37

AIX

November 1998

In this issue

update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 550955
From USA: 01144 1635 33823
E-mail: HarryLewis@compuserve.com

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions

If you have anything original to say about AIX, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you actively be helping the free exchange of information, which benefits all AIX users, but you will also gain professional recognition for your expertise and that of your colleagues, as well as being paid a publication fee – Xephon pays at the rate of £170 ($250) per 1000 words for original material published in AIX Update.

To find out more about contributing an article, see *Notes for contributors* on Xephon's Web site, where you can download *Notes for contributors* in either text form or as an Adobe Acrobat file.

Editor

Harold Lewis

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, scripts, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1995 issue, are available separately to subscribers for £15.00 ($22.50) each including postage.

AIX Update on-line

Code from *AIX Update* is available from Xephon's Web page at www.xephon.com (you'll need the user-id shown on your address label to access it).

Printed in England.

Rescheduling jobs

Sometimes it's useful to be able to adjust the time when a job that's already been submitted executes. For instance, problems with system performance or unplanned, urgent maintenance may require us to reschedule jobs to a later time or move them forward.

Using the command **at -l** we can display the jobs waiting for execution. Running this command yields screen output similar to that below.

```
at -l

root.899244600.a        Tue Jun 30 23:10:00 1998
root.899236724.a        Tue Jun 30 20:58:44 1998
```

You can find the actual jobs submitted by users in the directory */var/spool/cron/atjobs*. Let us look at this directory listing.

```
ls -la

-r-Sr-Sr--  1 root  system  1410 Jun 30 17:58 root.899236724.a
-r-Sr-Sr--  1 root  system  1424 Jun 30 13:19 root.899244600.a
```

THE SOLUTION

Using the script **ch_at** you can modify the execution time of any submitted job.

THE INPUT

- *Jobnumber*
  For instance *root.18923841*. This identifies the job in the **at -l** listing. To look at a job's contents, use **ls_at *jobnumber***. If you want to select the job interactively, use the **-i** switch.
- *Operator*
  + Moves execution forwards in time.
  – Moves execution backwards in time.
- *Time*
  hhh-mm-ss is the amount of time to move the job.

USAGE

The command below moves job *root.2849849024* back 10 hours and five minutes.

```
ch_at root.2849849024 + 10-5-0
```

OUTPUT

The new execution time of the job.

CH_AT

```
#!/usr/bin/ksh
# ------------------------------------------------------+
# scriptname: ch_at                                     !
# Author    : Michael Imhotep                           !
# Ojectives : Change the execution time of a scheduled  !
#             job forwards or backwards.                !
#                                                       !
# Usage :     ch_at [jobname|-i] [+|-] hh-mm-ss         !
# ------------------------------------------------------+
#
integer _hours
integer _time
integer _minutes
integer _seconds
integer _param=""
integer _new_time
atname_path=/var/spool/cron/atjobs
typeset -lu YESNO=n
#
# ++++++++++++++++++++++++++
# Define usefull functions  !
# ++++++++++++++++++++++++++
#
display_usage ()
{
echo you chose a wrong parameter or combination
case $_param in
 1) echo "1. parameter should be -i for interactive "
    echo "   selection or a jobname like root.899255692.a"
    ;;
 2) echo "2. parameter should be a move operator like"
    echo "   + to move the job in the future or a"
    echo "   - to move the job more closer "
    ;;
 3) echo "3. parameter should be the move time specified"
```

```
    echo "   with hours minutes and seconds delimited with a - dash"
    echo "   E.g  2-10-3 to move the specified job 2 hours 10 minutes"
    echo "   and 3 seconds "
    ;;
 4) echo "usage of ch_at is"
    echo "ch_at [-i|jobname] [+|-] hh-mm-ss "
    ;;
 *) echo "wrong option of display_usage"
    ;;
esac
}
move_job ()
{
 _user=${jobname%%.*}     # cut stamp.a from user.stamp.a
 _prefix=${jobname##*.}   # cut user.stamp  from user.stamp.a
 work=${jobname%.[abc]}   # cut .a .b or .c from user.stamp.a
 _actual_time=${work#*.*} # cut user. from user.stamp
 _new_time=${_actual_time}${operator}${_time}
 new_job=${_user}"."${_new_time}"."${_prefix} # user.newtime.a
 #
 # check f the planned modification leads to any conflict
 #
 if test -f ${atname_path}/${new_job}
 then
    echo The atjob ${new_job} allready exists current
    echo job not moved - use atrm ${new_job} to remove
    echo the old at-job first
    return 97
 fi
 mv ${atname_path}/${jobname} ${atname_path}/${new_job}
 rc_code=$?
 if test $rc_code -ne 0
 then
    echo something went wrong $rc_code returned from mv
    return 96
 else
    new_proc_time=`at -l ${new_job} |  cut -c18-36 `
    echo $1 successfully moved to ${new_job} new processing time
$new_proc_time
    return 0
 fi
}
# ++++++++++++++++++++++++
# Start main processing  !
# ++++++++++++++++++++++++
#
# 1. step check parameters
#
if test $# -lt 3
then
```

```
  _param=4
  display_usage
  exit 99
fi
#
# 2. check the sense of the parameters
#    parameter should be -i or username.timestamp.a
#
if [[ $1 != @(-i|+([0-9|a-z|A-Z]).+([0-9]).+([a-c])) ]]
then
  _param=1
  display_usage
  exit 1
else
 jobname=${1}
fi
#
#    parameter should be + or -
#
if [[ $2 != @([+-]) ]]
then
   _param=2
   display_usage
   exit 2
else
   operator=${2}
fi
#
#    parameter should be in the form hh-mm-ss
#
if [[ $3 !=  @(+([0-9])-+([0-9])-+([0-9])) ]]
then
   _param=3
   display_usage
   exit 3
else
   hh_mm_ss=${3}
fi
# ++++++++++++++++++++++++++++++++++++++++++++
# input checked - now compute the move time  !
# ++++++++++++++++++++++++++++++++++++++++++++
#
#
hh_mm=${hh_mm_ss%-*}       # should cut -ss from hh-mm-ss
                           # -> result hhh-mm
_minutes=${hh_mm#*-}       # should cut hh- from hh_mm
                           # -> result mm
_hours=${hh_mm_ss%%-*-*}   # should cut -mm-ss from $hh_mm_ss
                           # -> result hh
_seconds=${hh_mm_ss##*-*-} # should cut hh-mm- from $hh_mm_ss
```

```
                              # -> result ss
_time=${_hours}*60*60+${_minutes}*60+${_seconds} # move time
#
case $1 in
-i)
 # interactive processing
 at -l -o |& >/dev/null 2>&1
 echo ' '
 echo 'For each job you want to move type y/return for no, q for quit'
 echo "Jobname                  execution time"
 while read -p _jobname _actual_time
 do
   echo $_jobname "          " $_actual_time
   read YESNO
   case ${YESNO}. in
     "y.")
     jobname=${_jobname}
     move_job ;;
     "q.") exit 0 ;;
   esac
 done
 ;;
*)
 if test -f ${atname_path}/${jobname}
 then
    echo atjob $1 does not exist - nothing moved
    exit 98
 fi
 move_job
esac
#
#                --------- End of script ch_at -------
```

LS_AT

```
#!/usr/bin/ksh
# ------------------------------------------------------+
# Script_name: ls_at                                    !
# Author     : Michael Imhotep                          !
# Ojectives  : Display contents of submitted at-jobs    !
#                                                       !
# Usage      : ls_at [jobname|-i]                       !
# ------------------------------------------------------+
#
atname_path=/var/spool/cron/atjobs
typeset -lu YESNO=n
#
if test $# -lt 1
then
```

```
   echo usage \:  ls_at jobnumber
   echo or
   echo usage \:  ls_at -i   # for interactiv selection
   exit 99
fi
case $1 in
 -i) echo ' '
     echo 'For each job you want to view type y or return for no, q
     ➤  for quit '
     at -l -o |&  > /dev/null 2> /dev/null
     echo "Jobname                    execution time"
     while read -p _name _time
     do
       echo $_name "       " $_time
       read YESNO
       case ${YESNO}. in
         "y.") pg ${atname_path}/${_name} ;;
         "q.") exit ;;
       esac
     done
     ;;
  *)
#
#
    if test -f  ${atname}
    then
      echo " "
      echo display only the specified job $atname
      echo " "
      pg ${atname_path}/${atname}
    else
      echo atjob with name $1 not found
      exit 1
    fi
  ;;
esac
#                 ------ End of script ls_at ------
```

Michael Imhotep (Australia)

You may have noticed in the code above that we're now using the symbol '➤' as a continuation character, when one line of code corresponds to several lines of print. We hope this makes code easier to read. This change does not affect code on our Web site.

On-line documentation with AIX 4.3

IBM has introduced many new functions in AIX 4.3. In common with earlier releases of AIX, some of them are not easily visible to the user or administrator. For instance, support for new systems and devices, and also performance improvements, are in most cases 'under the covers'. But there are also changes to AIX that are clearly visible and require some effort to get used to.

OVERVIEW

One of the first things you may struggle with is the fact that the on-line documentation is now available in HTML format. At first sight this is OK, as HTML is 'open', standardized, and driven by new technologies, such as the Internet. On the other hand, there is the question of how to access the documentation, given that the well-known **man** and **info** commands don't work unless some additional tasks are completed first.

By the way, if somebody still needs the **info** explorer – perhaps because some important home-grown documentation was created in 'info format' – it's still available as an option on the AIX licence.

INSTALLING THE ON-LINE DOCUMENTATION

During a normal manual installation of AIX, the administrator uses the first three CDs of the AIX package. These contain the base AIX system, but there are another four AIX CDs that are unaccounted for. Here is an overview of the contents of all CDs that come with AIX 4.3, including the Netscape CD:

- AIX V4.3 Volume 1 of 3
- AIX V4.3 Volume 2 of 3
- AIX V4.3 Volume 3 of 3
- AIX V4.3 Base Documentation
- AIX V4.3 Extended Documentation

- AIX V4.3 3D Graphics
- AIX V4.3 Bonus Pack
- Netscape FastTrack Server V2.1.

So, after installing AIX itself, we have to use the CD labelled ‘Base Documentation’ to install the on-line documentation. Using **smit** or the **installp** command, you install the filesets listed below:

- OpenGL.html.en_US.hypertext
- PEX_PHIGS.html.en_US.hypertext
- X11.html.en_US.hypertext
- bos.html.en_US.cmds.hypertext
- bos.html.en_US.files.hypertext
- bos.html.en_US.hypertext
- bos.html.en_US.manage_gds.hypertext
- bos.html.en_US.prog_gds.hypertext
- bos.html.en_US.techref.hypertext
- bos.html.en_US.topnav
- bos.html.en_US.user_gds.hypertext
- dsmit.html.en_US.hypertext
- sysmgt.help.en_US.

These filesets contain all the information referred to as ‘base documentation’. This refers to all manuals necessary for the day-to-day operation of an AIX system, including the command reference, file reference, system management guides, programming reference, etc. After installing the filesets you are immediately ready to access the information using the **man** command (see Figure 1).

Note the new format of the **man** pages – while in earlier releases of AIX **man** output was derived from the **info** database, in AIX 4.3 it’s derived from the HTML source and therefore looks a little different from that to which we are used.

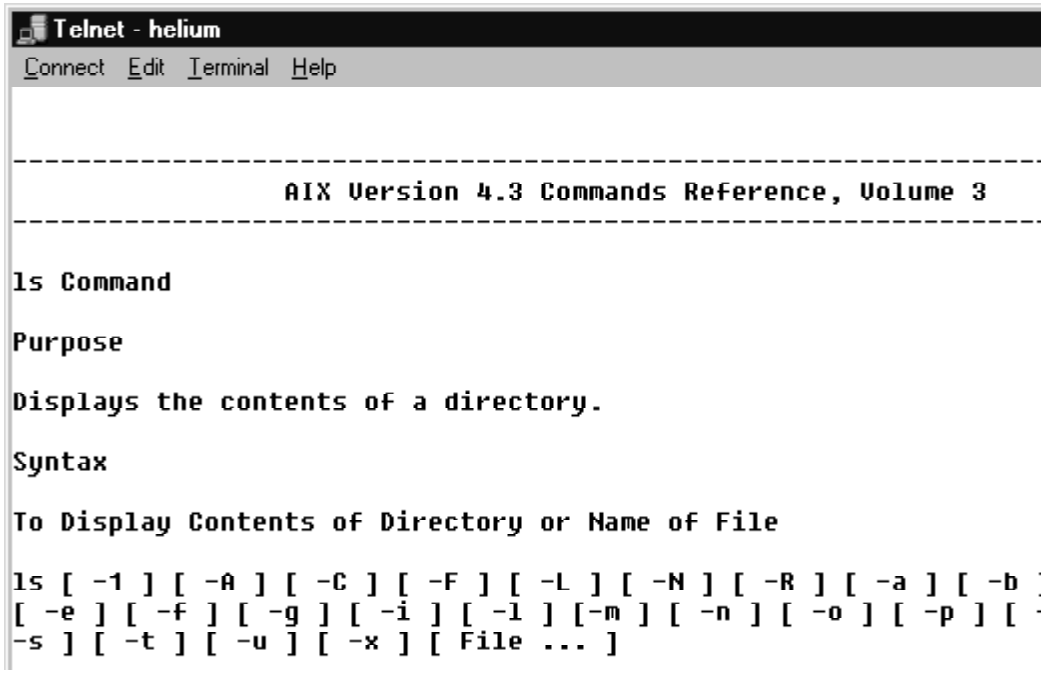

Figure 1: Accessing manual pages for AIX 4.3

Instead of **man** you may now use any browser installed on the local machine to view pages directly from the HTML source. As the Bonus Pack that's on one of the CD-ROMs in the AIX package contains Netscape Navigator, we recommend that you install the corresponding filesets using either **smit** or the **installp** command.

After starting Netscape (the binary is linked to */bin* and should be accessible through the *PATH*) you can specify the local file:

```
/usr/share/man/info/en_US/a_doc_lib/aixgen/topnav/topnav.htm
```

This opens the home page of the on-line documentation, giving you access to the other parts of the system.

ACCESSING BASE DOCUMENTATION FROM CD-ROM

If you want to save space on your disk (the documentation described above takes up approximately 200 MB of disk space) you may also access the documentation directly from the CD-ROM. You have to prepare a mount point called */infocd* and use **smit** or the command line

to create a CD-ROM filesystem: start **smit**, then follow the menu sequence below.

- *System Storage Management*
- *File Systems*
- *Add/Change/Show/Delete File System*
- *CDROM File Systems*
- *Add a CDROM File System*.

This takes you to the **smit** screen shown below.

```
                        Add a CDROM File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* DEVICE name                                         /dev/cd0          +
* MOUNT POINT                                         [/infocd]
  Mount AUTOMATICALLY at system restart?              no                +


Esc+1=Help          Esc+2=Refresh       Esc+3=Cancel        Esc+4=List
Esc+5=Reset         Esc+6=Command       Esc+7=Edit          Esc+8=Image
Esc+9=Shell         Esc+0=Exit          Enter=Do
```

After mounting the CD, you have to run a script located in the root directory of the CD:

```
/infocd/linkbasecd
```

This script creates the necessary links from the directories where the on-line documentation normally resides on disk to their images on CD-ROM. After running the script, you can access the information using either **man** or a browser, as discussed above.

Please note that before un-mounting the CD, it is necessary to remove the links created by the **linkbasecd** utility. Be sure to run **/infocd/unlinkbasecd** first. You may then unmount the CD and use the drive for other purposes. Of course, the on-line documentation isn't available unless you mount the CD or install the filesets.

ACCESSING EXTENDED DOCUMENTATION FROM CD

The CD labelled 'Extended Documentation' contains further manuals that are not necessary for regular system administration on a normal AIX system. On the other hand, some programmers and administrators may need the additional information on this CD, which includes manuals like:

- *AIX Programming Guides*
- *AIX Programming Reference*
- *X11 R6 Technical Specifications*
- *7318 Model S20 Guide And Reference*
- *AIX Versions 3.2 and 4 Asynchronous Communication Guide*
- *Fibre Channel Adapter/1063 User's Guide and Reference*
- *Enhanced SSA 4-Port Adapter Installation and Reference.*

To access the information, insert the CD into the drive, create a mount point called */infocd*, and use **smit** or the command line to create a CD-ROM filesystem, as described above. After mounting the filesystem, execute the script */infocd/linkextcd* to create links between the CD and the directories on the disk. You now are able to use **man** or a browser to access the information.

In common with the base documentation CD, ensure that you **use /infocd/unlinkextcd** to remove links between the CD and disk filesystems before you unmount the extended documentation CD.

ACCESSING THE DOCUMENTATION FROM A PC

If you have a PC, then you may also use its CD-ROM drive to access AIX's documentation. Just insert the CD and navigate to the directory:

```
E:\usr\share\man\info\en_US\a_doc_lib\aixgen\topnav
```

where *E:* is your CD-ROM drive. Double-clicking the file *topnav.htm* starts your default browser and gives you access to the documentation. Of course, you can also invoke your browser to open the file:

```
E:\usr\share\man\info\en_US\a_doc_lib\aixgen\topnav\topnav.htm
```

The path for accessing the extended documentation is:

```
E:\usr\share\man\info\en_US\a_doc_lib\aixgen\wxinfnav\topnav.htm
```

The ability to access AIX documentation from your PC is especially handy if – like the consultants and trainers in my company – you have to travel regularly. This allows you not only to while away those long and boring evenings in hotel rooms with new enlightenments from browsing through the data, but also makes you independent of a customer system that's possibly not so well installed and could otherwise be your only source of access to the information you need.

INSTALLING EXTENDED DOCUMENTATION

There are no installable filesets on the extended documentation CD, which means you can't use **installp** or **smit** to install the data on your local disk. Also there is no (easy) way of distributing the files using NIM (Network Installation Management). If you want to copy the data to your disk, you have to do it yourself using commands like **cp -r**. First, as described above, mount the CD as */infocd*, then make sure that you copy all the directories from the directory on the CD below to the corresponding directory on your disk.

```
/infocd/usr/share/man/info/en_US/a_doc_lib
```

SETTING UP A WEB SERVER

As HTML is the language of the Web, there are no major technical obstacles to overcome to make AIX 4.3's on-line documentation available over the network. If you're not familiar with Web server technology, here's how it's done.

Installing the filesets that contain the on-line documentation is step number one (out of three). Make sure that local access using **man** or a browser works well.

The second step is to install a Web server to provide an HTTP daemon that's able to serve HTML documents on the network. IBM provides two Web servers at no cost as part of the AIX Bonus Pack that's located on one of the CDs we mentioned above.

One easy solution is to install the Lotus Domino Go Web server that's based on the HTTP daemon formerly known as the IBM Internet Connection Server. Use **smit** or **installp** to install the filesets listed below. Because of US export regulations, users in Europe only get access to the less secure version of this server. There is also a special version for France (perhaps in reward for doing a good job in the World Cup, though you'd imagine winning it is enough!).

The third and last step is to link the HTML files to the data directory of the Web server to enable the **httpd** daemon to serve files. Using **smit**, WebSM, or the command line you have to configure your AIX system's Internet environment. I recommend you use the command **smit change_doc_search_server** to get immediate access to the dialogue overleaf.

```
                Change Local Documentation and Search Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
  Web server SOFTWARE                           Lotus Domino Go Webser>
* Local web server PORT number                  [80]                   #
* Local web server cgi-bin DIRECTORY            [/usr/lpp/internet/serv>
* Local web server HTML document directory      [/usr/lpp/internet/serv>


Esc+1=Help          Esc+2=Refresh       Esc+3=Cancel        Esc+4=List
Esc+5=Reset         Esc+6=Command       Esc+7=Edit          Esc+8=Image
Esc+9=Shell         Esc+0=Exit          Enter=Do
```

After setting those parameters you're now able to access the on-line documentation from any other system on the network using a browser and pointing to the URL:

```
http://host.domain/doc_link/en_US/a_doc_lib/
        aixgen/topnav/topnav.htm
```

Replace *host* with the hostname of the Web server's system and *domain* with your domain name. As Domino comes with its own search engine, you may click on the *Search* button and use the form returned to search through all or part of the documentation. The result is displayed as a list with links to articles that contain the actual information (Figure 2 overleaf).

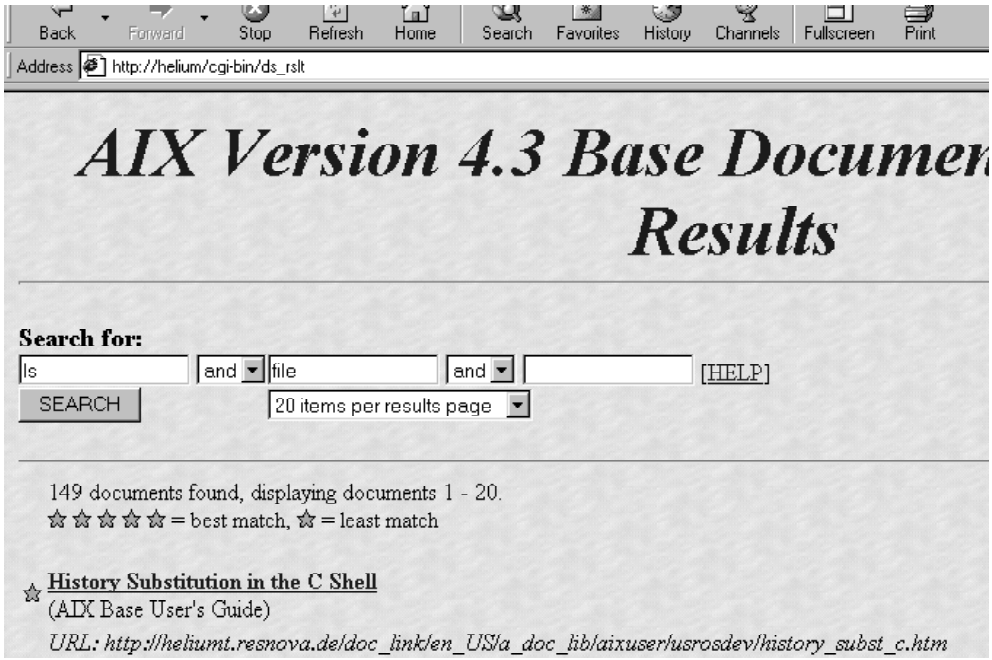

Figure 2: Result set of searching AIX documentation

ACCESSSING INFORMATION ON THE INTERNET

If necessary, you may access the AIX 4.3 on-line documentation via the Internet. This is especially useful if you have severe problems installing your AIX box and desperately need access to the manuals. Just start your preferred browser on any system and point it to the following URL.

```
http://www.rs6000.ibm.com/doc_link/en_US/a_doc_lib/
        aixgen/topnav/topnav.htm
```

You also may want to have a look at the following Web site:

```
http://www.rs6000.ibm.com/resource/
```

Here you'll find pointers to other documentation and information is available on the Internet.

FURTHER INFORMATION

More information on installing and using the AIX 4.3 on-line

documentation is available from:

- AIX Release Notes specific to the release of AIX on your installation CDs
- AIX 4.3 Installation Guide
- AIX 4.3 Systems Management Guide.

Michael Abel
IBM Certified AIX Technical Expert (CATE)
Senior Consultant
res nova Unternehmensberatung GmbH (Germany)

Implementing a reliable server process

INTRODUCTION

When implementing a daemon server process on AIX, it's very important to ensure the reliability of the process itself. The code provided with this paper is an example of a reliable server process: it can be used 'as is' or improved to meet your needs.

WHAT DOES THIS SERVER DO?

The code you find in this paper implements a sample TCP service on AIX systems named **aixtime**. When a client connects to this service, it receives back a timestamp string computed on the server. The server allows more than one client to be connected at the same time. When a client connects to the **aixtime** service, the daemon process 'forks', and the newly created process manages the conversation with the client while the original process continues listening for incoming requests. While in this example the conversation consists of no more than sending a timestamp computed when the *fork()* is executed back to the client, it could be more complex!

THE DAEMON PROCESS

When the daemon process starts, it performs some initialization tasks to ensure the daemon is reliable:

1 First of all, it performs a *fork()*, which makes it unnecessary to start the daemon process with the final '&' on the command line.

2 It sets the process's signal handling policies using the *signal()* function. In this example, all signals other than *SIGKILL* are ignored.

3 It closes all inherited file descriptors to avoid wasting resources.

4 It disconnects from the terminal to allow it to ignore all signals propagated by the terminal from which the server itself was started.

5 It tries to lock a specified file, the lock file, to avoid multiple copies of the daemon running at the same time.

6 If it is able to lock the lock file exclusively, it then writes its own PID in the file. This is useful for getting the PID of the server without executing a **ps** shell command.

7 It initializes the **syslog** facility to trace the activity of the daemon.

At the end of these tasks, the daemon starts listening for connections on a specified TCP port. If the daemon is started by *root*, the TCP port number can be specified either in the */etc/services* file or via the command line. Let's suppose you want the **aixtime** service to listen on TCP port 20000 and you're logged in as *root*. In this case, either you add the following line to */etc/services* file:

```
aixtime    20000/tcp    # aixtime sample service
```

or you start the daemon using the following command line:

```
aixtimed 20000
```

where **aixtimed** is the executable file produced by compiling the C code in this article. If the daemon is started by a non-*root* user, the only alternative is to specify the number port at the command line.

If you want the daemon process to stop, you have to send to it a

SIGKILL signal. The PID of the daemon process currently running is in the file *aixtimed.lock*, which is located in the same directory that contains the executable file **aixtimed**.

If the daemon has been started on your system, you may want to test the service. The test can be performed using **telnet**, for example, from an NT client. The command line is:

```
telnet <hostname> <port>
```

This should yield a response something like:

```
Fri Aug 28 13:12:45 1998.
```

Another way you can test the **aixtime** service is by compiling the code provided for the client. It's very simple, and, if you are using it in a production environment, you'll probably want to make some improvements to it.

SOFTWARE PRE-REQUISITES

The code has been tested on AIX 4.2 and compiled with the IBM C Set++ Compiler.

The server provided with this article uses AIX's **syslog** to log daemon activity. This means that it's important that the subsystem is running on the same AIX system as you are using.

The following command line reports the status of the **syslog** subsystem on your AIX system:

```
lssrc -l -s syslogd
```

If this subsystem is not running, you can start it with the following command line:

```
startsrc -s syslogd
```

The **syslogd** daemon writes all messages it receives to a file specified in */etc/syslog.conf*. If, for example, you want all the messages to be written to */home/reports/syslog.log*, you have to add the following line to */etc/syslog.conf* before starting the subsystem:

```
*.debug /home/reports/syslog.log
```

AIXTIMED.C

```
/*
**
** aixtimed.c
**
** 'aixtime' service server
**
*/

/*
 *---------------------------------------------------------------------
 * include section
 *---------------------------------------------------------------------
 */
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/errno.h>
#include <sys/signal.h>
#include <sys/socket.h>
#include <sys/resource.h>

#include <time.h>
#include <fcntl.h>
#include <stdio.h>
#include <ctype.h>
#include <netdb.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <syslog.h>
#include <signal.h>

/*
 *---------------------------------------------------------------------
 * define section
 *---------------------------------------------------------------------
 */
#define EXIT_OK    0        /* correct exit code                   */
#define EXIT_KO    1        /* incorrect exit code                 */

/*
 *---------------------------------------------------------------------
 * proto section
 *---------------------------------------------------------------------
 */
int Server( int sd );
```

```
/*
 *---------------------------------------------------------------------
 * main - Concurrent server for 'aixtime' service
 *---------------------------------------------------------------------
 */
int
main(int argc, char *argv[])
{
  int          lfd;                /* lock file descriptor         */
  int          msock;              /* master server socket         */
  int          ssock;              /* slave server socket          */
  size_t       socklen;            /* slave socket length          */
  char         pid[10];            /* PID number in ASCII format   */
  char         service[]="aixtime"; /* listen port number          */
  struct sockaddr_in csin;         /* client address               */
  struct sockaddr_in ssin;         /* server address               */
  struct servent  *pse;            /* service information entry    */
  struct protoent *ppe;            /* protocol information entry   */

  /*
  ** I'm setting the signals' handling policy
  */
  (void) signal(SIGHUP, SIG_IGN);      /* HUP   1 */
  (void) signal(SIGINT, SIG_IGN);      /* INT   2 */
  (void) signal(SIGQUIT,SIG_IGN);      /* QUIT  3 */
  (void) signal(SIGILL, SIG_IGN);      /* ILL   4 */
  (void) signal(SIGTRAP,SIG_IGN);      /* TRAP  5 */
  (void) signal(SIGABRT,SIG_IGN);      /* ABRT  6 */
  (void) signal(SIGEMT, SIG_IGN);      /* EMT   7 */
  (void) signal(SIGFPE, SIG_IGN);      /* FPE   8 */
                                       /* KILL  9 Cannot be trapped */
  (void) signal(SIGBUS, SIG_IGN);      /* BUS  10 */
  (void) signal(SIGSEGV,SIG_IGN);      /* SEGV 11 */
  (void) signal(SIGSYS, SIG_IGN);      /* SYS  12 */
  (void) signal(SIGPIPE,SIG_IGN);      /* PIPE 13 */
  (void) signal(SIGALRM,SIG_IGN);      /* ALRM 14 */
  (void) signal(SIGTERM,SIG_IGN);      /* TERM 15 */
  (void) signal(SIGURG, SIG_IGN);      /* URG  16 */
  (void) signal(SIGSTOP,SIG_IGN);      /* STOP 17 */
  (void) signal(SIGTSTP,SIG_IGN);      /* TSTP 18 */
  (void) signal(SIGCONT,SIG_IGN);      /* CONT 19 */
  (void) signal(SIGCHLD,SIG_IGN);      /* CHLD 20 Avoid zombies */
  (void) signal(SIGTTIN,SIG_IGN);      /* TTIN 21 */
  (void) signal(SIGTTOU,SIG_IGN);      /* TTOU 22 */
  (void) signal(SIGIO,  SIG_IGN);      /* IO   23 */
  (void) signal(SIGXCPU,SIG_IGN);      /* XCPU 24 */
  (void) signal(SIGXFSZ,SIG_IGN);      /* XFSZ 25 */
                                       /* 26 - 59 Not trapped */
  (void) signal(SIGGRANT,  SIG_IGN);   /* GRANT   60 */
  (void) signal(SIGRETRACT,SIG_IGN);   /* RETRACT 61 */
```

```
(void) signal(SIGSAK,    SIG_IGN);   /* SAK        */

/*
** Now I'm scanning and analyzing arguments
*/
switch (argc)
{
  case 1:
        break;
  case 2:
        strcpy(service,argv[1]);
        break;
  default:
        fprintf(stderr,"usage: %s [port]\n",argv[0]);
        exit (EXIT_KO);
}

/*
** I'm performing a fork() to execute in backgroud
*/
switch ( fork() )
{
  case  0: /* child process  */
         break;
  case -1: /* error in fork  */
         fprintf(stderr,"fork() error [%s]\n",
                         strerror(errno));
         exit (EXIT_KO);
         break;
  default: /* parent process */
         exit (EXIT_OK);
}

/*
** Now I'm initializing the syslog facility
*/
(void) openlog(argv[0], LOG_PID | LOG_CONS, LOG_USER);
syslog(LOG_INFO,"aixtime service starting on [%s - tcp]",
                 service);

/*
** I'm closing inherited standard files
*/
(void) fclose(stdin);
(void) fclose(stdout);
(void) fclose(stderr);

/*
** I'm disconnecting myself from terminal
*/
```

```
(void) setpgrp();

/*
** I'm setting the umask value
*/
(void) umask(027);

/*
** I'm checking a lockfile to avoid multiple running copies
*/
lfd = open("aixtimed.lock", O_RDWR | O_CREAT, 0640);
if ( lfd<0 )
{
  syslog(LOG_CRIT,"main :: open() [%s]\n",
                  strerror(errno));
  exit (EXIT_KO);
}
if ( lockf(lfd, F_TLOCK, 0) )
{
  syslog(LOG_INFO,"Unable to obtain exclusive lock!");
  exit (EXIT_OK);
}

/*
** Writing my own PID to lock file
*/
if ( sprintf(pid, "%6d\n", getpid())==EOF )
{
  syslog(LOG_CRIT,"main :: sprintf() [%s]\n",
                  strerror(errno));
  exit (EXIT_KO);
}
if ( write(lfd, pid, strlen(pid))<0 )
{
  syslog(LOG_CRIT,"main :: write() [%s]\n",
                  strerror(errno));
  exit (EXIT_KO);
}

/*
** Now I'm allocating a passive socket
*/
memset(&ssin, 0, sizeof(ssin));
ssin.sin_family = AF_INET;
ssin.sin_addr.s_addr = INADDR_ANY;

/*
** I'm mapping service name to port number
*/
if ( pse = getservbyname(service,"tcp") )
```

```
  ssin.sin_port=htons(ntohs((u_short)pse->s_port));
else if ( (ssin.sin_port=htons((u_short)atoi(service)))==0 )
     {
       syslog(LOG_CRIT,"main :: getservbyname(%s,tcp) [%s]\n",
                       service,
                       strerror(errno));
       exit (EXIT_KO);
     }

/*
** I'm mapping protocol name to protocol number
*/
if ( (ppe = getprotobyname("tcp"))==0 )
{
  syslog(LOG_CRIT,"main :: getprotobyname(tcp) [%s]\n",
                  strerror(errno));
  exit (EXIT_KO);
}

/*
** Allocating the passive socket
*/
msock = socket(PF_INET, SOCK_STREAM, ppe->p_proto);
if ( msock<0 )
{
  syslog(LOG_CRIT,"main :: socket() [%s]\n",
                  strerror(errno));
  return (EXIT_KO);
}

/*
** Binding the passive socket
*/
if ( bind(msock,(struct sockaddr *)&ssin,sizeof(ssin))<0 )
{
  syslog(LOG_CRIT,"main :: bind() [%s]\n",
                  strerror(errno));
  return (EXIT_KO);
}

/*
** listen()
*/
if ( listen(msock, 5)<0 )
{
  syslog(LOG_CRIT,"main :: listen() [%s]\n",
                  strerror(errno));
  return (EXIT_KO);
}
```

```
/*
** Requests' handling loop
*/
while (1)
{
  /*
  ** Getting the first incoming request
  */
  socklen = sizeof(csin);
  ssock = accept(msock, (struct sockaddr *)&csin, &socklen);
  if ( ssock<0 && errno!=EINTR )
  {
    syslog(LOG_CRIT,"main :: accept() [%s]\n",
                    strerror(errno));
    exit (EXIT_KO);
  }

  switch ( fork() )
  {
    case 0: /* child process */
          /*
          ** Writing client IP address to Log file
          */
          syslog(LOG_INFO,"Slave for aixtime service started by [%s]",
          ➤  inet_ntoa(csin.sin_addr));

          /*
          ** Inherited files closure
          */
          (void) close(lfd);

          /*
          ** Inherited sockets closure
          */
          (void) close(msock);

          /*
          ** Now I'm starting to handle client's requests
          */
          if ( Server(ssock) )
          {
            (void) close(ssock);
            syslog(LOG_INFO, "aixtime slave for [%s] stopped
            ➤  abnormally", inet_ntoa(csin.sin_addr));
            exit (EXIT_KO);
          }
          else
          {
            (void) close(ssock);
            syslog(LOG_INFO,"aixtime slave for [%s] normally stopped",
```

```
              ➤  inet_ntoa(csin.sin_addr));
              exit (EXIT_OK);
            }
      case -1: /* fork() error */
            syslog(LOG_CRIT,"main :: fork() [%s]\n",
                           strerror(errno));
            exit (EXIT_KO);
            break;
      default: /* parent process */
            (void) close(ssock);
    } /* switch */
  } /* while */
} /* main */

int
Server ( int socketd )
{
  char            msg[50];          /* message buffer               */
  struct linger   linger;           /* linger structure             */
  time_t          timestp;          /* timestamp                    */

  /*
  ** Set NO_LINGER option for soft closure
  */
  linger.l_onoff  = 1;
  linger.l_linger = 5;
  if ( setsockopt(socketd,SOL_SOCKET,SO_LINGER,&linger,sizeof
  ➤  (linger))<0 )
  {
    syslog(LOG_CRIT,"server :: setsockopt() [%s]\n",
                    strerror(errno));
    exit (EXIT_KO);
  }

  /*
  ** Sending back the timestamp
  */
  (void) time(&timestp);  strcpy (msg,ctime(&timestp));
  if ( send(socketd,msg,strlen(msg),1)<0 )
    return (EXIT_KO);
  else
    return (EXIT_OK);
}
```

AIXTIME.C

```
/*
**
** aixtime.c
```

```
**
** Client for 'aixtime' service
**
*/

/*
 *---------------------------------------------------------------------
 * include section
 *---------------------------------------------------------------------
 */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include <stdio.h>
#include <netdb.h>
#include <errno.h>

/*
 *---------------------------------------------------------------------
 * define section
 *---------------------------------------------------------------------
 */

#define BUFFLEN 128

/*
 *---------------------------------------------------------------------
 * main function
 *---------------------------------------------------------------------
 */
int
main(int argc, char *argv[])
{
  int s;                            /* connected socket descriptor  */
  struct hostent *hp;               /* info for remote host         */
  struct sockaddr_in myaddr_in;     /* for local socket address     */
  struct sockaddr_in peeraddr_in;   /* for peer socket address      */
  int addrlen, i;                   /* auxiliary vars               */
  char buf[BUFFLEN];                /* char buffer                  */

  /*
  ** Args analysis
  */
  if (argc != 3) {
      fprintf(stderr, "Usage:  %s <host> <port>\n",
          argv[0]);
      exit(1);
  }
```

```
/*
** Clear out address structures
*/
memset ((char *)&myaddr_in, 0, sizeof(struct sockaddr_in));
memset ((char *)&peeraddr_in, 0, sizeof(struct sockaddr_in));

/*
** Set up the peer address to which we will connect.
*/
peeraddr_in.sin_family = AF_INET;
hp = gethostbyname (argv[1]);
if (hp == NULL) {
    fprintf(stderr, "%s: gethostbyname(%s) [%s]\n",
            argv[0],
            argv[1],
            strerror(errno));
    exit(1);
}
peeraddr_in.sin_addr.s_addr =
         ((struct in_addr *)(hp->h_addr))->s_addr;
peeraddr_in.sin_port = atoi(argv[2]);

/*
** Create the socket.
*/
s = socket (AF_INET, SOCK_STREAM, 0);
if (s == -1) {
    fprintf(stderr, "%s: socket() [%s]\n",
        argv[0],
        strerror(errno));
    exit(1);
}
if (connect(s, (struct sockaddr *) &peeraddr_in,
        sizeof(struct sockaddr_in)) == -1) {
    fprintf(stderr, "%s: connect() [%s]\n",
        argv[0],
        strerror(errno));
    exit(1);
}

addrlen = sizeof(struct sockaddr_in);
if (getsockname(s, (struct sockaddr *)&myaddr_in,
        (size_t *)&addrlen) == -1) {
    fprintf(stderr, "%s: getsockbyname() [%s]\n",
        argv[0],
        strerror(errno));
    exit(1);
}
```

```
  /*
  ** Print out a startup message for the user.
  */
  printf("Connected to %s on port %u\n",
          argv[1],
          ntohs(myaddr_in.sin_port));

  /*
  ** Receive timestamp from aixtimed server
  */
  while ( (i = read(s, buf, BUFFLEN-1)) > 0 )
  {
        buf[i] = '\0';
        printf("%s", buf);
  };
  printf("\n\n");
  exit(0);
}
```

Marco Pirini
System Administrator (Italy)

DNS – where to start

When you first investigate setting up DNS for a production environment, it can be, to say the least, a trying experience. You start by looking at all the sources you can find, including the Internet, books, magazines, RFCs, and anything you can lay your hands on. And, let me tell you, there's a lot of information out there.

Now comes the task of weeding through this pile of 'stuff' that you've found. All this effort just to set up a basic DNS server – you start to pull your hair out!

If you're like me, the next step is to locate tools that will do the job for you. And, of course, you will find them, ranging from the very basic to the very powerful. However, I found that the tools available typically either need some other product to be installed (for instance, Perl) or are too cryptic to understand. (By this time you have pulled out large clumps of hair.)

This is why I chose to write a Korn shell script, **create.named**, that would:

1 Run on any flavour of Unix without having to pull down the latest version of Perl, etc.

2 Would be straightforward enough for my co-workers to use quickly without having to go through the same learning process that I had to go through.

Well, I started work on this task months ago, and to this day I've managed to adapt the script to fit only the operating systems I use most – AIX and Red Hat Linux. Maybe someday it will be 'open', like Unix, but only time will tell.

This script is designed to lead you through the basics of DNS. However, it assumes a few concepts are understood first.

ZONES

Primary zone

Consider the 'primary zone' to be the area of the Internet that you control directly. In basic terms, this is your 'domain'. If any host directly queries your DNS server about the hosts you have entered into it, the response is considered 'authoritative'.

Secondary zone

This includes servers that you may or may not directly control. Servers in this zone are able to examine and copy your DNS tables. They can translate names to IP addresses for your domain, but the response you get back from them is not considered to be an authoritative response.

At this point it's traditional to introduce the domain's naming tree, which for our company is:

```
ca   - top of the CA domain
 |
mb   - top of The MB.CA domain
 |
mpi  - top of the MPI.MB.CA domain (Manitoba Public Insurance)
```

In Canada the top node of our 'branch' of the Internet is the *ca* domain. There is an organization with which you have to register your 'domain name' (for instance '*mpi.mb.ca*') officially before you can publish on the Internet. And, although you can register and receive multiple domain names, I will only deal with one for the purpose of discussing the script.

The main reason for this is that we have a registered name outside our organization's firewall and a different one inside it (which, in our case, happens to be registered, but doesn't have to be). To handle this, we run different DNS servers: one outside the firewall and one for inside. The DNS server inside the firewall caches names from the outside, but the one outside the firewall can see the only names that belong outside our firewall.

So, on to the meat – the **create.named** script. Basically, this has two parts. The first one creates a configuration file (the default being *named.cfg*, which is located in your current working directory) with appropriate values that are utilized to create DNS files. The script runs through a series of information screens during the initial run, though this is also done if information is missing or the step is specifically requested by the user. At this stage the user is presented with screens prompting for specific information. Each screen indicates what the information is used for, gives examples (where appropriate), presents a default value, and asks the user to set a value that's appropriate for their environment. During this process, if the user wishes to select the default value, they just press <*enter*>. The configuration file is left after the script is run.

The second part of the script takes the configuration file and creates appropriate database files, along with *named.boot* (which tells the named daemon where all the database files are), and a cache file (the script explain the creation of the cache file). This step can take from five minutes to an hour or more. For our internal network, which consists of over 6000 hosts and 110 different IP networks, it took 30 minutes to run.

NOTES

1 Back up any database files you currently have in place.

2 During the configuration stage the script asks you where you would like to place the database files. I suggest putting them in */tmp/test* or a similar location first, before letting them loose in a production environment.

3 Once you move it into production, it takes a while for changes to filter through to the Internet, so it can seem frustrating at times.

USAGE

All options except **-h** can be used in conjunction with each other.

```
create.named
```

The first time this command is run, it goes through all the information screens and creates *named.cfg* in the current directory. In subsequent runs, it looks for *named.cfg* in the current directory and uses it to create the DNS files.

```
create.named -h
```

Prints the help screen.

```
create.named -f /dirname/named.cfg
```

Tells **create.named** where the configuration file that we're going to use is located.

```
create.named -r
```

Regenerates (runs through the configuration screens again) *named.cfg* before creating the DNS database files.

After you've run the script a few times, reviewed the output files, and put the files into a production location, you have to tell **named** that you've changed something. You can do this using the following commands:

```
stopsrc -s named
startsrc -s named
```

After you update your DNS configuration, I recommend that you try it out using **nslookup**. Below is the script. You'll notice I have left in some debugging options, so have fun enhancing the script! Also note the use of the continuation character, '➤' (see page 8).

CREATE.NAMED

```
#!/bin/ksh
#
#@(#)
#@(#) Program: create.named, author: Rob Crittenden, date: Aug 1998.
#@(#)
#@(#) Used to create files associated with named:
#@(#)   - named.boot
#@(#)   - database files (forward and reverse lookup)
#@(#)   - named.cfg (configuration file utilized by this program).
#@(#)
#@(#) Note: backup all named files before running this script.
#@(#)
#@(#) All options except -h can be used with each other.
#@(#)
#@(#) Examples:
#@(#)   create.named
#@(#)     - Runs through all information screens and creates
#@(#)       named.cfg in the current directory.
#@(#)     - Looks for named.cfg in the current directory.
#@(#)   create.named -h
#@(#)     - Prints out this help screen.
#@(#)   create.named -f /dirname/named.cfg
#@(#)     - Tells create.named where the configuration file
#@(#)       is that it's going to use.
#@(#)   create.named -r
#@(#)     - Regenerates named.cfg before creating the database
#@(#)       files.
#@(#)
#
#=============================================================================
#---------------------------NOTES ON PROGRAM----------------------
#=============================================================================
#
#  This script is designed to:
#     1. Create a configuration file for building the named database.
#        It leads you through a series of screens that (hopefully)
#        give an overview of the basics of DNS.
#     2. Create the named database files.
#
#  This script uses the following scheme for setting up variables:
#
#  Screen name = The first four characters of the variable and the
#                number of the screen. For example, the first screen
#                of the section about the Hostname is HOST01.
#
#  Variable names associated with screen HOST01 have a letter in
#  front of 'HOST01'. For example, the title of the HOST01 screen
#  is in environment variable THOST01.
```

```
#
#  The letters used are as follows:
#       T for the screen title
#       N for screen notes
#       E for the example
#       P for the screen prompt
#       D for the variable that holds the default value.
#
#  The following option doesn't work with all operating systems, as
#  DEBUG is not an accepted kill command. Therefore, it has limited
#  usage:
#       create.named -v basic | debug | step
#         - verbose mode on
#         - basic = section headings
#         - debug = line printouts
#         - step = line printouts and step through each line
#============================================================================
#---------------------------NOTES ON PROGRAM-------------------------
#============================================================================
#
#-------------PART ONE---------SETTING UP THE ENVIRONMENT------------
#
HOSTNAME=`hostname | cut -f1 -d'.'`
export PATH=/usr/ucb:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/
➤  local/sbin:/usr/local/bin/$HOSTNAME:/usr/local/sbin/$HOSTNAME:$HOME/
➤  bin:.

export PROG=`basename $0`
export PID=$$
export REGEN=OFF
export VERBOSE=OFF
export CONFIG=named.cfg
export OLDCONFIG=$CONFIG.old.$LOGNAME
export RESET=`tput sgr0`            #-----  RESET FORMATTING
export UNDER=`tput smul`            #-----  SET UNDERLINE FORMAT
export UPLINE=`tput cuu1`           #-----  GO UP ONE LINE
export TMP=/tmp/.$PROG.tmpfile.$$
export TMP1=/tmp/.$PROG.tmpfile1.$$
export TMP2=/tmp/.$PROG.tmpfile2.$$
export TMP3=/tmp/.$PROG.tmpfile3.$$
export SHHOSTS=/tmp/.$PROG.short.hosts.$$

trap 'rm -f $SHHOSTS $TMP $TMP1 $TMP2 $TMP3' 0

trap 'if [ ! -s $OLDCONFIG ]&&[ "$REGEN" = "ON" ]; then
  rm -f $CONFIG
else
  if [ -s $OLDCONFIG ]; then
    mv $OLDCONFIG $CONFIG
  fi
```

```
fi
echo ""
rm -f $SHHOSTS $TMP $TMP1 $TMP2 $TMP3
kill -9 $PID' 1 2 3 15

function what                        #-----  WHAT FUNCTION
{
VERBOSE --------------------  WHAT FUNCTION
#
#  Some versions of Unix do not support the what command, so I
#  added it as a function in the script itself to avoid hassle
#
echo $1:
cat $1 | grep "^#@(#)" | sed "s/#@(#)/  /g"
}

function DEBUG                       #-----  DEBUG SHOWS EVERY LINE
{
  echo "\n        ${UNDER}LINE $2: \c"    # Echo Line number
  cat $1 | head -$2 | tail -1             # Echo out the current line
  echo "$RESET\n"
}

function STEP                        #-----  STEP THROUGH EVERY LINE
{
  echo "\n        ${UNDER}LINE $2: \c"    # Echo Line number
  cat $1 | head -$2 | tail -1             # Echo out the current line
  echo "$RESET\n"
  echo "Press Enter To Continue\c"        # Prompt on screen
  read junk                               # read input
  echo "${UPLINE}                       " # Clear Prompt
}

function VERBOSE                           #-----  VERBOSE SHOWS TITLES
{
        if [ "${VERBOSE}" = "BASIC" ]; then
                echo "\n\n$*"
        fi
}

VERBOSE --------------------  PROCESS COMMAND LINE OPTIONS

while getopts :v:hrf: arguements ; do
  case $arguements in
  v)  VERBOSE=`echo ${OPTARG} | tr '[a-z]' '[A-Z]'`

      if [ "${VERBOSE}" != "BASIC" ]&&[ "${VERBOSE}" != "DEBUG" ]&&
      ➤  [ "${VERBOSE}" != "STEP" ]; then
        echo "\nInvalid Option. (${OPTARG})\n"
        what $0 | more
```

```
        exit
      fi

      if [ "${VERBOSE}" = "STEP" ]; then
        trap 'LN=${LINENO};STEP $0 $LN' DEBUG
      fi

      if [ "${VERBOSE}" = "DEBUG" ]; then
        trap 'LN=${LINENO};DEBUG $0 $LN' DEBUG
      fi
      ;;
  h)  what $0 | more                  #-----  SHOW HELP SCREEN
      exit
      ;;
  f)  export CONFIG=$OPTARG           #-----  CONFIGURATION FILE NAME
      ;;
  r)  export REGEN=ON                 #-----  REGENERATE CONFIG FILE
      ;;
 \?)  echo "\nInvalid Option. (${OPTARG})\n"
      what $0 | more
      exit
      ;;
  esac
done

VERBOSE --------------------  THIS SECTION RUN IF REGENERATE ON OR NEW

CHKCONFIG=`cat $CONFIG | wc -l`
integer CHKCONFIG
if [ -s $CONFIG ]&&(( $CHKCONFIG < 14 )); then
  echo "\nConfiguration file, ${CONFIG}, is missing some values."
  sleep 1
  echo "\nREGENERATION - AUTO STARTUP."
  sleep 2
  REGEN=ON
fi

if [ -s $CONFIG ]&&[ "$REGEN" = "ON" ]; then
  mv $CONFIG $OLDCONFIG
fi

if [ ! -s $CONFIG ]; then
  REGEN=ON
fi

function GETINFO
{
VERBOSE --------------------  FUNCTION GETINFO STARTED

VERBOSE --------------------  GET TITLE FOR SCREEN
```

```
  eval TITLE=\$T$1

  if [ "$TITLE" = "" ]; then
    export TITLE="ERROR - SCREEN CODE - $1 - NOT FOUND"
  fi

  integer LEN=${#TITLE}
  while (( LEN < 80 )); do
    TITLE=" ${TITLE} "
    integer LEN=${#TITLE}
  done

  export TITLE=`echo "${TITLE}" | cut -c2-81`


VERBOSE --------------------  GET NOTES FOR SCREEN

  eval NOTES=\$N$1
  export NOTES=`echo "$NOTES" | xargs -n10 echo "\t"`


VERBOSE --------------------  GET EXAMPLE FOR SCREEN

  eval EXAMPLE=\$E$1
  EXAMPLE=`echo "$EXAMPLE" | xargs -n8 echo "\t\t"`
  if [ "$EXAMPLE" != "" ]; then
    export EXAMPLE="\t Example:\n$EXAMPLE"
  fi


VERBOSE --------------------  GET PROMPT FOR SCREEN

  eval PROMPT=\$P$1

VERBOSE --------------------  GET THE DEFAULT VALUE

RESOLVCONF=/etc/resolv.conf
NAMEDBOOT=/etc/named.boot
PRIMARYDIR=`cat /etc/named.boot 2>/dev/null | grep "^directory" | head
➤  -1 | awk '{print $NF}'`
PRIMARYFILE=`cat /etc/named.boot 2>/dev/null | grep "^primary" | head
➤  -1 | awk '{print $NF}'`
PRIMARYDB="$PRIMARYDIR/$PRIMARYFILE"

DEFAULT=

if [ "$REGEN" = "ON" ]; then
  DEFAULT=`cat $OLDCONFIG 2>/dev/null | grep -v "^#" | cut -f1 -d# |
  ➤  grep -i "^$1" | awk '{print $NF}'`
else
```

```
  DEFAULT=`cat $CONFIG 2>/dev/null | grep -v "^#" | cut -f1 -d# | grep
  ➤  -i "^$1" | awk '{print $NF}'`
fi

if [ "$1" = "SERIO1" ]; then
  DEFAULT1=$DEFAULT
  DEFAULT=
  DEFAULT2="`date +%Y%j`00"
fi

if [ "$DEFAULT" = "" ]; then
  for x in $NAMEDBOOT $PRIMARYDB $RESOLVCONF; do
    DEFAULT=`cat $x 2>/dev/null|grep -v "^#"|grep -v "^\;" | grep -i
    ➤  "$2" | head -1 | cut -f1 -d# | cut -f1 -d';' | sed "s/)//g" |
    ➤  sed "s/(//g" | awk '{print $NF}'`
  if [ "$DEFAULT" != "" ]; then
    break
  fi
  done
fi

if [ "$DEFAULT" = "" ]; then
  DEFAULT="$3"
fi

if [ "$1" = "SERIO1" ]; then
  DEFAULT3=DEFAULT

  DEFAULT=`echo "$DEFAULT1\n$DEFAULT2\n$DEFAULT3" | sort -nr | head -1`

  integer LENSERIAL=${#DEFAULT}

  (( LASTSERNO = $LENSERIAL - 1 ))
  (( FIRSTSERNO = $LENSERIAL - 2 ))

  LASTTWOSER=`echo $DEFAULT | cut -c${LASTSERNO}-${LENSERIAL}`
  FIRSTSERIAL=`echo $DEFAULT | cut -c1-${FIRSTSERNO}`
  DEF2SERIAL=`echo $DEFAULT2 | cut -c1-${FIRSTSERNO}`

  if ["$DEF2SERIAL" = "$FIRSTSERIAL" ]&&[ "$LASTTWOSER" = "99"] ; then
  echo "\nThe maximum number of Serial Number changes in a day is 99."
  echo "\nYou have just reached it.  Sorry no Update."
  echo "\nExiting.\n"
  exit 1
fi

  DEFAULT=`echo "$DEFAULT" | awk '{print $1+1}'`

fi
```

```
eval D$1=$DEFAULT
}

function MAIN
{
VERBOSE --------------------  MAIN FUNCTION STARTED
#
#  THIS FUNCTION IS USED TO DO MOST OF THE ACTUAL WORK.
#    - CALLS THE GETINFO FUNCTION TO GET ALL THE ENVIRONMENTAL VALUES
#    - PRODUCES/UPDATES THE CONFIG FILE
#    - GENERATES THE DATABASE FILES
#

  GETINFO $1 "$2" "$3"

  if [ "$REGEN" = "ON" ]; then
    clear
    echo "(${1})"
    echo "$TITLE\n"
    echo "$NOTES\n"
    echo "$EXAMPLE\n\n"
    echo "$PROMPT ($DEFAULT): \c"
    read ANSWER junk
    if [ "$ANSWER" = "" ]; then
      echo $1 $DEFAULT >> $CONFIG
    else
      echo $1 $ANSWER >> $CONFIG
      eval D$1=$ANSWER
    fi
  fi

  if [ "$1" = "SERIO1" ]; then
    cat $CONFIG | grep -v SERIO1 > $TMP
    echo "$1 $DEFAULT" >> $TMP
    cp -p $CONFIG $CONFIG.bak
    cat $TMP > $CONFIG
  fi

}

#=============================================================================
#=============================================================================
VERBOSE --------------------  STARTING POINT OF THE PROGRAM
#
#  This section is where the title, notes and examples are written
#  out for each screen that will be shown to the user.
#
#------------------------------SCREEN TITLE-------------------------

 export THOSTO1="HOSTNAME OF DNS SERVER"
```

```
 export THOST02="NAME OF INPUT HOSTS FILE"
 export TDOMA01="DOMAIN NAME"
 export TSERI01="SERIAL NUMBER"
 export TREFR01="REFRESH PARAMETER"
 export TRETR01="RETRY PARAMETER"
 export TEXPI01="EXPIRE PARAMETER"
 export TMINI01="MINIMUM PARAMETER"
 export TCACH01="INTERNET CACHE FILE"
 export TMAIL01="PRIMARY MAIL SERVER"
 export TMAIL02="SECONDARY MAIL SERVER"
 export TNAME01="PRIMARY NAME SERVER"
 export TNAME02="SECONDARY NAME SERVER"
 export TDIRN01="NAMED FILES DIRECTORY"

#----------------------------SCREEN NOTES-------------------------

 export NHOST01="The hostname is the name of the machine on which the
                 DNS server runs. It is the short form of the name,
                 not the fully-qualified form."
 export NHOST02="The hosts file is a list of IP addresses, machine
                 names, and aliases."
 export NDOMA01="The domain name refects the hierarchy of network
                 zones that your DNS server serves. It contains hosts
                 for which you administer names and IP addresses."
 export NSERI01="The serial number is used by secondary servers that
                 query your DNS server. If the serial number is
                 incremented, the next time the secondary server
                 checks, it pulls down a new copy of your database.
                 This field is incremented each time you update your
                 DNS tables. The serial number is a positive, signed
                 32-bit number."
 export NREFR01="Polling by secondary servers is controlled by the
                 values REFRESH, RETRY, EXPIRE, and MINIMUM. All are
                 measured in seconds. REFRESH is used by secondary
                 servers after they are restarted or updated. This
                 value tells the server to wait X seconds before
                 checking with the your DNS server for an update."
 export NRETR01="RETRY tells the secondary server the number of
                 seconds to wait before trying again if it fails to
                 update from your DNS server."
 export NEXPI01="EXPIRE is used by secondary servers when they
                 reload DNS tables. If the server cannot perform a
                 serial number check during the specified interval,
                 it assumes that the copy of your DNS data is
                 obsolete and discards it."
 export NMINI01="MINIMUM relates to the TIME TO LIVE of DNS entries.
                 This means that, if the server grabs a copy of your
                 DNS database, the entries it grabs remain valid only
                 until the MINIMUM time is reached. MINIMUM overrides
                 TTL values that might be associated with each entry
```

```
                in the DNS tables."
 export NCACHO1="Your tables include only your domain, so what if
                you are searching for a site outside your domain?
                That is where the cache comes in. It lists top-level
                root servers that can reach all sites on the Net.
                When an address is found, it is cached on your server
                until the server restarts or the TTL is reached. You
                can download the latest list of root servers from:
                FTP.RS.INTERNIC.NET:/domain/named.root."
 export NMAILO1="The name of the first SMTP server or gateway.
                The location where mail is sent when addressed to:
                someone@your.domain.name."
 export NMAILO2="The name of the secondary SMTP server in case the
                first one cannot be reached in a timely manner."
 export NNAMEO1="The name of the primary DNS server for your domain or
                zone. This server is contacted first. If it is not
                available, secondary servers respond."
 export NNAMEO2="The name of the second DNS server or back-up for your
                domain. If the first server is unavailable for some
                reason, this one is contacted."
 export NDIRNO1="The directory where database files are created. The
                directory name should be fully qualified. The file
                named.boot is created in this directory and should be
                moved to /etc and modified to point to where database
                files eventually end up."

#-------------------------SCREEN EXAMPLES-------------------------

 export EHOSTO1="mpidev, not mpidev.mpic.mb.ca."
 export EHOSTO2="The usual location of the hosts file is /etc.
                However, you may chose to use another location."
 export EDOMAO1="mpic.mb.ca is the domain name and mpidev.mpic.mb.ca
                is a fully qualified hostname in this domain."
 export ESERIO1="The program defaults to yyyyjjj## (the current year,
                the julian day of the year, and a two-digit number).
                This allows 99 changes per day. If the current
                number is greater than 99, it uses that number."
 export EREFRO1="Keep this low during initial start-up to force
                secondaries to refresh frequently. 1800 seconds
                (30 minutes) is a good starting value. This puts
                an extra load on your DNS server, so, after the
                initial roll out, bump this value up."
 export ERETRO1="If you have difficulty starting your DNS server,
                have a quick retry from the secondary server. A
                start-up value of 900 seconds (15 minutes) is
                reasonable."
 export EEXPIO1="This value should be a little higher, so that, if
                your server is down when a secondary refreshes,
                your table will not expire for a few hours
                (7200 seconds)."
```

```
 export EMINIO1="Keep this value low during initial startup to force
                 secondaries to refresh frequently. A good value is
                 1800 seconds (30 minutes)."
 export ECACHO1="Specifying the fully qualified name of a current
                 cache file indicates that the program should copy
                 the file. If the cache does not exist or the default
                 value (-) is chosen, a cache file is created for
                 you."
 export EMAILO1=""
 export EMAILO2=""
 export ENAMEO1=""
 export ENAMEO2=""
 export EDIRNO1=""

#-------------------------------------  SCREEN PROMPT

 export PHOSTO1="Enter the hostname of the DNS server"
 export PHOSTO2="Enter the name of the hosts file"
 export PDOMAO1="Enter the domain name of this DNS server"
 export PSERIO1="Enter the next serial number to use"
 export PREFRO1="Enter the refresh rate [in seconds]"
 export PRETRO1="Enter the retry value [in seconds]"
 export PEXPIO1="Enter the expire value [in seconds]"
 export PMINIO1="Enter the minimum value [in seconds]"
 export PCACHO1="Enter the current cache file name"
 export PMAILO1="Enter the name of the primary mail server"
 export PMAILO2="Enter the name of the secondary mail server"
 export PNAMEO1="Enter the name of the primary DNS server"
 export PNAMEO2="Enter the name of the secondary DNS server"
 export PDIRNO1="Enter the directory name"

VERBOSE --------------------  CALL THE MAIN FUNCTION OF THE PROGRAM

MAIN HOSTO1 hostname $HOSTNAME
MAIN HOSTO2 hostfilename /etc/hosts
MAIN DOMAO1 domain `cat $DHOSTO2 2>/dev/null | grep $DHOSTO1 | head
➤  -1 | awk '{print $2}' | cut -f2-10 -d'.'`
MAIN SERIO1 serial 100000000
MAIN REFRO1 refresh 1800
MAIN RETRO1 retry 900
MAIN EXPIO1 expire 1800
MAIN MINIO1 minimum 1800
MAIN CACHO1 cache -
MAIN MAILO1 "MX  10" smtp.${DDOMAO1}
MAIN MAILO2 "MX  20" smtp2.${DDOMAO1}
MAIN NAMEO1 nameservername $DHOSTO1
MAIN NAMEO2 nameservername -
MAIN DIRNO1 directory /etc

if [ "$REGEN" = "ON" ]; then
```

```
  clear
  echo "The configuration information has been gathered.\n"
  echo "Processing...."
fi

VERBOSE --------------------  END OF CONFIGURATION PORTION

################################################################################
VERBOSE --------------------  PROCESSING THE DATABASE FILES
################################################################################
#
VERBOSE --------------------  CREATE DIRECTORY IF DOES NOT EXIST
#
if [ ! -d $DDIRNO1 ] ; then
  mkdir -m777 -p $DDIRNO1
fi
#
VERBOSE --------------------  CHECK THE INPUT FILE FOR DUPLICATES
#

if [ ! -s $DHOSTO2 ] ; then
  clear
  echo "The input hosts file does not exist - $DHOSTO2.\n"
  echo "Please correct and re-run $PROG with the -r option."
  exit 1
fi

while read LINE; do
  WORDLINE=${LINE%%#*}
  if [[ "$WORDLINE" != "" ]]; then
    for WORD in $WORDLINE; do
      echo "$WORD" >> $TMP1
    done
  fi
done < $DHOSTO2

cat $TMP1 | sort > $TMP2
mv $TMP2 $TMP1

cat $TMP1 | sort -u > $TMP2

comm -3 $TMP1 $TMP2 > $TMP3
if [ -s $TMP3 ]; then
  echo " "
  echo " "
  cat $TMP3
  mv $TMP3 `pwd`/duplicate-list
  echo " "
  echo "The IP/Name list above are duplicates found in $DHOSTO2."
  echo " "
```

```
  echo "Please correct them before re-running this script"
  echo " "
  echo "The duplicate IP/Name list stored in:"
  echo "     `pwd`/duplicate-list"
  echo " "
  echo " "
  echo "exiting..."
  rm -f $TMP1 $TMP2 $TMP3
  exit
fi

VERBOSE --------------------  CREATE SHORT LIST FROM HOST LIST

cat $DHOST02 | cut -f1 -d'#' | grep "^127\.0\.0\.1"  > $SHHOSTS

CHKLCLHST=`cat $SHHOSTS | grep "$DDOMA01"`
if [ "$CHKLCLHST" = "" ]; then
  CHKLCLHST=`cat $SHHOSTS | wc -c`
  if [ "$CHKLCLHST" = "" ]; then
    echo "127.0.0.1  localhost.${DDOMA01}" >> $SHHOSTS
  else
    cat $DHOST02 | cut -f1 -d'#' | grep "^127\.0\.0\.1" |\
    awk -vDOMAIN=$DDOMA01 '{print $1" "$2"."DOMAIN" "$3" "$4" "$5"
    ➤  "$6}' > $SHHOSTS
  fi
fi

cat $DHOST02 | cut -f1 -d'#' | grep $DDOMA01 | grep "^[0-9]" | grep -v
➤  "^127\.0\.0\.1" |  sort -n >> $SHHOSTS

VERBOSE --------------------  CREATE REQUIRED ENVIRONMENT VARIABLES
export SHDOMAIN=`echo $DDOMA01 | cut -f1 -d'.'`

export SHTMAIL1=`echo $DMAIL01 | sed "s/\.$DDOMA01//g"`
if [ "$SHTMAIL1" != "" ];then
  export LNGMAIL1=`echo $SHTMAIL1 | sed "s/$/\.$DDOMA01/g"`
  export MAILIP1=`cat $SHHOSTS | grep "$DDOMA01" | grep "$LNGMAIL1" |
  ➤  head -1 | awk '{print $1}'`
fi

export SHTMAIL2=`echo $DMAIL02 | sed "s/\.$DDOMA01//g"`
if [ "$SHTMAIL2" != "" ];then
  export LNGMAIL2=`echo $SHTMAIL2 | sed "s/$/\.$DDOMA01/g"`
  export MAILIP2=`cat $SHHOSTS | grep "$DDOMA01" | grep "$LNGMAIL2" |
  ➤  head -1 | awk '{print $1}'`
fi

export SHTHOST1=`echo $DHOST01  | sed "s/\.$DDOMA01//g"`
export LNGHOST1=`echo $SHTHOST1 | sed "s/$/\.$DDOMA01/g"`
```

```
export SHTDNAM1=`echo $DNAME01  | sed "s/\.$DDOMA01//g"`
export LNGDNAM1=`echo $SHTDNAM1 | sed "s/$/\.$DDOMA01/g"`

export SHTDNAM2=`echo $DNAME02  | sed "s/\.$DDOMA01//g"`
export LNGDNAM2=`echo $SHTDNAM2 | sed "s/$/\.$DDOMA01/g"`

#-----------------------------------------------------------------
# PROCESS_IP - used to read each line from the host list provided
# and break it down in to the primary name and secondary names.
function PROCESS_IP
{
  IP=$1

  PRIMNAME=""
  for x in $*; do
    PRIMNAME=`echo $x | grep "$DDOMA01" | sed "s/\.$DDOMA01//g"`
      if [ "$PRIMNAME" != "" ]; then
        break
      fi
  done

# SECLIST=`echo $* | sed "s/${IP}//g" | sed "s/${PRIMNAME}//g" | \
#   sed "s/\.$DDOMA01//g"`

  SECLIST=
  for x in `echo $* | sed "s/\.$DDOMA01//g"`;do
    if [ "$x" != "$IP" ]; then
      if [ "$x" != "$PRIMNAME" ]; then
        SECLIST="$SECLIST $x"
      fi
    fi
  done

  echo "$PRIMNAME IN A $IP"

  for SECNAME in $SECLIST; do
    echo "$SECNAME IN CNAME $PRIMNAME.$DDOMA01."
  done
}

#-----------------------------------------------------------------
# MHD is used to add comments to database file headers. It converts
# seconds into the appropriate unit of time.

function MHD
{
  integer SECONDS MINUTES HOURS DAYS
  SECONDS=$1
  (( MINUTES = $SECONDS / 60 ))
  (( HOURS = $MINUTES / 60 ))
```

```
  (( DAYS = $HOURS / 24 ))

  if (( $DAYS > 1 )); then
    export COMMENT="$DAYS days"
  else
    if (( $HOURS > 1 )); then
      export COMMENT="$HOURS hours"
    else
      if (( $MINUTES > 1 )); then
        export COMMENT="$MINUTES minutes"
      else
        export COMMENT="$SECONDS seconds"
      fi
    fi
  fi
}

MHD $DREFRO1
REFRCOMM=$COMMENT

MHD $DRETRO1
RTRYCOMM=$COMMENT

MHD $DEXPIO1
EXPCOMM=$COMMENT

MHD $DMINIO1
MINCOMM=$COMMENT

if [ "$LNGDNAM2" != "" ];then
  LDNAME2="\n                      IN  NS    $LNGDNAM2."
fi

if [ "$LNGMAIL1" != "" ];then
  LMAIL1="\n                     IN  MX 10 $LNGMAIL1."
fi

if [ "$LNGMAIL2" != "" ];then
  LMAIL2="\n                     IN  MX 20 $LNGMAIL2."
fi

HEADER="@ IN  SOA $LNGDNAM1. root.$LNGDNAM1. (\t; R.Crittenden 98
                          $DSERIO1\t\t; Serial Number
                          $DREFRO1\t\t; Refresh - $REFRCOMM
                          $DRETRO1\t\t; Retry - $RTRYCOMM
                          $DEXPIO1\t\t; Expire - $EXPCOMM
                          $DMINIO1  )\t\t; Min. TTL - $MINCOMM
                     IN  NS    $LNGDNAM1.${LDNAME2}${LMAIL1}${LMAIL2}\n"

VERBOSE --------------------  PRIMARY DATABASE CREATE
```

```
if [ -s $DDIRNO1/db.$SHDOMAIN ]; then         # BACKUP OF OLD FILES
  mv $DDIRNO1/db.$SHDOMAIN $DDIRNO1/db.$SHDOMAIN.old
fi

echo "$HEADER" > $DDIRNO1/db.$SHDOMAIN

if [ "$MAILIP1" != "" ];then
  echo "$DDOMAO1. IN A $MAILIP1" | awk '{printf "%-20s %-3s %-5s %-0.50s
  ➤  \n",$1,$2,$3,$4}' >> $DDIRNO1/db.$SHDOMAIN
fi

while read HST ; do
  PROCESS_IP $HST | awk '{printf "%-20s %-3s %-5s %-0.50s\n",$1,$2,$3,
  ➤  $4}'
done < $SHHOSTS > $TMP
cat $TMP | grep "^[a-zA-Z]" | sort >> $DDIRNO1/db.$SHDOMAIN

VERBOSE --------------------  START NAMED.BOOT CREATE

if [ -s $DDIRNO1/named.boot ]; then           # BACKUP OF OLD FILES
  mv $DDIRNO1/named.boot $DDIRNO1/named.boot.old
fi

echo ";
;   The named.boot should reside in /etc
;
;   If you change the directory where the database files reside,
;   change the line:
;     directory $DDIRNO1
;   to point to their location. However, named.boot should still
;   reside in /etc
;
;   R.Crittenden 98
;
;
;type               domain                         source file or host
;" > $DDIRNO1/named.boot

echo "directory $DDIRNO1\n" >> $DDIRNO1/named.boot

function FNAMEDB
{
  echo $* | awk '{printf "%-20s%-30s%-0.50s\n",$1,$2,$3}'>> $DDIRNO1/
  ➤  named.boot
}

FNAMEDB primary $DDOMAO1 db.$SHDOMAIN

VERBOSE --------------------  REVERSE DATABASE CREATE
```

```
for SUBNET in `cat $SHHOSTS | cut -f1-3 -d. | sort -u`; do

  if [ -s $DDIRNO1/db.$SUBNET ]; then         # BACKUP OF OLD FILES
    mv $DDIRNO1/db.$SUBNET $DDIRNO1/db.$SUBNET.old
  fi

  echo "$HEADER" > $DDIRNO1/db.$SUBNET

  RSUBNET=`echo $SUBNET | sed "s/\./ /g" | awk '{print $3"."$2"."$1}'`

#--------------------------UPDATE NAMED.BOOT----------------------
  FNAMEDB primary ${RSUBNET}.IN-ADDR.ARPA db.${SUBNET}

  cat $DDIRNO1/db.$SHDOMAIN | grep "$SUBNET" | grep -v "$DDOMAO1" | sed
  ➤  "s/\./ /g" | awk -v RSUBNET="$RSUBNET" -v DDOMAO1="$DDOMAO1"
  ➤  '{print $NF"."RSUBNET".IN-ADDR.ARPA. "$1"."DDOMAO1"."}' | sort -n
  ➤  | awk '{printf "%-31s IN  PTR   %-0.50s\n",$1,$2}' >> $DDIRNO1/
  ➤  db.$SUBNET
done

#-------------------------END OFF NAMED.BOOT-----------------------
FNAMEDB cache . db.cache

VERBOSE --------------------  DB.CACHE CREATE

if [ -s $DDIRNO1/db.cache ]; then           # BACKUP OF OLD FILES
  mv $DDIRNO1/db.cache $DDIRNO1/db.cache.old
fi

if [ "$DCACHO1" = "$DDIRNO1/db.cache" ]; then
  DCACHO1=$DDIRNO1/db.cache.old
fi

if [ "$DCACHO1" = "-" ]||[ ! -s "$DCACHO1" ];then
echo ";
;  This file holds information on root name servers needed to
;  initialize the cache of Internet domain name servers
;  (you need to reference this file in the:    cache  .  [file]
;  configuration file of BIND domain name servers).
;
;  This file is made available by InterNIC registration services
;  under anonymous FTP as
;      file                /domain/named.root
;      on server           FTP.RS.INTERNIC.NET
;  -OR- under Gopher at    RS.INTERNIC.NET
;      under menu          InterNIC Registration Services (NSI)
;      submenu             InterNIC Registration Archives
;      file                named.root
;
;  last update:    Aug 22, 1997
```

```
;  related version of root zone:   1997082200
;
;  R.Crittenden 98
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                        3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
```

```
;
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
; End of File" > $DDIRN01/db.cache

else
  cat $DCACH01 > $DDIRN01/db.cache
fi

VERBOSE --------------------  END OF PROGRAM
```

Rob Crittenden
Senior Unix Administrator
Manitoba Public Utilities (Canada)

Importing /etc/passwd and /etc/group

How do I import an */etc/passwd* or */etc/group* file from another system? If the other system is not AIX, then the first thing to do is copy the password and group entries for non-system users into AIX's */etc/passwd* and */etc/group* files.

Next, run **/bin/pwdck -t ALL**. This creates the required entries in the shadow password file (*/etc/security/users*). You should also run **usrck** and **grpck** to check the password and group files.

To duplicate the password and group entries from another AIX system, copy */etc/passwd*, */etc/group*, */etc/security/passwd*, */etc/security/group*, */etc/security/user*, */etc/security/limits*, and */etc/security/environ*. The last three are optional, unless you modified them. If you modified */etc/security/login.cfg*, you should also copy this file.

System Programmer (Switzerland)

What is the Object Database?

AIX stores most of the system management information in */etc/objrepos*, */usr/lib/objrepos*, and */usr/share/lib/objrepos*. Files (also referred to as 'system object classes') in these directories are administered by the Object Database Manager, ODM, which is a set of library routines and programs providing basic object-oriented database facilities.

Under most circumstances, only **smit** or the commands **smit** calls should be used to change the contents of system object classes. A harmless way to look at the object database is to use **odmget <Class>** where **<Class>** is one of the files in */etc/objrepos*.

Experienced users can use the ODM editor, **odme**, to navigate the database in detail. However, modifying the database should only be attempted if you know exactly what you are doing.

System Specialist (Switzerland)

Xephon's spelling checker has been caught red-handed – the **unmount** command in *AIX Filesystems* (Issue 32) should have been **umount**. Our apologies for this mistake.

AIX news

IBM announced that the RS/6000 Enterprise Server Model S70 is to be the first processor to support the new RS64-II microprocessor, providing it with significantly enhanced performance. To emphasize the new system's performance, IBM released data showing that the new S70 set a SPECweb96 record of 9,081 http operations per second in 12-way configuration, making it the world's fastest Web server by this measure. IBM also announced price reductions for previous-generation S70s and cuts of up to 37% on memory prices.

For further information contact your local IBM representative.

* * *

net.Genesis Corp has announced net.Analysis for AIX version 3.5, which provides analysis and reporting of Web sites hosted on AIX servers to allow organizations with business-critical Web applications to monitor the performance of these applications.

net.Analysis 3.5 for AIX is available now and priced at US$9,500 per licence. Each licence includes the net.Analysis Engine, one net.Analysis Reporter client, and ReportSite, which produces reports for different departments, divisions, or business units.

For further information contact:
net.Genesis, 215 First Street, Cambridge, MA 02142, USA
Tel: +1 617 577 9800
Fax: +1 617 577 9850
Web: http://www.netgen.com

Renaissance Virtual Software Limited
2 St Mary's Court, High Street, Newmarket, Suffolk CB8 8HQ, UK
Tel: +44 1638 569700
Fax: +44 1638 569701
Web: http://www.rvsl.com

* * *

Forte Software has announced Forte WebEnterprise Professional Edition, an application server for building, deploying, and managing transactional Web-based applications.

The product provides facilities for state and session management, class libraries for generating dynamic HTML pages, open editor support, a request parser, and a database query formatter. It also allows for the distributed execution of business rules with in-built deployment and management services.

Out in the first quarter of 1999, WebEnterprise Professional Edition will start at US$24,990 and be available on (in addition to AIX) Digital Unix, Solaris, and NT.

For further information contact:
Forte Software, 1800 Harrison St, 24th Fl, Oakland, CA 94612, USA
Tel: +1 510 869 3400,
Fax: +1 510 869 3480
Web: http://www.forte.com

Forte Software, Edenfield, London Road, Bracknell, Berks RG12 2XH, UK
Tel: +44 1344 482100
Fax: +44 1344 482500

xephon