AIX Questions

Compiled by Charlie Rundles

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

Communications

How can I determine the current heap allocation for a loaded Streams module?

To get the data, use the commands `strmsg` and `strinfo` as a pair. The `strinfo` command stores the data in kernel buffers; `strmsg` extracts and displays stored data. To run these commands, start `strmsg` first so that it empties any data from the kernel buffers. Then issue the `strinfo` command with one or more of the following flags:

-h for heap information
-q for runq information
-m for module information

—Ismat Dhanjibhai

Charlie Rundles

My Distributed Computer Environment (DCE) application fails to link when using implicit binding. The following symbols, required by the client stub object file, fail to resolve:

```
rpc_binding_handle_t_unbind
rpc_binding_handle_t_bind
```

The contents of `idl` and `acf` files are shown in Figure 1.

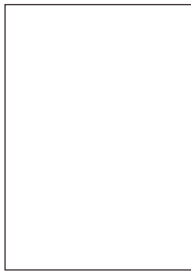
If the `acf` file is changed so that `handle_t` is used instead of `rpc_binding_handle_t`, the application links. I do not understand why the format in use fails since `handle_t` and `rpc_binding_handle_t` are the same thing.

The types `rpc_binding_handle_t` and `handle_t` are indeed the same. The difference is apparent only when compiling an `idl` file through the IDL compiler, which recognizes only `handle_t` as the primitive handle type.

A primitive handle contains the binding information needed to make Remote Procedure Calls (RPCs). Since a handle of any other type is assumed to be an application structure, the IDL compiler generates calls to `application_handle_type_[bind,unbind]` so that the application can supply the primitive handle for the runtime.

In this case, the IDL compiler generates calls to `rpc_binding_handle_t_[bind,unbind]` functions, expecting those to be supplied in another object file. Calls to these functions are generated under the assumption that an application structure exists and only the application can get the primitive structure from that structure.

—Ismat Dhanjibhai

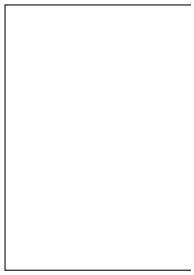


Ismat Dhanjibhai

I have been told to redirect (>) my `stdout` and `stderr` on the `rcnfs` entry in my `/etc/inittab` to make the `YPBIND` command work. Why does it not work normally?

In a system of 70 RISC System/6000s, most will continue to run NFS/NIS daemons from the start (`/etc/rc.nfs` via `/etc/inittab`) after rebooting.

On some systems, the NFS (NFSD and BIOD) and NIS (YPBIND) daemons will either not start or will not last until the login (as root user). However, on the same systems, the entry can be changed in `/etc/inittab` for `rcnfs` as shown in Figure 2, and the daemons will run normally.



Venkatesh M. S. Iye

```
idl file

   uuid(00219E04-5172-1CEB-961A-02608C2EF5AD),
pointer_default(ref)
   interface conftest
   {
     import "dce/cdltypes.idl";
     void conf1([in,out]    pic_9_9_comp_5
*running_value,
                [in]        pic_9_4_comp_5 *amount);
     void conf2([in,out]    pic_9_9_comp_5
*running_value,
                [in]        pic_9_4_comp_5 *amount);
   }

acf file

   [ /* implicit binding method */
     implicit_handle(rpc_binding_handle_t
global_binding_h)
```

Figure 1. Contents of idl and acf files

From:

```
rcnfs:2:wait:/etc/rc.nfs > /dev/console 2>&1 # Start
NFS Daemons
```

To:

```
rcnfs:2:wait:/etc/rc.nfs > <filename> 2>&1 # Start NFS
Daemons
```

Figure 2. Changing entry to rcnfs

The problem was that `/etc/inittab` had the `cons:0123456789:respawn:/etc/getty /dev/console` entry between `rctcpip:2:wait:/etc/rc.tcpip > /dev/console 2>&1` and `rcnfs:2:wait:/etc/rc.nfs > /dev/console 2>&1`. This presents competition between `/etc/getty` and NFS/NIS daemons trying to display on the master console when one is not available. NFS/NIS daemons must have a master console for displaying error messages or they will not continue to live.

—Rick Malone

When a Network Computing System (NCS) server is busy with an NCS client and a client is waiting, does the client send the data more than once? How frequently does a client send the data to the server? Is it possible when the client resends requests many times that the data size exceeds the number of `mbufs`?

Yes, when a client is waiting, it can send its data more than once. The mechanism is as follows:

When a client request times out, the client NCS runtime attempts to ping the server with a NULL RPC. If the ping succeeds, the client resends the data.

The ping's success or failure is determined differently for two separate cases: when normal (long) timeouts are set and when short timeouts are set. In the former case, the client sends Remote Procedure Calls (RPCs) to ping the server at one-second intervals up to 30 times. In the latter case, only one ping (with a one-second timeout) is sent. A client that times out and resends requests will not cause a server machine to run out of `mbufs` since the vast majority of the RPCs sent are "ping" RPCs, with only a header and no body (data).

—Venkatesh M. S. Iyer

How do you approximate the amount of physical disk space that will be required by a NetWare-for-AIX-`INODES` file?

The approximate physical disk space required by NetWare-for-AIX-INODES files is reported by `du -s`, which shows the actual amount of disk space in use. Since the `INODES` file is a sparse file, the size of the file reported by `ls -l` will be larger.

Figure 3 shows the formula first, followed by the technical background on how the formula was derived. This information is based on a Version 3.11A `INODES` file.

This solution has the following caveats:

1. The formula yields an approximation only; use it as a lower boundary and as a guideline.
2. Always round up.
3. For a single directory with more than 133 entries, this formula will yield a value slightly smaller than actual. It should, however, be usable within the constraints of caveat 1.

—Rich Lewis and Jeff Barrett

Rich Lewis

```
ESTIMATION OF DISK SPACE USED BY NETWARE FOR AIX INODES FILE
============================================================
NUM_INODE_ENTRIES = NUMBER__OF_FILES + NUMBER__OF_DIRECTORIES
NUM_DIRBLOCK_ENTRIES = NUMBER_OF_FILES + 3 * NUMBER_OF_DIRECTORIES
NUM_SEGMENTS = MAX( NUM_INODE_ENTRIES / 15, NUM_DIRBLOCK_ENTRIES / 225 ) + 1
DISK_SPACE_USED = NUM_SEGMENTS * 25.64 KB

Example:
This example is from the /opt/netware/sys as installed with Version 3.11 and then updated with the
Version 3.11A PTF.

Number of files = 545  (via find ./ -type f -print | wc -l)
Number of directories = 22  (via find ./ -type d -print | wc -l)

INODE_ENTRIES = 545 + 22 = 567
DIRBLOCK_ENTRIES = 545 + 3 * 22 = 611
NUM_SEGMENTS = MAX( 567 / 15, 611 / 225 ) + 1 = MAX( 38, 3 ) + 1 = 39

==>  DISK_SPACE = 39 * 25.64 KB = 999.96 KB

Note: The actual value reported via du -sk inodes is 1,000 KB.

Derivation of Formula:
For more information on the structure of the INODES file, see NetWare-for-AIX Version 3.11A file
/usr/lpp/netware/README. The nwfsck section of the Utility Addendum discusses the format of the Version
3.11A INODES file.

The INODE file consists of a FILE SYSTEM HEADER, followed by one or more SEGMENTS. Each SEGMENT
contains 16 file system blocks: 1 INODE Block and 15 Directory Blocks (DIRBLOCKs.

The FILE SYSTEM HEADER is the same size as a SEGMENT.

A SEGMENT can also contain Indirect blocks. Indirect Blocks are used only if there are more than 133
entries in a single directory. Indirect Blocks are beyond the scope of this document.

The INODE Blocks:
An INODE Block can contain up to 15 INODE entries.
Each SEGMENT has One INODE Block; therefore, a single SEGMENT can contain 15 * 1 = 15 entries.
There is one INODE Block entry for each FILE and one INODE Block entry DIRECTORY in the volume.

The number of SEGMENTs required for INODE entries is:
      (Number of Files + Number of Directories)/15

The Directory Blocks:
Each DIRBLOCK element can contain 15 entries.
Each SEGMENT contains 15 DIRBLOCK elements. Thus, a single SEGMENT can have 15 * 15 = 225 entries.
There is one DIRBLOCK entry for each file and 3 DIRBLOCK entries for each directory (the actual
directory, ".", and ".." directories) in the volume.

The number of SEGMENTs required for DIRBLOCK entries is:
      (Number of Files + 3 * Number of Directories ) / 225

Number of SEGMENTs:
Each SEGMENT contains 1 INODE Block and 15 DIRBLOCKs.
When more space is required—another INODE Block or DIRBLOCK—a new SEGMENT is allocated (1 INODE Block
and 15 DIRBLOCKs).
Since the FILE SYSTEM HEADER is the same size as one SEGMENT, the number of SEGMENTs required for a
volume is:
      Maximum (Number of segments required by INODES, Number of SEGMENTs required by DIRBLOCKs) + 1

Disk Space Estimation:
```

Figure 3. Method for estimating disk space used by NetWare-for-AIX-INODES file

```
# -- This script assumes that the format of the 'cup'
attribute is
# -- consistent with the form used by 'aixterm'. Terminal
types
# -- 'hft' and 'xterm' are examples of terminal
definitions using the
# -- same format.
# --

function place_cursor {  # open brace
PLACE=$(echo $tmp | sed -e "s/%i%p1%d/$1/" -e
"s/%p2%d/$2/")
echo "$PLACE Here we are at $1,$2\c"
sleep 1
}  # close brace

export tmp=$(tput cup)

tput clear
```

Figure 4. Program to emulate tput cup

Graphics

Does AIX support `tput cup` with x,y coordinates for cursor positioning?

Jeff Simon

The System 5 Interface Definition manual states that the future direction for `terminfo` stanzas is to support giving arguments to the capabilities. The 5.2 code currently being used does not support the future direction; that is, passing arguments to the `cup` capability is not supported. Using `tput cup` without any arguments produces some nonsense on the screen, such as putting the cursor in the same position as previously located. The current implementation does not support passing arguments to `terminfo` capabilities.

AIX code is AT&T® 5.2-based, which cannot support parameter passing to the capability specified, in this case, `cup`. Parameter passing was added in AT&T 5.3. A DCR was generated on this and should be implemented in a later release.

In the meantime, the program in Figure 4 (which is not supported) will emulate `tput cup`. The only restriction is the terminal type. (Thanks to Jeff Rampy for his assistance in writing the program, based on AT&T's 5.2.)

—Jeff Simon

When `xinit` is migrated to X11R5 and Motif 1.2, it crashes after opening the screen.

Sai P. Ramanath

Check the `LIBPATH`. If it is set to `/usr/lpp/X11/lib:/usr/lib:/lib:`, change it to blank.

With X11R5, `xinit` becomes an executable rather than a shell script. If the `LIBPATH` was set as above, then the X libraries were still pointed to old directories. These directories are provided for convenience to those applications that cannot be run on X11R5 or require large changes.

Since `xinit` was a shell script in old X, it crashes in the new environment.

—Sai P. Ramanath

AIX

I have two RISC System/6000 Model 7013s with two disks mirrored. `Hdisk0`, with two descriptors, is being mirrored on `hdisk1`, which has one descriptor. When booting up on the disk with one descriptor, I get an LED of 552.

Priyamvada

You are using two-way mirroring for `rootvg` that can help recovery only if the mirrored copy fails. In this case, where `hdisk0` (with two `vgdas`) fails, the system will not vary on because there is no quorum. You could use three-way mirroring with three disks, each having a `vgda`. This would enable recovery regardless of which disk fails. Of course, if two disks go offline at the same time, quorum is again lost.

You should also maintain all of your data on a separate volume group, not `rootvg`, because this situation holds only when `rootvg` is mirrored. If one disk that belongs to the data volume groups fails, the system would still boot successfully, and recovery procedures could then be performed.

—Priyamvada

***Charlie Rundles**, IBM Corporation, RISC System/6000 Division, 11600 Burnet Road, Austin, TX 78758. Internet: rundles@austin.ibm.com. Mr. Rundles currently provides technical support to software vendors at the AIX Solution Provider Technical Support group. Before joining IBM, he worked on SS7 for ISDN communications. He holds an MA in Theoretical Math-*