AIX Questions

Compiled by Jeff Simon

The AIX Solution Provider Technical Support Group in Austin, Texas, supports software vendors who are developing or porting applications to AIX. This article is a compilation of questions that are frequently asked by vendors. The name of the responding Technical Support Group staff member appears after each response.

How can I determine which shell I am using?

One option, `chsh`, which is shipped with `bos.rte.security`, shows the shells installed on your system and provides an option to switch to another shell. Alternately, if your shell has not changed since the initial login, you can use the following:

```
grep <userid> /etc/passwd
```

which provides output similar to

```
jsimon:!:200:1::/u/jsimon:/usr/bin/ksh
```

—Jeff Simon

Which AIX® signals are thread-specific and use signal handlers?

The following synchronous signals are thread-specific and include signal handlers: `SIGILL`, `SIGTRAP`, `SIGARBT`, `SIGEMT`, `SIGFPE`, `SIGBUS`, `SIGSYS`, and `SIGSEGV`.

—Jeff Simon

Can pthread_exit() be called from within a signal handler?

A signal handler cannot call any `pthread` primitive, including `pthread_exit`.

—Jeff Simon

Are sigsetjmp() and siglongjmp() thread safe?

Yes, `sigsetjmp` and `siglongjmp` are signal- and thread-safe.

—Jeff Simon

Jeff Simon

How can I identify a 64-bit enabled RS/6000®?

If the following line appears in `/etc/inittab`, you are running on a true 64-bit enabled machine:

```
load64bit:2:once:/etc/methods/cfg64
```

—Wade Carlin

Are there any known performance problems using Java™ 1.1.4 with BOS 4.3.1?

It is necessary to apply APAR IX77074 to achieve top performance from the Java Development Kit (JDK) 1.1.4 on AIX 4.3.1.

—Jeff Simon

Which system calls are available to all kernel extensions?

Figure 1 shows a list of the system calls and describes the function of each. For additional information, see *AIX Kernel Extensions and Device Support Programming Concepts* (SC23-2611) or InfoExplorer™.

—Wade Carlin

How do I make AIX system calls from a Java program using Java Native Interface (JNI)?

The code example in Figures 2 through 5 demonstrates this procedure and includes the following filesets:

- Makefile
- JavaCrypt.c
- JavaCrypt.java
- JavaCrypt.exp

—Hung Dinh and David Carew

System Call	Function
gethostid	Gets the unique identifier of the current host
getpgrp	Gets the process ID, process group ID, and parent process ID
getppid	Gets the process ID, process group ID, and parent process ID
getpri	Returns the scheduling priority of a process
getpriority	Gets or sets the nice value
semget	Gets a set of semaphores
seteuid	Sets the process user IDs
setgid	Sets the process group IDs
sethostid	Sets the unique identifier of the current host
setpgid	Sets the process group IDs
setpgrp	Sets the process group IDs
setpri	Sets a process scheduling priority to a constant value
setpriority	Gets or sets the nice value
setreuid	Sets the process user IDs
setsid	Creates a session and sets the process group ID
setuid	Sets the process user IDs
ulimit	Sets and gets user limits
umask	Sets and gets the value of the file creation mask

Figure 1. System calls for kernel extensions

```
/* Makefile */

# The JAVA_HOME variable should be set to the installation directory for
# your AIX JDK.
#
# note: The JAVA_HOME variable will need to point to the current
# JDK directory.
```

Figure 2. Makefile (continued)

Wade Carlin

```
JAVA_HOME=/usr/lpp/J1.1.1

libcrypt.so: JavaCrypt.o
        rm -f libcrypt.so
        ld -o libcrypt.so JavaCrypt.o -bnoentry -bM:SRE -bE:JavaCrypt.exp \
           -blibpath:/lib:/usr/lib -lc_r \
           -L${JAVA_HOME}/lib/aix/native_threads -ljava

JavaCrypt.class: JavaCrypt.java
        ${JAVA_HOME}/bin/javac JavaCrypt.java
        ${JAVA_HOME}/bin/javah -jni JavaCrypt

JavaCrypt.o: JavaCrypt.c JavaCrypt.class
        xlc_r -c -I. -I${JAVA_HOME}/include -I${JAVA_HOME}/include/aix \
              -o JavaCrypt.o JavaCrypt.c

clean:
        rm -f *.class *.o JavaCrypt.h libcrypt.so
```

Figure 2. Makefile

David Carew

Hung Dinh

```
/* JavaCrypt.c */

/*
This program demonstrates the ability to make an AIX system call (i.e., crypt)
from within a Java application.
*/

#include “JavaCrypt.h” /* Include file generated by javah command */

JNIEXPORT jstring JNICALL Java_JavaCrypt_getEncryptedPassword
  (JNIEnv *env, jobject obj, jstring jpassword, jstring jsalt)
{
  const char *password;   /* String to be encrypted */
  const char *salt;       /* The 2 character salt (see crypt) documentation */
                          /* for details                                    */
  jstring encrypted;

  /* convert Java strings to C strings */
  password = (*env)->GetStringUTFChars(env, jpassword, 0);
  salt = (*env)->GetStringUTFChars(env, jsalt, 0);

  /* Encrypt the string */
  encrypted =  (*env)->NewStringUTF(env, (const char *) crypt(password, salt));

  /* tell Java we've finished with the strings */
  (*env)->ReleaseStringUTFChars(env, jsalt, salt);
  (*env)->ReleaseStringUTFChars(env, jpassword, password);

  /* Return results */
  return encrypted;

}
```

Figure 3. JavaCrypt.c

```
/* JavaCrypt.java */

/*
This class uses the JNI interface to call the AIX routine crypt. This technique
can be utilized to call any AIX system call from within a Java application.

Use the provided Makefile to compile this program.

It should be run as follows:

          java JavaCrypt “anystring”

Where “anystring” is the string to encrypt (enclosed in quotes if it contains
any whitespace characters).


*/

/**
  * This class uses the native crypt subroutine to encrypt a string.
  *
  * @version 1.0
  *
  */

public class JavaCrypt {

    /**
     * This constructor loads the library containing the native code
     *
     */

    public JavaCrypt() {
       System.loadLibrary(“crypt”);
    }

    /**
     * Calls the native routine to encrypt a string.
     *
     * @param String password - The string to be encrypted
     * @param String salt     - The salt used in the encryption.
     *                          Consult the crypt documentation
     *                          for more information about this parameter.
     *
     * @returns String - The encrypted string
     */

    public native String getEncryptedPassword(String password, String salt);

    /*
     * Used to test the JavaCrypt class
     *
     */
    public static void main (String [] args) {
        if (args.length != 1) {
           System.out.println(“Usage: java JavaCrypt \”String to encrypt\””);
           System.exit(1);
        }
```

Figure 4. JavaCrypt.java (continued)

```
        System.out.println("Encrypted string = " +
                           new JavaCrypt().getEncryptedPassword(args[0],
"dc"));
    }

}
```

Figure 4. JavaCrypt.java

```
/* JavaCrypt.exp */

Java_JavaCrypt_getEncryptedPassword


/* This program demonstrates the uses of the wait() system call and the
   WIF macros to determine the status from the child process  */

#include <sys/wait.h>
#include <stdlib.h>
main()
{
 int i,pid,status;
 pid = fork();
 if (pid == -1)
  {
   perror (" problems with fork child process ");
   exit(1);
  }
 if (pid ==0)   /* successfully forked  - child starts */
  {
  printf("child  does li -al    \n");
  execlp("li", "li", "-al",NULL);

 /*  The execlp subroutine searches each of the directories
     listed in the PATH environment variable for the 'li' command,
     and then it overlays the current process image with this command.
     The execlp subroutine is not returned unless 'li' command cannot
     be executed. */

  perror("  li command not found    \n");
  exit(2);
  }            /*  child finishes   */


  printf("parent does some tasks   \n");
  for (i=0;i<4;i++)
    {
     printf("parent sleeps for 1 sec \n");
     sleep(1);
     printf("parent wakes up \n");
    }

  printf("parent finishes the task   \n");
  printf("and child status is \n");
  wait(&status);
```

Figure 5. JavaCrypt.exp (continued)

```
/*
 The wait subroutine suspends the calling thread until the
 process receives a signal that is not blocked or ignored,
 or until any one of the calling process' child processes
 stops or terminates. The wait subroutine returns without
 waiting if the child process that has not been waited for
 has already stopped or terminated prior to the call.

 If the wait subroutine is unsuccessful, a value of -1 is
 returned and the errno global variable is set to indicate
 the error. In addition, the waitpid and wait3 subroutines
 return a value of 0 if there are no stopped or exited child
 processes, and the WNOHANG option was specified. The wait
 subroutine returns a 0 if there are no stopped or exited
 child processes, also.

*/

/*

 The value pointed to by StatusLocation when wait subroutines
 are returned, can be used as the ReturnedValue parameter for
 the following macros defined in the sys/wait.h file to get
 more information about the process and its child process.

 WIFSTOPPED
 Returns a nonzero value if status returned for a stopped child.

 WIFEXITED
 Returns a nonzero value if status returned for normal termination.

 WIFSIGNALED
 Returns a nonzero value if status returned for abnormal termination.

*/

if WIFSTOPPED(status)
  printf(" child stopped and stop sig was %d \n", WSTOPSIG(status));
else if WIFSIGNALED(status)
  printf("child was terminated and signal was %d\n", WTERMSIG(status));
else if WIFEXITED(status)
  printf("child ended ok and status was %d\n", WEXITSTATUS(status));

exit(0);

}
Describe mail filters on AIX.
```

Figure 5. JavaCrypt.exp

Describe mail filters on AIX.

Figure 6 demonstrates the use of mail filters on AIX. Figure 7 shows the message generated by the mail filter.

—David Carew and Hung Dinh


```
/*

 The program demonstrates the use of mail filters on AIX.


 To compile use the following:

    cc -o mailf mailf.c

 The program can be triggered by adding the following changes to the .forward
 file in your home directory.

    username, | mailf username

Where username is your login ID. The executable file should be in a directory
like /bin that is in the default path. Alternatively, you can add the fully
qualified path name to the executable file name in the .forward file.

note: A message file (mailf.msg) must be installed in the user's home
      directory (see below).

*/

#include <string.h>
#include <sys/errno.h>
#include <pwd.h>
#include <stdio.h>

extern struct passwd *_getpwnam_shadow(char *, int);

/*
**  MAILF — return a message to the sender of a mail message.
**
*/

main(int argc, char **argv)

{
   char msgf[128];
   char *username;
   char *sender;
   struct passwd *pw;
   char *getsender();

   /* Make sure the username is specified */
   if (argc != 2) {
               fprintf(stderr, "Unknown user %s", username);
               exit(EINVAL);
   }

   /* Get the username of the user who has installed the filter */
```

Figure 6. Mail filters on AIX (continued)

```
     username = argv[argc - 1];

         /* find user's home directory */
         pw = _getpwnam_shadow(username, 0);
         if (pw == NULL)
         {
                 fprintf(stderr, "Unknown user %s", username);

                exit(EINVAL);
         }

    /* Construct message file name */
         strcpy(msgf, pw->pw_dir);

         (void) strcat(msgf, "/mailf.msg");

    /* Get the sender of the mail */
    sender = getsender();

    /* Send reply to sender */
    sendreply(msgf, sender, username);
}


/*
**  GETSENDER -- read message from standard input and return sender
**
**      Parameters:
**              none.
**
**      Returns:
**              pointer to the sender address.
**
**      Side Effects:
**              Reads first line from standard input.
*/

char *
getsender()
{
        static char line[512];
        char *p;

        /* read for the line */
        if (fgets(line, sizeof line, stdin) == NULL ||
            strncmp(line, "From ", 5) != (int)NULL)
        {
                fprintf(stderr,"Invalid message format");
                exit(EINVAL); /* Invalid message format */
        }

        /* Parse out sender info and return it */

        p = strstr(&line[5], " ");
        if (p == NULL)
        {
           fprintf(stderr,"Invalid message format");
           exit(EINVAL); /* Invalid message format */
}
```

Figure 6. Mail filters on AIX (continued)

```
        *p = '\0';

        /* return the sender address */
        return (&line[5]);
}

/*
**  SENDREPLY - send a message to a particular user.
**
**      Parameters:
**              msgf - file name containing the message.
**              user - user who should receive it.
**    myname - sender of message
**
**      Returns:
**              none.
**
**      Side Effects:
**              sends mail to 'user' using /usr/sbin/sendmail.
*/
int sendreply(msgf, sender, user)
        char *msgf;
        char *user;
        char *sender;
{
        FILE *f;
        char newaddr[128];
   char *to;

   /* Verify that the destination address is in correct format */
   /* i.e., in the form <joe@mycompany.com>           */
   /* Fix it if it isn't.                             */
        if (sender[0] == '<' && sender[strlen(sender) - 1] == '>')
                to = sender;
        else {
                newaddr[0] = '<';
      newaddr[1] = '\0';
                strcat(newaddr,sender);
                strcat(newaddr, ">");
                to = newaddr;
        }

   /* find the message to send and replace stdin with this file */
   /* sendmail will read input of message from stdin            */
        f = freopen(msgf, "r", stdin);
        if (f == NULL)
        {
           fprintf(stderr, "No message to send");
      exit(EINVAL);
        }

   /* Execute sendmail */
        execl("/usr/sbin/sendmail", "sendmail", "-f", user, sender, NULL);

   /* If we get here then execl failed */
        fprintf(stderr, "Cannot exec /usr/sbin/sendmail") ;
   exit(ENOEXEC);
}
```

Figure 6. Mail filters on AIX (continued)

```
/* mailf.msg
**
** This file contains the replied message generated
** by the mail filter.
**
** The accompanying message file, mailf.msg, must be installed in the user's
** home directory. This file should be edited to include the correct e-mail
** address of the sender and the appropriate message.
**
*/

From: joe@mycompany.com (Joe User)
Subject:  Mail filter test

SYSTEM GENERATED MESSAGE - DO NOT REPLY -

Enter your message
```

Figure 7. Message generated by the mail filter

Compiled by Jeff Simon, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Simon has worked in technical support since 1990 and currently works as a technical lead and systems administrator for a series of technical Web sites. He has a BS in Computer Science from Southwest Texas State University.