d'exploitation. Par exemple, dans le système d'exploitation Windows, vous pouvez ouvrir le Panneau de configuration et sélectionner Options d'accessibilité pour activer et définir les réglages d'accessibilité du clavier. Les options d'accessibilité du clavier sont également disponibles sur la console graphique AIX exécutant le logiciel X Window System ou l'environnement Common Desktop (CDE).

Les extensions clavier X prennent en charge les fonctions d'accessibilité clavier standard de l'environnement et des applications X Window System. Le tableau suivant fournit une description de ces fonctions et des moyens permettant de les activer.

Tableau 1. Extensions clavier X prenant en charge les fonctions d'accessibilité clavier

Fonction	Description	Activation des fonctions de navigation clavier
Touches rémantentes (StickyKeys)	Interprètent des pressions séquentielles de touches de modification comme si elles étaient actionnées simultanément. Par exemple, la combinaison de touches Ctrl+Alt+Suppr et la combinaison MAJ+combinaison numérique qui permet d'obtenir un symbole. Cette fonction est destinée aux utilisateurs qui ne peuvent pas appuyer sur plusieurs touches en même temps.	Appuyez 5 fois sur la touche MAJ pour activer les touches rémanentes (StickyKeys). Un signal audio est émis lorsque cette fonction est activée et utilisée. Appuyez 5 fois sur la touche MAJ ou appuyez simultanément sur deux touches de modification pour les désactiver. Un signal audio est émis lorsque cette fonction est désactivée.
Touches à bascule (ToggleKeys)	Emettent un signal audio lorsqu'une touche à bascule (touches Verr. Maj. et Verr. Num., par exemple) est actionnée. Cette fonction est destinée aux utilisateurs souffrant d'une déficience visuelle.	Activation et désactivation automatique comme pour les touches rémanentes (StickyKeys).
Touches affleurantes (SlowKeys)	Permettent d'éliminer les erreurs de manipulation de touches par les utilisateurs souffrant de déficience visuelle.	Appuyez sur la touche MAJ de droite pendant 12 secondes. Au bout de 8 secondes, des bips sonores retentissent. A l'issue des 4 secondes restantes, des bips de confirmation retentissent.
Touches de répétition (RepeatKeys)	Activent et désactivent les principales commandes de répétition automatique. Cette fonction a un effet sur le délai de répétition automatique et les commandes d'intervalle. Cette fonction est destinée aux utilisateurs à mobilité réduite.	Activation et désactivation automatique comme pour les touches affleurantes (SlowKeys). Dans l'environnement CDE, activez ou désactivez la fonction Répétition automatique via l'application clavier dans le gestionnaire de styles CDE.
Touches rebond (BounceKeys)	Permettent d'éliminer les erreurs liées à l'actionnement répété de la même touche par les utilisateurs à mobilité réduite.	Activation et désactivation automatique comme pour les touches affleurantes (SlowKeys).
Touche souris (MouseKeys)	Permet de contrôler le pointeur de la souris à l'aide du clavier.	Appuyez sur la combinaison de touches MAJ + Verr. Num. après avoir modifié le fichier keymap. Reportez–vous aux explications ci–dessous.

Activation des extensions clavier X

Vous devez modifier le fichier **/usr/lpp/X11/defaults/xserverrc** afin d'activer les extensions clavier X sur le système AIX.

L'entrée de fichier **xserverrc** appelle les options suivantes :

+kb	Active les extensions clavier
+accessx	Active les extensions d'accessibilité

Vous pouvez ajouter ces options au fichier **/usr/lpp/X11/defaults/xserverrc** en ajoutant la ligne suivante au fichier :

```
EXTENSIONS="$EXTENSIONS +kb +accessx"
```

Touche souris (MouseKeys)

MouseKeys vous permet d'utiliser les touches de direction et les touches numériques du pavé à 10 touches du clavier pour contrôler le pointeur de souris à l'écran. Cette fonction vous permet aussi de sélectionner des objets comme si vous utilisiez les boutons sur la souris.

Cette fonction est activée lorsque le fichier keymap est modifié. Les fichiers keymap sont spécifiques à l'emplacement et sont situés dans le répertoire **/usr/lpp/X11/defaults/xmodmap/** *locale* **/keyboard** où *locale* est le nom de vos paramètres régionaux système, par exemple **en_US**.

Après avoir localisé le fichier keymap, éditez le fichier de définition **xmodmap keyboard** pour remplacer l'entrée de la version décalée de la touche Verr. Num. par la chaîne suivante : Pointer_EnableKeys.

Voici un exemple de la ligne de définition pour la touche Verr. Num. :

```
!Keypad         Base        Shift                 Alt-Gr(Mod2)
!----------     -------     --------              ------------------
Keycode 98 =    Num_Lock    Pointer_EnableKeys    NoSymbol
```

Utilisation du gestionnaire de styles dans l'environnement Common Desktop (CDE)

Utilisez le gestionnaire de styles CDE pour personnaliser la couleur, la police et la forme du pointeur de la souris ainsi que la sortie audio.

Changement de la couleur et de la forme du pointeur de la souris

Le fichier **/usr/include/X11/cursorfont.h** contient une liste des noms de formes de curseur disponibles pour modifier le curseur de différentes manières :

X–cursor	Le curseur prend la forme d'un X grand format.
arrow	Le curseur prend la forme d'une flèche dirigée vers la droite.
base_arrow_down	Le curseur prend la forme d'une flèche dirigée vers le bas.
base_arrow_up	Le curseur prend la forme d'une flèche dirigée vers le haut.
circle	Le curseur prend la forme d'un cercle.

Exécutez la commande suivante dans la fenêtre racine pour changer la couleur et la forme du pointeur de la souris :

```
xsetroot –cursor_name   name   –bg   color
```

où *name* désigne le nom de la forme de curseur et *color* la couleur sélectionnée.

Par exemple, pour changer la forme du curseur pour qu'il devienne un X grand format de couleur rouge, exécutez la commande suivante :

```
xsetroot –cursor_name X–cursor –bg red
```

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull AIX 5L Guide de l'utilisateur – Système d'exploitation et unités

Nº Référence / Reference Nº : 86 F2 44EM 01

Daté / Dated : Février 2005

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : _____ Date : _____

SOCIETE / COMPANY : _____

ADRESSE / ADDRESS : _____

_____

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

To order additional publications, please fill up a copy of this form and send it via mail to:
Pour commander des documents techniques, remplissez une copie de ce formulaire et envoyez-la à :

BULL CEDOC
ATTN / Mr. L. CHERUBIN
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone / Téléphone : +33 (0) 2 41 73 63 96
FAX / Télécopie +33 (0) 2 41 73 60 19
E–Mail / Courrier électronique : srv.Cedoc@franp.bull.fr

Or visit our web sites at: / Ou visitez nos sites web à :

http://www.logistics.bull.net/cedoc
`http://www-frec.bull.com` `http://www.bull.com`

CEDOC Reference # Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
[_ _] : **no revision number means latest revision** / pas de numéro de révision signifie révision la plus récente					

NOM / NAME : ____________________ Date : __________

SOCIETE / COMPANY : ____________________

ADRESSE / ADDRESS : ____________________

____

TELEPHONE / PHONE : ____________________ FAX : ____________________

E–MAIL : ____________________

For Bull Subsidiaries / Pour les Filiales Bull :
Identification: ____________________

For Bull Affiliated Customers / Pour les Clients Affiliés Bull :
Customer Code / Code Client : ____________________

For Bull Internal Customers / Pour les Clients Internes Bull :
Budgetary Section / Section Budgétaire : ____________________

For Others / Pour les Autres :
Please ask your Bull representative. / Merci de demander à votre contact Bull.

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
86 F2 44EM 01

Utiliser les marques de découpe pour obtenir les étiquettes.
Use the cut marks to get the labels.

AIX
AIX 5L Guide de l'utilisateur Système d'exploitation et unités
86 F2 44EM 01

AIX
AIX 5L Guide de l'utilisateur Système d'exploitation et unités
86 F2 44EM 01

AIX
AIX 5L Guide de l'utilisateur Système d'exploitation et unités
86 F2 44EM 01