AIX Questions

Compiled by Bruce Pine

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 do I install a nodelock license?

As root, cut and paste the following:

```
 <vendor_id> <product_password>
"<annotation>" "<version>"
```

into `/usr/lib/netls/conf/nodelock`. Here is the result of that procedure:

```
5da54a553b4c.02.09.15.31.05.00.00.00
ftzznpe9hfgqw "2" "3"
```

It may be helpful to place a comment above your license indicating expiration date/compiler; for example:

```
## C Set ++ for AIX version 3. Good from
<date> until <date>
```

Next, verify that your system's date is correct.

—Jeff Simon

How can I recover from a NetLS problem?

Here are the steps for NetLS recovery:

1. `lssrc -g ncs` shows which daemons are running (`llbd`, `glbd`, `nrglbd`)

2. Stop the daemons: `stopsrc -s netlsd`, `stopsrc -s glbd`, and `stopsrc -s llbd`

3. Remove the files shown in Figure 1.

4. Run the following:

```
/etc/ncs/lb_admin
lb_admin: use local
lb_admin: clean (y to delete bad entries)
lb_admin: use global
lb_admin: clean (y to delete bad entries)
lb_admin: quit
```

If no errors occur, then the brokers are in sync. If there are other license servers in the same cell, also execute the following:

```
     /tmp/llbdbase.dat
     /usr/lib/netls/conf/cur_db      **!
/usr/lib/netls/conf/lic_db        !-  Make a backup copy of these files, since they
/usr/lib/netls/conf/lic_db.bak  **!   contain your password. Hence, you will need to
                                      restore them or get new passwords?
/etc/ncs/glb.e     ***!--These are your 2 GLB data bases
/etc/ncs/glb.p     ***!
/etc/ncs/glb_log
/etc/ncs/glb_obj.txt
/etc/ncs/glb_site.txt  ***! This identifies a specific server (usually a faster
                           machine) in your cell that you want your password from.
/usr/lib/netls/conf/log_file
```

Figure 1. Files removed for NetLS recovery

Jeff Simon

```
# drm_admin
drm_admin: set -o glb -h ip:<hostname>
drm_admin: merge_all
drm_admin: quit
```

This causes all of the other Global Location Broker Daemons (`glbd`) in the cell to update their location databases to reflect the changes that were made.

5. Remove `netls_first_time`. Run `/usr/lib/netls/conf/netls_config`, as shown in Figure 2.

Do you want the `llbd` started automatically when the machine boots?	yes
Do you want `netlsd` started automatically when the machine boots?	yes
An initialized database already exists for the `glbd`. Do you wish to use that database when starting the `glbd` daemon?	no
1. Continue with installation without choosing a `Cell_Name`.	
2. Use the default for the system Cell Name.	
3. Create a new alternate cell for the system Cell Name.	
Please indicate your choice (1, 2, 3):	2
Daemons started automatically?	yes

Figure 2. NetLS configuration

6. Run `/usr/lib/netls/conf/netls_first_time`. This should restart the three daemons—`glbd`, `llbd`, and `netlsd` in sync.

—Jeff Simon

How can I call a C program from a FORTRAN program?

Figure 3 shows an example.

—Jeff Simon

How can I change the number of processes assigned per user?

As root, select the options shown in Figure 4 to change the number of processes assigned per user. This can also be done from the command line:

```
chdev -E -l sys0 -a maxuproc=n
```

where `n` is a number between 40 and 10,000.

—Jeff Simon

```
c program name: FortranCallingC.f
      write (6,*) "FORTRAN is not just for mathmeticians"
      call hello
      end


/* program name: Cprog.c */
void hello()
{
  printf("Most of UNIX is written in  C\n");
}

syntax of compilation:  xlc -c Cprog.c
                        xlf FortranCallingC.f Cprog.o -o <filename>
```

Figure 3. Calling a C program from a FORTRAN program

```
smitty
  System Environments
    Change / Show Characteristics of Operating System
      Maximum number of PROCESSES allowed per user
```

Figure 4. Changing the number of processes per user

Is it possible to distinguish the setting for maxdata with an executable?

Maxdata is a binder option that is set at link time; that is, `-bmaxdata:0xy0000000` where y > 2. This information can be obtained from an executable by entering the following:

```
dump -ov <executable>
```

A section called Optional Header will contain this information:

```
maxSTACK      maxDATA
0x00000000  0x50000000
```

—Jeff Simon

How do I print the scope of a class from within DBX?

Use the `which` command from within DBX; for example, `which <class_name>`

—Jeff Simon

How do I print the type of a class from within DBX?

Use the `whereis` command from within DBX; for example, `whereis <function>`

—Jeff Simon

How do I print a variable assignment inside a class?

Here are the steps to print a variable assignment of a class:

1. Compile program using `-g`, which inserts debug information into the symbol table

2. Step through code until the variable needed has been "seen"

3. Print `<variable_name>`

Note: Many DBX subcommands contain a "fastpath" by substituting the DBX subcommand with the first letter of the command; that is, `p <variable_name>`.

—Jeff Simon

How can I do a fast IPL on an SMP system?

David Stewart

Either enter the following command in `inittab` or run it from the command line:

```
mpcfg -cf 11 1    (command line)
```

A fast IPL can also be set in the Standby menu. Follow the steps described below (see the Service Guide for the R30 for more information):

1. Place the key mode switch to service mode; press the enter key on a terminal connected to the S1 or S2 line.

2. The BUMP clears the screen on the selected line, issues a prompt, and waits for a keyword.

3. Enter the keyword `sbb` to display the Standby menu. The Standby menu consists of a main menu with several options as shown below:

   0 Display Configuration
   1 Set Flags
   2 Set Unit Number
   3 Set Configuration
   4 SSbus Maintenance
   5 I2C Maintenance

Choose option 1. This will give you some flags to toggle ON or toggle OFF. Figure 5 shows option 1.

—David Stewart

Remote Authorization flag	Disabled or Enabled
BUMP Console flag	Disabled or Enabled
Autoservice IPL flag	Disabled or Enabled
Extended Tests parameter	Disabled or Enabled
Power-On Tests in Trace Mode flag	Disabled or Enabled
Power-On Tests in Loop Mode flag	Disabled or Enabled
Fast IPL flag	Disabled or Enabled
Set Electronic Mode Switch to Normal	Disabled or Enabled
Select x to exit once the flags are set.	

Figure 5. Standby menu—Option 1

How can I identify what shared libraries an executable file uses?

Use the `dump -H` command on the executable file. Figure 6 shows an example.

The Import File Strings section shows the library path searched for the shared libraries and the shared libraries that are actually loaded (in this case, `libXt.a`, `libXext.a`, `libX11.a`, and `libc.a`).

—Bill Woodward

How can I determine what shared libraries and dynamically loadable code are being used by a running process?

Since some executables (such as the X server) dynamically load executable code, the `dump -H` command may not show all of the code dependencies. In this case, the `genld` command can be used. The `genld` command is part of the `perfagent.tools` Licensed Program Product (LPP).

To run `genld`, just type `genld` and redirect the output to a file. Because this prints information for all of the system processes, you will need to edit the file and search for the process name or process id.

For example, `genld` shows that X server is using the files shown in Figure 7.

Figure 7 shows that the X server has dynamically loaded the XVideo extension (`/usr/lpp/UMS/lib/loadxv/`), and has also dynamically loaded the device-specific code for the GXT500 (`/usr/lpp/gai/60x00004002/loadddx/`) as well as the other libraries shown by `genld`.

—Bill Woodward

```
# dump -H /usr/bin/X11/bitmap

/usr/bin/X11/bitmap:

                        ***Loader Section***
                     Loader Header Information
VERSION#         #SYMtableENT     #RELOCent        LENidSTR
0x00000001       0x000000e4       0x00000b52       0x00000061

#IMPfilID        OFFidSTR         LENstrTBL        OFFstrTBL
0x00000005       0x00009d58       0x00000c65       0x00009db9

                        ***Import File Strings***
INDEX  PATH                          BASE               MEMBER
0      /usr/lib:/lib:/usr/lpp/xlC/lib
1                                    libXt.a            shr4.o
2                                    libc.a             shr.o
3                                    libX11.a           shr4.o
4                                    libXext.a          shr.o
```

Figure 6. dump -H example

```
Proc_pid:   6036   Proc_name: X
                    d04f3000  /usr/lpp/UMS/lib/loadxv/
                    d04ef000  /usr/lpp/gai/60x00004002/loadrms/
                    d04ec0a8  /usr/lib/libgair4.a/shr.o
                    d0471000  /usr/lpp/gai/60x00004002/loadddx/
                    d012e0a8  /usr/lib/libodm.a/shr.o
                    d013e0a8  /usr/lib/libcfg.a/shr.o
                    d01c40a8  /usr/lib/libdbm.a/shr.o
                    d01266f0  /usr/lib/libc.a/meth.o
                    d0000380  /usr/lib/libc.a/shr.o
                    10000000  X
```

Figure 7. Files used by X server

How can I identify what workstation type is actually being used for a graPHIGS™ application with the workstation type specified as "X", meaning to use the best available workstation type?

Since the workstation-dependent code is dynamically loaded depending on the actual workstation type selected, you must use `genld` to generate the list of shared libraries and dynamically loaded code. Then use that information to determine the actual workstation type selected.

In a typical graPHIGS application, `genld` will show a large number of shared libraries and dynamically loaded code. However, only a few are necessary to determine the actual workstation type.

Example 1: The soft graPHIGS pipeline (`/usr/lib/libgppipe.a`) is in use, and the graPHIGS rasterizer is the soft rasterizer (`/usr/lib/librengp_soft.a`). The application is using the XSOFT workstation type. See Figure 8.

Example 2: The soft graPHIGS pipeline (`/usr/lib/libgppipe.a`) is in use, but we are using a device-specific rasterizer (`/usr/lib/librengppex_ppr.a`). The application is using the XDWA workstation type with the soft pipeline introduced in AIX 4.1.4. See Figure 9.

Example 3: Here we are using a device-specific `load3dm1` module. The `load3dm1` module is the interface to the graPHIGS code running on the graphics adapter. The application is using the XDWA workstation type that was used prior to AIX 4.1.4. See Figure 10.

```
Proc_pid:   21388   Proc_name: gptest
                        [deleted irrelevant libs]
                        d17ad0c0  /usr/lib/libgppipe.a/shr.o
                        d1e3df18  /usr/lib/librengp_soft.a/shr_r.o
                        d14300c0  /usr/lib/librengp_soft.a/shr.o
                        d0241000  /usr/lpp/graPHIGS/bin/loads3d/
                        [deleted irrelevant libs]
                        10000000  gptest
```

Figure 8. XSOFT workstation type

```
Proc_pid:   21434   Proc_name: gptest
                        [deleted irrelevant libs]
                        d25270c0  /usr/lib/librengppex_ppr.a/shr.o
                        d2526000  /usr/lpp/gai/adapter4.r4/GPrasterizer/
                        d17ad0c0  /usr/lib/libgppipe.a/shr.o
                        d2525000  /usr/lpp/gai/adapter4.r4/GPpipeline/
                        d2522000  /usr/lpp/gai/adapter4.r4/load3dm3/
                        d01ac000  /usr/lpp/gai/adapter4.r4/loadrms/
                        [deleted irrelevant libs]
                        10000000  gptest
```

Figure 9. XDWA workstation type

```
Proc_pid:   21442   Proc_name: gptest
                         [deleted irrelevant libs]
                         d25a9000  /usr/lpp/gai/adapter4.r4/load3dm1/
                         d01ac000  /usr/lpp/gai/adapter4.r4/loadrms/
                         [deleted irrelevant libs]
                         10000000  gptest
```

Figure 10. XDWA workstation type prior to AIX 4.1.4

```
Proc_pid:   21418   Proc_name: gptest
                        [deleted irrelevant libs]
                        d14250c0  /usr/lib/libXi.a/shr.o
                        d013a0c0  /usr/lib/libodm.a/shr.o
                        d01790c0  /usr/lib/libgair4.a/shr.o
                        d0e690c0  /usr/lib/libXext.a/shr.o
                        d0191c30  /usr/lib/libX11.a/shr4net.o
                        d018c0c0  /usr/lib/libIM.a/shr.o
                        d017c0c0  /usr/lib/libiconv.a/shr4.o
                        d02a90c0  /usr/lib/libX11.a/shr4.o
                        [deleted irrelevant libs]
                        10000000  gptest
```

Figure 11. XLIB workstation type

Example 4: None of the other conditions apply. The application is using the XLIB workstation type. See Figure 11.

In cases where graphics adapter-specific code is loaded, the entries may be slightly different. However, for post-AIX 4.1.4 graPHIGS, the device-specific rasterizer will be named `librengppex_xxx.a`, where the `xxx` is a three-letter code specific to the device. For the pre-AIX 4.1.4 code, the `load3dm1` will be consistent, although it may be loaded from a different directory.

—Bill Woodward

Bruce Pine, *IBM Corporation, 11400 Burnet Road, Austin, TX 78758.*