70

AIX

August 2001

update

In this issue

AIX Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £180.00 in the UK; $275.00 in the USA and Canada; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1995 issue, are available separately to subscribers for £16.00 ($23.00) each including postage.

AIX Update on-line

Code from *AIX Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/aixupdate.html; you will need to supply a word from the printed issue.

Editors

Trevor Eddolls and Richard Watson

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.

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/contnote.html.

Printed in England.

Failed login report for multiple nodes

Here is a little script we use to go out across all SP/2 nodes each day and produce a report of failed logins. The failed login file is also initialized at the same time. The readable format of the failed login file is kept each day, then archived weekly. It is kicked off via cron on the Control Workstation. Archive text logs are also kept on the Control Workstation. The report is then e-mailed to an appropriate administrator for review.

SCRIPT

```
#!/usr/bin/ksh
# Ø1/23/2ØØ1 DBM new script - gather up and print/email
#  failed login entries
#
# collect up all failed login files from each node, format them, archive
# into one master file. master file is rotated on Mondays
#
. /.profile
export K5MUTE=1
DAY=`date +%a`
LOG=/tmp/--flx--
#
# gather up failedlogin entries from all nodes
#
/usr/lpp/ssp/bin/dsh /usr/sbin/acct/fwtmp "<" /etc/security/failedlogin
> /tmp/--fl--
#
# use awk to strip off only fields we need to save and format a report
#
echo " " > $LOG ;echo "Failed Login report for: " `date` >> $LOG
echo " " >> $LOG
cat /tmp/--fl-- | awk ' {printf("%-1Ø%-9s%-7s%-16s%-4s%-4s%-3s%-1Ø%s\n",
$1, $2, $3,\ $9, $1Ø, $11, $12, $13 ) }' >> $LOG
#
# send out null to all /failedlogin files to initialize
#
/usr/lpp/ssp/bin/dsh ">" /etc/security/failedlogin
#
echo " " >> $LOG ;echo "End of Failed Login report for: " `date` >> $LOG
echo " " >> $LOG
#
# add today's entries to the "master" log
```

```
#
cat $LOG >> /usr/local/bin/logs/failedlogins.txt
#
# If today is Monday trim the log
#
if [ `date +%a` = Mon ]; then
   echo " " >> $LOG
   echo  "Today is Monday, cleaning out old master log" >> $LOG
   rm -f /usr/local/bin/logs/failedlogins.txt.previous*
   cp /usr/local/bin/logs/failedlogins.txt usr/local/bin/logs/
failedlogins.txt.previous
   compress /usr/local/bin/logs/failedlogins.txt.previous
   > /usr/local/bin/logs/failedlogins.txt
fi
mail -s"Failed Login Report - SP/2 Nodes" @ntmail:dave.miller@bhs.org <
$LOG
exit
```

EXAMPLE REPORT

```
Failed Login report for:  Sun Jun 3 23:58:Ø3 EDT 2ØØ1

bhsspØ11: xxxxxxxx pts/Ø  172.16.35.61    Sun Jun 3  Ø3:4Ø:Ø6
bhsspØ11: UNKNOWN_ pts/8  172.16.69.25    Sun Jun 3  Ø8:Ø1:48
bhsspØ15: UNKNOWN_ pts/46 172.16.121.14   Sun Jun 3  Ø8:41:48
bhsspØ15: UNKNOWN_ pts/46 172.16.121.14   Sun Jun 3  Ø8:41:53
bhsspØ15: UNKNOWN_ pts/49 172.16.11.14Ø   Sun Jun 3  Ø9:22:27
bhsspØ15: xxxxxxos pts/4Ø 172.16.45.175   Sun Jun 3  Ø9:41:21
bhsspØ15: UNKNOWN_ pts/12 172.16.121.13   Sun Jun 3  11:Ø4:25
bhsspØ15: UNKNOWN_ pts/26 172.16.122.229  Sun Jun 3  11:17:17
bhsspØ15: UNKNOWN_ pts/26 172.16.122.229  Sun Jun 3  11:17:27
bhsspØ15: UNKNOWN_ pts/26 172.16.122.229  Sun Jun 3  11:17:31
```

David Miller
Database Architect
Baystate Health Systems (USA)

Online split mirror back-up

This script is used for doing a back-up of a filesystem while experiencing the least possible downtime. This downtime is the time it takes the computer to set variables, create a file allocation map, unmount the filesystem, split off one copy, and re-mount the filesystem. At this point customers can start their application(s). On most machines this is between 5 and 30 seconds. This allows customers the maximum use of their machines while preserving the integrity of the data.

The filesystem needs to be set up with two mirrored copies. To check this, use **lsvg –l <volume_group_name>**. The application is stopped and the filesystem unmounted. The second copy of the filesystem is removed and the orginal filesystem remounted, allowing operations to continue as per normal. A new filesystem is created (using an allocation map created previously) that contains the same data as the original filesystem.

The new filesystem is mounted as read-only to protect data integrity and then backed up. After the back-up is completed the filesystem is deleted to release the physical partitions to the free pool. Once the back-up is complete the second copy is removed, added back, and synchronized with the original filesystem, leaving the computer fully operational.

There are a lot of checks and tests throughout the program to ensure all operations are verified before proceeding.

The script can be called from anywhere although there are several files kept in the **/usr/local/Habackup** directory.

Debugging information is written to **/usr/local/Habackup/report**.

The file allocation map is **/usr/local/Habackup/alternate.<logical_volume_name>**.

There are only two variables that need to be set:

- **FILESYS="/data"** (/data being the filesystem to back up).

- **VOLUME="rootvg"** (rootvg being the volume group name /data is within).

This is an excellent program that several of our customers are running, giving a clean back-up with total data integrity. I have several different versions of the script that are easy to modify.

One customer has a three-way mirror with a fourth disk holding yesterday's data. Every day disks 3 and 4 swap, so that on the alternate day disk 3 has the copy of yesterday's data with disks 1, 2, and 4 being the mirror. They run the back-up as read-only then remount the filesystem as read-write to run reports without the live data changing (so they get accurate point-in-time reports). Also, at any point in time, they have no fewer than two copies of the data.

Remember that it is the recovery time in the event of a disaster that is critical not the time it takes to do the back-up.

CODE

```
#!/usr/bin/ksh
###############################################################################
##           Split Mirror Back-up created by Paul Tomlinson                  #
###############################################################################
###############################################################################
# This script will check to see if there are two mirrored copies of    #
# the data. The filesystem will be unmounted and the second copy will  #
# be split off, allowing the filesystem to be remounted and the        #
# application to continue. The second copy will then be remounted as   #
# read-only to provide data integrity and a complete back-up done      #
# online. After the back-up is completed the second copy is then       #
# unmounted, added as the second copy back to the orginial copy, and   #
# resynced dynamically on-the-fly.                                     #
###############################################################################
# Utility Functions
################################################################
setup_varibles ()
{
    FILESYS="/data"
    VOLUME="rootvg"
    REPORT="/usr/local/HAbackup/report"
    REM=0;PV1="";PV2="";PV3=""
    LVNAME=`lsfs $FILESYS | grep $FILESYS |cut -d "/" -f3 | awk  '{
print $1 }'`
    LVID=`getlvodm -l $LVNAME`
    VGNAME=`getlvodm -b $LVID`
```

```
    NUMBER=`getlvcb -c $LVNAME`
    SYNC=`lsvg -l $VOLUME | grep stale`
    PV1=`lslv -m $LVNAME | grep 0001 | head -n 1 | awk '{ print $3 }'`
    PV2=`lslv -m $LVNAME | grep 0001 | head -n 1 | awk '{ print $5 }'`
    PV3=`lslv -m $LVNAME | grep 0001 | head -n 1 | awk '{ print $7 }'`
    REM=`lslv -m $LVNAME | grep 0001 | head -n 1 | awk '{ print $5 }'`
    mkdir -p /usr/local/HAbackup
}

cleanup_files ()
{
    rm $REPORT > /dev/null 2>&1
    rm /usr/local/HAbackup/lastbackup > /dev/null 2>&1
    rm /usr/local/HAbackup/alternate.* > /dev/null 2>&1
}

snapshot_report ()
{
    date >> $REPORT
    echo Before Script ONE >> $REPORT
    lsvg -l $VOLUME >> $REPORT
    mount >> $REPORT
}

check_integrity ()
{
    if [ $NUMBER -eq 2 ]
      then echo
        echo "Filesystem mirrored 2 times" >> $REPORT
      else
        echo "Filesystem not mirrored to 2 physical partitions. Process
stop." >> $REPORT
        kill $PPID;exit 1
    fi

    # Make sure that the copies of the data is in sync
    if [ "$SYNC" = "" ]
      then
        echo "Filesystem synced." >> $REPORT
      else
        echo "Filesystem not synced. Process stop." >> $REPORT
        kill $PPID;exit 1
    fi
}

file_allocation_map ()
{
    lslv -m $LVNAME | grep hdisk | awk '{print $5":"$4}' > /usr/local/
HAbackup/alternate.$LVNAME
```

```
    if [ -f alternate.* ]
      then echo "File allocation map created" >> $REPORT
        else echo "Error No file created" >> $REPORT
    fi

    MAPSIZE=`wc -l /usr/local/HAbackup/alternate.$LVNAME | awk '{print
$1}'` >> $REPORT 2>&1
    echo $MAPSIZE >> $REPORT
}

umount_filesys ()
{
    sync;sync
    fuser -kxuc $FILESYS >> $REPORT 2>&1
    umount $FILESYS > /dev/null 2>&1
    MOUNT=`mount | grep $FILESYS | awk '{print $2}'`

    if [ "$MOUNT" = "$FILESYS" ]
      then
        echo "$FILESYS Filesystems still mounted" >> $REPORT
        fuser /dev/seclv1 >> $REPORT 2>&1
        ps -ef >> /usr/local/HAbackup/processes
        exit 1
      else
        echo "$FILESYS umounted. Done" >> $REPORT
    fi
}

splitoff_lv ()
{
    rmlvcopy $LVNAME 1 $REM
    mount $FILESYS > /dev/null 2>&1

    MOUNT=`mount | grep "$LVNAME" | awk '{print $2}'`
    if [ "$MOUNT" = "$FILESYS" ]
      then
        echo "$FILESYS split and remounted."  >> $REPORT
      else
        echo " Error Mounting $LVNAME" >> $REPORT
    fi
    snapshot_report
}

mount_bklvname ()
{
    LVNAME=`lsfs $FILESYS | grep $FILESYS | cut -d "/" -f3 | awk  '{
print $1 }'`
    LVID=`getlvodm -l $LVNAME`
    VGNAME=`getlvodm -b $LVID`
    MAPSIZE=`wc -l /usr/local/HAbackup/alternate.$LVNAME | awk '{print
```

```
$1}'`

    mklv -m /usr/local/HAbackup/alternate.$LVNAME -y bk$LVNAME $VGNAME
$MAPSIZE >> $REPORT

    if [ -d /mnt$LVNAME ]
      then
        echo /mnt$LVNAME already exists >> $REPORT
      else
        echo Creating /mnt$LVNAME as mount point for read only directory
>> $REPORT
        mkdir /mnt$LVNAME
    fi

    dd count=1 bs=4k skip=31 seek=1 if=/dev/bk$LVNAME of=/dev/bk$LVNAME
> /dev/null 2>&1

    mount -v jfs -o ro /dev/bk$LVNAME /mnt$LVNAME

    MOUNT=`mount | grep /mnt$LVNAME | awk '{print $1}'`
    if [ "$MOUNT" = "/dev/bk$LVNAME" ]
      then echo " Mounted /mnt$LVNAME OK" >> $REPORT
      else echo "Error mounting tmp filesystem" >> $REPORT
    fi
    snapshot_report

}

backup_filesys ()
{
    backup -Ø -uf /dev/rmtØ /mnt$LVNAME >> $REPORT
}

restore_mirrors ()
{
    umount /mnt$LVNAME

    UNMOUNT=`mount | grep /mnt$LVNAME`
    if [ "$UNMOUNT" = "" ]
      then echo "Unmounted /mnt$LVNAME." >> $REPORT
        else echo "  Error Unmounting /mnt$LVNAME "  >> $REPORT
    fi

    rmlv -f bk$LVNAME > /dev/null 2>&1

    REMLV=`lsvg -l $VOLUME | grep bk$LVNAME`
    if [ "$REMLV" = "" ]
      then
        echo "bk$LVNAME removed "  >> $REPORT
      else
```

```
        echo " Error removing bk$LVNAME" >> $REPORT
    fi

    LVNAME=`lsfs $FILESYS | grep $FILESYS |cut -d "/" -f3 | awk  '{
print $1 }'`
    NUM=Ø
    NUM=`getlvcb -c $LVNAME`
    NUM=`expr $NUM + 1`

    snapshot_report

    mklvcopy -m /usr/local/HAbackup/alternate.$LVNAME $LVNAME $NUM

    NUM=`getlvcb -c $LVNAME`
    if [ "$NUM" -eq "2" ]
      then
        echo "Added 2nd logical volume copy back to $LVNAME  Done"  >>
$REPORT
      else
        echo " Error creating logical volume" >> $REPORT
    fi

    NUM=`getlvcb -c $LVNAME`

    syncvg -l $LVNAME
    BIRD=`lsvg -l $VOLUME | grep stale`
    if [ "$BIRD" = "" ]
      then
        echo $LVNAME "has now be synced" >> $REPORT
      else
        echo "Error syncing" $LVNAME >> $REPORT
    fi

    snapshot_report
}

tape_verify ()
{
    restore -Tqvf /dev/rmtØ > /usr/local/HAbackup/lastbackup
    LINES=`wc -l /usr/local/HAbackup/lastbackup | awk '{ print $1 }'`
    echo "The backup has saved and verified $LINES files to tape" >>
$REPORT
    tctl -f /dev/rmtØ offline
}

### MAIN
#############################################################################
setup_varibles
cleanup_files
snapshot_report
```

```
check_integrity
file_allocation_map
umount_filesys
splitoff_lv
mount_bklvname
backup_filesys
restore_mirrors
tape_verify
```

Paul Tomlinson
Technical Consultant
Computer Systems Implementation Limited (New Zealand)

Improving AIX performance using HMT

It is amazing to see how fast the latest advances in the field of computer development are adopted for implementation. Hardware Multi-Threading (HMT), which was invented in the late nineties, has already been implemented in IBM's eServer pSeries (formerly RS/6000) servers. This article will provide a description of HMT concepts as well as giving directions for the implementation and utilization of this important new feature.

HMT CONCEPTS

Hardware Multi-Threading (HMT) is a concept in which a single physical computer processor (CPU) is simultaneously shared by two or more threads of software for execution. These threads provide additional instruction-level parallelism, enabling the processor to better utilize all of its resources. When one of the threads would normally be stalled, instructions from the other threads can utilize the processor's resources. For instance, when either a cache L1 or L2 miss occurs, which would normally delay the processor for many cycles, the processor switches to a different thread and attempts to execute its instructions. In this fashion the CPU is more fully utilized and the overall throughput improves.

HMT, as implemented in IBM's eServer running AIX OS, supports two execution contents (logical processors) per physical processor. At any time, only one of the contexts is active. The CPU performs switches between the contexts (threads) of execution at a fairly rapid rate, creating the illusion of the existence of two processors to the software that is using the server.

The operating system software controls the context switching of the processor's threads. Three basic classes of switch event exist:

- Hardware events such as cache misses and virtual memory translation misses.
- Time-out to prevent starvation of one thread by another.
- Software hints to allow for software control of various threads' priorities.

The switch event criteria are set by the operating system at system initialization. The hardware events are the primary triggers for switching between logical processors. According to IBM's research, the most frequent cause of a context-switch is an L1 cache miss. It is interesting to note that it is possible for both threads to cause switch-context events. In this case, HMT doesn't provide any additional parallelism.

HMT HARDWARE SUPPORT

HMT is implemented by RS64 IV Power PC microprocessor (code named SSstar) and is enabled on the following servers that support this processor:

- RS/6000 Enterprise Server M80 (with system firmware at level MM001108 or above)
- RS/6000 Enterprise Server S80
- IBM eServer pSeries p680
- IBM eServer pSeries p620 Model 6F1
- IBM eServer pSeries p660 Model 6H1.

Note that HMT is not supported on models p620 and p660 with RS64 III processors. HMT is not supported on systems that have only one functioning processor.

HMT SOFTWARE SUPPORT

Both AIX Versions 4.3.3 and AIX 5.1 support HMT.

The procedure below should be followed in order to enable HMT on servers that support it:

- Execute the command ***bosdebug –H on.*** The following response will be displayed, 'HMT on'.
- Execute the command ***bosboot.***
- Reboot the server, ***fastboot now***.

The procedure below should be followed in order to disable HMT on servers that support it:

- Execute the command ***bosdebug –H off***.
- Execute the command ***bosboot.***
- Reboot the server, ***fastboot now***.

The procedure below should be followed in order to check that HMT is enabled. Execute the command: ***bosdebug***. The following output will be displayed if HMT is enabled: 'HMT on'.

The procedure below should be followed in order to check that HMT is active. Execute the command ***bindprocessor –q.*** If HMT is enabled, the number of CPUs reported will be double the number of physical CPUs.

LIMITATIONS AND IMPLEMENTATION CHANGES CAUSED BY HMT

Unpredictable results (including system crashes) may occur when software has built-in dependencies on the number of processors in the system.

Dynamic Processor Deallocation (DPD) is not supported when HMT

is enabled. If DPD has been activated prior to HMT activation, it will not be configured at the next system boot and AIX will not deallocate problematic CPUs. If subsequently HMT is disabled, the DPD will be re-activated after the next re-boot.

When HMT is enabled, CPU utilization metrics at the thread, process, and processor level will be skewed depending on a workload running on the server. In particular, some of the sampling-based tools are less accurate when HMT is used when the CPU is lightly utilized.

The output of commands ***lsdev –C*** and ***lscfg*** will continue to report the actual number of physical CPUs installed in the system.

The ***diag*** command will continue to report and test the actual number of physical processors present.

The error EXDEV can be reported by the system subroutine ***bindintcpu***.

The following error message can be reported by system subroutine ***bindintcpu***:

```
Unable to assign interrupt level to specified processor.
```

The error EXDEV can be reported by system service ***i_int2cpu_ppc***.

When HMT is enabled, Capacity Upgrade On Demand (CuoD) constraints will be rejected.

The number of processors reported during initialization of the system after the display of the 'Welcome to AIX' message would be the actual number of physical processors present on the system.

The following messages will be displayed during a system boot with HMT and system debugger enabled:

- Starting NODE#000 physical CPU#000 as logical CPU#000 … done.
- Starting NODE#000 physical CPU#000 as logical CPU#001 … done.
- Starting NODE#000 physical CPU#001 as logical CPU#002 … done.

This signifies the start up of the system with the number of logical processors equal to twice the number of physical processors.

The command ***netstat –m*** will report the number of logical CPUs defined in the system.

The command ***bindprocessor -q*** will report the number of logical CPUs defined in the system.

The amount of CPU time reported as used by various threads may vary significantly for different runs.

The amount of CPU time reported as used by a thread performing a particular task will be increased in comparison to the amount of time reported when HMT is disabled.

HMT ATTRIBUTE PERFORMANCE CONSIDERATIONS

As can be seen from the references, it is unrealistic to expect gross improvements in the system's performance resulting from HMT implementation. In general, HMT increases raw throughput of CPU-intensive workloads, where CPU utilization is over 90%. Light workloads may see a decline in throughput when HMT is enabled. HMT should not be enabled in cases when near real-time response is expected for an application and an increase in the elapsed run-time for the application is undesirable.

REFERENCES

The following are useful references:

- /usr/lpp/bos/README.HMT.
- HMT White Paper, B R Olszewski, M Srivinas, G Mewhinney.
- ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr/misc_documents/HMT_wp.ps.
- *A multithreaded PowerPC processor for commercial servers*, J M Borkenhagen, R J Eikmeyer, R N Kalla, S R Kunker; *IBM Journal of Research and Development*, Vol. 44, No. 6, November 2000.

Alex Polak
System Engineer
APS (Israel)

Back-up files and directories

These scripts were produced when we were faced with 3GB of user files that had been put on a non-productive NT server. This server then required system maintenance on the disk layout. The original intention for the server was to be a standard HTTP help server which was installed out-of-the-box, but as users needed extra disk space we were left with important files on an NT server with no back-up device. So we developed the following scripts to back-up the directory and files on the server to an RS/6000 with sufficient disk space. The first part of the scripts was developed to copy the files from the NT server – we thought we would not need the 'restore' part because we were confident the maintenance work would be non-destructive.

The scripts as printed relate to a Unix FTP copy of files. The scripts were modified to copy files from a Unix box where we had only FTP user rights. The scripts to back-up files via FTP are listed below with explanations to tailor its use in the following section.

BACK-UP FILES VIA FTP SCRIPT

```
#!/bin/ksh
#############################################################
#
# FTP files from remote server
# generates files /tmp/makedirs
#                        /tmp/makefiles
#       These files need to be executed after this script
#############################################################
HOME=/sap40b
HOST=/export
NAME=hostname
USER=ftpuser
PW=xxxx

>/tmp/makedirs
>/tmp/makefiles

ftper()
{
ftp -ivn $NAME <<EOF >/tmp/getftp
user $USER $PW
```

```
cd $HOST
ls -lR
bye
EOF
}

ftper

grep "^[\.d]" /tmp/getftp |grep -v "\.\:" >/tmp/getdirs
echo "cd $HOME" >>/tmp/makedirs
cat /tmp/getdirs |while read LINE
do
CHAR=`echo $LINE|cut -c1 `
if [[ "$CHAR" = "." ]]
then
        echo $LINE|sed "s/:$//"|read ANS
        echo "mkdir -p \"$ANS\" ">>/tmp/makedirs
else
        if [[ "$ANS" != "" ]]
        then
                echo "cd \"$ANS\"" >> /tmp/makedirs
        fi
        #next lines make a cut on 60 for NT and 55 for Unix
        #make the necessary changes on the comments to get the NT line
 #NT line echo "mkdir -p \"`echo "$LINE" |cut -c60- `\"" >>/tmp/makedirs
        echo "mkdir -p \"`echo "$LINE" |cut -c55- `\"" >>/tmp/makedirs
fi

echo "cd $HOME">>/tmp/makedirs
done

ftper
grep "^[\.d-]" /tmp/getftp  >/tmp/getf
echo "cd $HOST" >>/tmp/makefiles
echo "lcd $HOME" >>/tmp/makefiles
cat /tmp/getf |while read LINE
do
CHAR=`echo $LINE|cut -c1 `
if [[ "$CHAR" = "." ]]
then
        echo $LINE|sed "s/:$//"|read ANS
                echo "cd \"$HOST/$ANS\"" >> /tmp/makefiles
                echo "lcd \"$HOME/$ANS\"" >> /tmp/makefiles
else
        if [[ "$CHAR" = "-" ]]
        then
        #NT line is cut on 60 and Unix on 55
        #echo "get \"`echo "$LINE" |cut -c60- `\"" >>/tmp/makefiles
        echo "get \"`echo "$LINE" |cut -c55- `\"" >>/tmp/makefiles
        fi
```

```
fi

done
#Now change permissions on file makedirs and makefiles
#and execute after checking files are accurate
```

CHANGING CODE FOR LOCAL USE

The following changes will need to be made to the code:

- HOME=/sap40b – host directory where the files will be backed up to.
- HOST=/export – remote target directory.
- NAME=hostname – remote target host name.
- USER=ftpuser – FTP user profile required to get access to remote files.
- PW=xxxx – password for FTP user.

NT AND UNIX CHANGES

To change the use for either NT or Unix, all lines noted in the script must be changed to cut out filenames at the correct place. (Always ensure files /tmp/makedirs and /tmp/makefiles are thoroughly checked before you execute them.) One line is printed below but make sure both entries in the script are consistent.

NT line:

```
echo "mkdir -p \"`echo "$LINE" |cut -c60- `\"" >>/tmp/makedirs
```

Unix line:

```
echo "mkdir -p \"`echo "$LINE" |cut -c55- `\"" >>/tmp/makedirs
```

EXECUTION OF MAKEDIRS AND MAKEFILES SCRIPTS

The files makedirs and makefiles are created after the above shell script is run; they are created in the /tmp directory. The makedirs file needs to be given execution permissions and checked thoroughly for consistency. Also check that the makefiles file is consistent because

this is used in the FTP execution. After these checks are done, run in the following order:

```
/tmp/makedirs
ftp -ivn < /tmp/makefiles
```

(ftp options – i = turns off interactive prompt, v = verbose, and n = turns off auto login checks.)

After both parts finish, all ordinary files on the remote server will have been copied via FTP to the current host.

GENERAL INFORMATION

The scripts do not copy hidden files (eg .profile or .sh_history), which will need to be copied manually if required.

The script has been used to FTP files on NT, AIX 4.3, and Solaris 7 without problems

No user, group ownership, or permissions are retained, but this is to be expected when using FTP.

RESTORE FILES VIA FTP SCRIPT

The restore part was needed because we originally lost the disk in the NT server. The restore recreates the directory structure first into a log file. The restore then recreates the file structure matching the host in another log file. This is all done via FTP on the target host.

```
#!/bin/ksh
#set -x
#########################################################
#       FTP utility                                     #
#     Puts directory structure to remote server         #
#       (normal files only)                             #
#       double check files after completion             #
#########################################################
integer BEEN
integer TEMP
integer NEXT
integer LINE
integer c
integer n
integer v
```

```
integer p
HOME=/h
HOST=/sap4Øb
tt1=/tmp/tt1
LOOP=y
c=2
T=1
echo "user anonymous ftp" >log
echo "bin" >>log
echo "prompt ">>log
echo "hash" >>log
echo "cd $HOME" >>log
echo "lcd $HOST" >>log
>/tmp/tmpfile
cd $HOST

find . -type d -print |grep -v "^\.$">$tt1

wrt()
{
#echo "print \"${TEXT}\" ">>log
echo "mkdir \"${TEXT1}\" ">>log
#echo "if [ \$? -gt Ø ]" >>log
#echo "then" >>log
#echo "print error" >>log
#echo "fi" >>log
echo "cd \"${TEXT1}\" " >>log
echo "lcd \"${HOST}/${TEXT}\" " >>log
#echo "mput *" >>log
}


while [ T -gt Ø ]
do
T=`cat $tt1 |cut -f$c -d/|grep -v ^$|wc -l`
c=$c+1
done
echo $c

n=2
for i in `cat $tt1 |cut -f$n -d/|uniq`
do
n=2
echo $i
TEXT=$i
TEXT1=$i
wrt
n=n+1
TEMP[$n]=`grep "^./$TEXT/" $tt1 |cut -f$n -d/|grep -v ^$|uniq |wc -l`
BEEN[$n]=Ø
```

```
while [[ "$LOOP" = "y" ]]
do
        if [ ${BEEN[$n]} -lt ${TEMP[$n]} ]
        then
                grep "^./$TEXT/" $tt1 |cut -f$n -d/|grep -v ^$|uniq >/
tmp/tmpfile$n
                grep "^./$TEXT/" $tt1 |cut -f$n -d/|grep -v ^$|uniq
                LINE=${TEMP[$n]}-${BEEN[$n]}
                echo $LINE
                TEXT=$TEXT/`tail -$LINE /tmp/tmpfile$n |head -1`
                TEXT1=`tail -$LINE /tmp/tmpfile$n |head -1`
                wrt
                BEEN[$n]=${BEEN[$n]}+1
                echo "a ${BEEN[*]}\t b  ${TEMP[*]}"
                echo a $TEXT
                n=n+1
                TEMP[$n]=`grep "^./$TEXT/" $tt1 |cut -f$n -d/|grep -v
^$|uniq |wc -l`
                BEEN[$n]=Ø
        else
                n=n-1
                v=n-2
                echo b $TEXT
                TEXT=`echo $TEXT| cut -f1-$v -d/`
                echo $TEXT| cut -f1-$v -d/
                echo "cd \"${HOME}/${TEXT}\" " >>log
                echo c $TEXT
                if [ n -eq 2 ]
                then
                        LOOP="n"
                fi
fi
done
echo "cd $HOME">>log
echo "lcd $HOST">>log
LOOP="y"
done
echo "user anonymous ftp" >logfile
echo "bin" >>logfile
echo "prompt ">>logfile
echo "hash" >>logfile
echo "cd /" >>logfile
echo "lcd /" >>logfile

cd $HOST
find . -type f -print |while read i
do
#i=`echo $i | sed "s#^\.##"`

echo "put \"${HOST}/$i\" \"${HOME}/$i\" ">>logfile
```

```
done

#ftp -ivn targethost <log >log.out 2>err.log &
#ftp -ivn targethost <logfile >logfile.out 2>errfile.log &
```

CHANGING CODE FOR LOCAL USE

The following changes to the code are necessary:

- HOME=/h – target directory for restore on the remote server.
- HOST=/sap40b – source directory on the host server.
- echo "user anonymous ftp" >log – change *both* the entries in the script with the appropriate user and password information.

EXECUTION OF THE RESTORE FTP SCRIPTS

The restore script produces two files that are used with FTP to perform the restore. These files are:

- log – for the directory restore.
- logfile – for the files restore.

To execute use the ‘log’ file with FTP first:

```
ftp -ivn targethost <log >log.out 2>err.log &
```

Afterwards, execute the ‘logfile’ with FTP:

```
ftp -ivn targethost <logfile >logfile.out 2>errfile.log &
```

GENERAL INFORMATION

File or directory names with spaces are catered for and will be restored.

Ensure the current directory you are working from has enough free space, because the log files can be quite long depending on the number of directories and files to transfer.

Robert Russell (UK) © Xephon 2001

Regular downloads via FTP

The following script displays a front-end menu, which allows us to quickly select various patches from (mainly SAP's) FTP servers. The main reason for the development was that SAP patches were updated and appended with a version number and previous versions existed in the same directory. Because our connection was over ISDN, we did not want to transfer files (at a cost) that we knew we did not need. So we wanted the ability to pick out what files we needed fairly quickly and transfer via FTP. The front-end is intended for use on servers with a small number of files that are changed frequently. Therefore it is limited to displaying 100 files or directories because any more and the practical use of the menu system would diminish.

FTPMENU SCRIPT

```
#!/bin/ksh
#########################################################################
# Dump out files that are for selection for FTP                         #
# this can then be read in to make the selection.                       #
# DOES NOT SUPPORT FILES OR DIRECTORIES WITH SPACES IN NAME #
# Controls via menu options                                             #
#                                                                       #
# when reading in                                                       #
# eg                                                                    #
# * file  = selected file                                               #
#   file  = no selection                                                #
# Always select with (z) before selecting (t) for transfer  #
#########################################################################
#
#Variables def
#set -x
set -A BONUS
set -A FILED
set -A STAR
integer count
integer TOT_REC
integer sele
integer max_sele
integer start_sele
integer fsele
integer max_fsele
```

```
integer start_fsele
integer f_count
integer temp
integer max
integer C
integer R
integer LINE
integer POS
integer TL
integer TOT_F
integer TOT_D

if [ $# -lt 1 ]
then
        echo "Usage ftpmenu (hostname) "
        echo "e.g. ftpmenu a "
        exit
fi

if [[ "$1" = "-h" ]]
then
echo "FTPMENU  "
echo "---------"
echo ""
echo "Command usage ftpmenu servername"
echo "Menu Options"
echo ""
echo "q) quit program"
echo "d) select directory "
echo "u) only valid for directories to go back up directory structure"
echo "f) select file"
echo "z) record file for ftp transfer"
echo "t) perform ftp transfer"
echo "b) select binary ftp transfer (only selected once)"
echo "a) select ascii ftp transfer (only selected once)"
echo "j) scroll down screen for more files/dirs"
echo "k) scroll up screen for previous files/dirs"
echo "r) right justify text"
echo "l) left justify text"
echo "v) view files for transfer"
exit
fi

ANS=$1
MAC=$1
#ADD new ftp servers into this section
case $ANS in
        aix)
                TARGET=master1
```



```
        USER=ftpuser
        PW=xxx
        ROOT_DIR=/h/files
        ENDOT="226 ASCII Transfer complete."
;;
s)
ENDOT="226 ASCII Transfer complete."
;;
n)
ENDOT="226 ASCII Transfer complete."
;;
l)
        TARGET=localhost
        USER=ftpuser
        PW=pass
        ROOT_DIR=/mnt/c/download
        ROOT_DIR=/home
        ENDOT="226 Transfer complete."
        MAC="l"
;;
targethost)
        TARGET=target
        USER=anonymous
        PW=ftp
        ROOT_DIR=/h
        ENDOT="226 ASCII Transfer complete."
;;
sapserv3)
        TARGET=sapserv3
        USER=ftp
        PW=ftp
        ROOT_DIR=/general/R3server/patches
        ENDOT="226 ASCII Transfer complete."
;;
sh9)
        TARGET=saphost
        USER=lnxadm
        PW=llllll
        ROOT_DIR=/export/home
        ENDOT="226 ASCII Transfer complete."
;;
Sunos)
        TARGET=sunserv1
        USER=ftpuser
        PW=pass
        ROOT_DIR=/tmp/dir1
        ENDOT="226 ASCII Transfer complete."
;;
*)
```

```
        echo "NOT VALID server"
        echo "tested on AIX4.3 SUNOS 2.7 NT4(no names with spaces)
Redhat 6.1"
        exit
esac

SET="d"
#TARGET=localhost
#USER=root
#PW=xxxxxx
#ROOT_DIR=/export

DATA=/tmp
BATCHFTP=$DATA/batchftp
FTPFILES=$DATA/ftpfiles
CONTROLFTP=$DATA/controlftp
LOGFTP=$DATA/logftp
EXIST=$DATA/existftp
FLIST=$DATA/flist
DATA_HOME=`pwd`
DFILES=$DATA/dfiles
OUTF=$DATA/outf
DIRS=$DATA/dirs
LOCK=$DATA/.lockmenu
BUILD="."
ACTUAL=""
RESET="A"
MODE="bin"
C=2
sele=Ø
PICKED="N"
fsele=Ø
if [ -f $LOCK ]
then
        echo "previous transfer not complete"
        echo "remove $LOCK file"
        exit
fi
touch $LOCK
if [ -f $CONTROLFTP ]
then
        mv $CONTROLFTP $CONTROLFTP.old
fi
>$FTPFILES
>$BATCHFTP
>$CONTROLFTP
>$EXIST
#SCREEN OUTPUT
stty rows 24
```

```
stty cols 8Ø
typeset -RZ2 OUT
typeset -L3Ø L_CHOICE
typeset -R3Ø R_CHOICE
BOLD=$(tput smso)
STND=$(tput smso)
NORM=$(tput rmso)
CH="L"
MORED="----"
MOREF="----"

ftper()
{
ftp -ivn<<EOF
open $TARGET
user $USER $PW
cd $ROOT_DIR
ls -laR
EOF

}

caller()
{
case $MAC in
        l)
                (ftper)|sed "s/\(.*\):$/\.\/\1:/"|tee $FLIST|grep "\:$"
|grep -v "\.\/\.\:"
        ;;
        *)
                (ftper)|tee $FLIST|grep "^\." |grep -v "\.\:"
        ;;
esac
}

dir_build()
{
caller|sed "s#:#/#g">$OUTF
trips
rips_f
}

trips()
{
set -A BONUS `cat $OUTF | grep "^$BUILD/"|sed "s/\://g" |cut -f$C -d/|
grep -v "^$" |sort |uniq|tee $DIRS|awk '{a=a$Ø" "}END{print a}'`
max_sele=`wc -l $DIRS|awk '{print $1}'`
#cat outf| awk '{a=a"\""$1"\" "}END{print a}'
}
```

```
rips_f()
{
>$DFILES

LINE=`wc -l $FLIST|awk '{print $1}'`
POS=`grep -n "^[\.]${ACTUAL}:" $FLIST|cut -f1 -d:`
TL=LINE-POS-1
FL=$(tail -${TL} $FLIST|head -1)
#while [[ "$FL" != "" && "$FL" != "226 ASCII Transfer complete." ]]
while [[ "$FL" != "" && "$FL" != $ENDOT ]]
do
        FL=$(tail -$TL $FLIST|head -1)
        T=`echo $FL|cut -c1`
        if [[ $T != "d" ]]
        then
                if [[ $T = "-" ]]
                then
                        echo $FL|awk '{print $9}' >>$DFILES
                fi
        fi
        TL=TL-1
done

set -A FILED `cat $DFILES | grep -v "^$" |awk '{a=a$Ø" "}END{print a}'`
max_fsele=$(wc -l $DFILES | awk '{print $1}')-1
}

d1()
{
if [[ "$SET" = "d" ]]
then
        tput cup $count 1;OUT=$sele;echo ${STND}${OUT}${NORM}
fi
CHOICE=${BONUS[$sele]}
if [[ "$CH" = "L" ]]
then
        L_CHOICE=$CHOICE
        tput cup $count 4;echo $L_CHOICE
else
        R_CHOICE=$CHOICE
        tput cup $count 4;echo $R_CHOICE
fi

}

f1()
{
if [[ "$SET" = "f" ]]
```

```
then
        tput cup $count 36;OUT=$fsele;echo ${STND}${OUT}${NORM}
fi
CHOICE=${FILED[$fsele]}
if [[ "$CH" = "L" ]]
then
        L_CHOICE=$CHOICE
        tput cup $count 39;echo $L_CHOICE
else
        R_CHOICE=$CHOICE
        tput cup $count 39;echo $R_CHOICE
fi
if [[ "${STAR[$fsele]}" = "*" ]]
then
        tput cup $count 38;echo "*"
fi
}

is_more()
{
#TOT_F=`wc -l $DFILES|awk '{print $1}'`
#TOT_D=`wc -l $DIRS|awk '{print $1}'`
TOT_F=start_fsele+16
TOT_D=start_sele+16

if [ TOT_F -lt max_fsele ]
then
        MOREF=${BOLD}"MORE"${NORM}
else
        MOREF="----"
fi
if [ TOT_D -lt max_sele ]
then
        MORED=${BOLD}"MORE"${NORM}
else
        MORED="----"
fi
}

outbatch()
{
echo "open $TARGET" >$BATCHFTP
if [[ "$SEC" != "y" ]]
then
        echo "user $USER $PW" >>$BATCHFTP
else
        echo "user $USER" >>$BATCHFTP
fi
echo "cd $ROOT_DIR">>$BATCHFTP
echo "lcd $DATA_HOME ">>$BATCHFTP
```

```
echo "$MODE">>$BATCHFTP
}

recorder()
{
                f_count=0
                while [ f_count -le max_fsele ]
                do
                        if [[ "${STAR[$f_count]}" = "*" ]]
                        then
                                #echo "${ACTUAL}${C}abc" >abc55
                                if [[ "$ACTUAL" = "" ]]
                                then
                                        echo "get ${ROOT_DIR}/
${FILED[f_count]} ${DATA_HOME}/${FILED[f_count]}">>$FTPFILES
                                        echo "${FILED[f_count]}">>$EXIST
                                        STAR[$f_count]=""
                                else
                                        echo "get ${ROOT_DIR}${ACTUAL}/
${FILED[f_count]} $DATA_HOME/${FILED[f_count]}">>$FTPFILES
                                        STAR[$f_count]=""
                                        echo "${FILED[f_count]}">>$EXIST
                                fi

                        fi
                                f_count=f_count+1
                done
PICKED="N"
TOT_REC=`wc -l $FTPFILES|awk '{print $1}'`

}

ref()
{
clear
tput cup 0 1
echo "${BOLD}FTP: ${TARGET}: Current Directory= ${BUILD}${NORM}"
tput cup 1 1
#echo"123456789*123456789*123456789*123456789*123456789*123456789*123456789*123456789"
echo "DATA_HOME=${DATA_HOME}\tMODE=${MODE} SELECTED FILES=${TOT_REC}"
echo ""
tput cup 2 1
echo "   Dir"; tput cup 2 35 ; echo "|";tput cup 2 39 ;echo "Files"
tput cup 3 1
echo "-------------------------------------------------------------------
----"
count=3
if [ max_sele -lt 16 ]
then
        sele=0
```

```
fi
if [ max_fsele -lt 16 ]
then
        fsele=Ø
fi
while (( (count=count+1) <= 19 ))
do
        if [[ "$SET" = "d" ]]
        then
                d1
                f1
        else
                d1
                f1
                #tput cup $count 37;OUT=$sele;echo ${BOLD}${OUT}${NORM}
        fi
        tput cup $count 35; echo "|"
        sele=sele+1
        fsele=fsele+1
done
tput cup 2Ø 1
echo "-------$MORED--------------------------------$MOREF----------------
-----"
tput cup 21 1
if [[ "$SET" = "d" ]]
        then
        echo "(q)uit (f)ile (u)pdir (z)rec (t)rans (b)in (a)scii (j)down
(k)up (r)ight (l)eft"
        else
        echo "(q)uit (d)dir (v)iew  (z)rec (t)rans (b)in (a)scii (j)down
(k)up (r)ight (l)eft"
        fi

}
runner()
{
outbatch
cat $BATCHFTP $FTPFILES >>$CONTROLFTP
clear
echo "Running FTP transfer"
echo "--------------------"
        ftp -ivn < $CONTROLFTP > $LOGFTP
more $LOGFTP
echo "----ftp-completed--------"
echo "exit"
while read FF
do
if [ ! -s $FF ]
then
        echo "Warning File $FF, zero length"
```

```
else
        echo "File $FF is `du -sk $FF|awk '{print $1}'` Kb"
fi
done<$EXIST
rm $LOCK
exit
}

#caller
dir_build
is_more
while true
do
ref
read NS?"> "
case $NS in
        a)
                MODE="ascii"
        ;;
        b)
                MODE="bin"
        ;;
        f)
                SET="f"
                sele=$start_sele
                fsele=$fstart_sele
        ;;
        d)
                SET="d"
                sele=$start_sele
                fsele=$fstart_sele
        ;;
        l)
                CH="L"
        ;;
        r)
                CH="R"
        ;;
        q)
                clear
                print "exit"
                outbatch
                cat $BATCHFTP $FTPFILES >>$CONTROLFTP
                exit
        ;;
        t)
                runner
        ;;
        v)
                clear
```

```
                cat $BATCHFTP $FTPFILES|more
                echo "---------------end--------------"
                read TEMP
        ;;
esac
if [[ "$SET" = "d" ]]
then
case $NS in
        [0-9]|[0-9][0-9])
                R=$NS
                if [[ "${BONUS[$R]}" != "" ]]
                then
                C=C+1
                if [ C -eq 2 ]
                then
                        BUILD="$BUILD${BONUS[$R]}""/"
                        ACTUAL="$ACTUAL${BONUS[$R]}""/"
                        trips
                        rips_f
                        sele=0
                        fsele=0
                        start_sele=0
                        start_fsele=0
                else
                        BUILD="$BUILD"/"${BONUS[$R]}"
                        ACTUAL="$ACTUAL"/"${BONUS[$R]}"
                        trips
                        rips_f
                        sele=0
                        fsele=0
                        start_sele=0
                        start_fsele=0
                fi
                if [[ "$PICKED" = "Y" ]]
                then
                        recorder
                fi
                fi
        ;;
        u|U)
                C=C-1
                if [ C -lt 2 ]
                then
                        C=2
                else
                        TEMP=`basename $BUILD`
                        BUILD=`echo $BUILD |sed "s/\/$TEMP//g"`
                        ACTUAL=`echo $ACTUAL |sed "s/\/$TEMP//g"`
                        trips
                        rips_f
```

```
                        sele=Ø
                        fsele=Ø
                        start_sele=Ø
                        start_fsele=Ø

                fi
                if [[ "$PICKED" = "Y" ]]
                then
                        recorder
                fi
        ;;
        j)
                temp=start_sele+16
                if [ temp -lt max_sele ]
                then
                        RESET="D"
                        start_sele=$start_sele+16
                fi
        ;;
        k)
                temp=start_sele-16
                if [ temp -ge Ø ]
                then
                        RESET="U"
                        start_sele=$start_sele-16
                fi
        ;;
        z)
                recorder
        ;;
esac
        sele=$start_sele
        fsele=$start_fsele
fi
if [[ "$SET" = "f" ]]
then
case $NS in
        [Ø-9]|[Ø-9][Ø-9])
                R=$NS
                #if [[ "${FILED[$R]}" = "*" ]]
                if [[ "${STAR[$R]}" = "*" ]]
                then
                        STAR[$R]=""
                else
                        STAR[$R]="*"
                        PICKED="Y"
                fi

        ;;
        j)
```

```
                temp=start_fsele+16
                if [ temp -lt max_fsele ]
                then
                        RESET="D"
                        start_fsele=$start_fsele+16
                fi
        ;;
        k)
                temp=start_fsele-16
                if [ temp -ge 0 ]
                then
                        RESET="U"
                        start_fsele=$start_fsele-16
                fi
        ;;
        z)
                recorder
        ;;

esac
        fsele=$start_fsele
        sele=$start_sele
fi

is_more

done
```

CHANGING CODE FOR LOCAL USE

There are two parts to the FTPMENU script that will need adapting for use. The section to add new FTP servers is highlighted in bold in the above script. The following is the template for AIX:

- aix) – shortcut when calling ftpmenu.
- TARGET=master1 – hostname of server.
- USER=ftpuser – FTP user ID.
- PW=xxx – password.
- ROOT_DIR=/h/files – target root directory on target FTP server, do not select '/' as the root directory, otherwise a long delay and problems will be created! Always narrow the selection as much as possible.

- ENDOT="226 ASCII Transfer complete." – needed to correctly determine the end of the FTP transfer. This is a standard entry and should not be changed, unless the termination message on your FTP server is different.

On Linux FTP servers (RedHat 6.1 tested) an extra variable is required; please note the following:

- ENDOT="226 Transfer complete." – needed to correctly determine the end of the FTP transfer.
- MAC="l" – needed to determine a Linux FTP server.

(In the script, ‘localhost’ is a Linux FTP server example.)

The following line can be changed to point the temporary work files to another directory:

```
DATA=/tmp
```

USING FTPMENU

The script should be set up in a directory in the PATH variable on the local server. This is because the script should be started from the directory where the FTP files will reside (as the current working directory is used to determine the host directory).

For example execute the following (in the Korn shell):

```
cd /usr/local/bin
vi ftpmenu
        (add the above ftpmenu script, make the necessary changes and
save the results)
PATH=$PATH:/usr/local/bin

cd /tmp
ftpmenu targethost
```

This will then execute the main function of the script.

An example of the FTPMENU options:

```
FTP: targethost: Current Directory= .
 DATA_HOME=/export      MODE=bin SELECTED FILES=
    Dir                            |   Files
 ------------------------------------------------------------------------
```

```
 00 dira                              |   file1
 01 dirb                              |   file10
 02                                   |   file11
 03                                   |   file12
 04                                   |   file13
 05                                   |   file14
 06                                   |   file15
 07                                   |   file16
 08                                   |   file17
 09                                   |   file18
 10                                   |   file19
 11                                   |   file2
 12                                   |   file20
 13                                   |   file21
 14                                   |   file22
 15                                   |   file23
 ----------------------------------------MORE---------------------
 (q)uit (f)ile (u)pdir (z)rec (t)rans (b)in (a)scii (j)down (k)up
(r)ight (l)eft
>
```

GENERAL INFORMATION

The script will not work if any directories or files have spaces in their names – entries in the menu will be corrupt and double entries will exist for single files/directories.

The limitation of 99 files or directories in the script is intended – any more and the practical use of the menu options would diminish because it would be quicker to manually select the files.

Files with the same name should be transferred in separate FTPMENU sessions because the last file transferred will overwrite any previous files in the current working directory.

The FTP file transfer can now be performed by following the menu options.

Menu options:

- d) select directory

  This is the default selection on start up of the script. This enables the selection of subdirectories.

  To select a directory the number range should be on the left of the screen. To select a directory simply select the number

corresponding to the directory required.

For example, in the example above, 0 or 00 would select directory dira. This would then change the screen display to show the directories and files in this directory.

- u) only valid for directories able to go back up the directory structure.

  This option is only valid when selecting directories (numbers on the left side of the menu). This option will go back up the directory structure until the ROOT_DIR (user-defined variable directory name).

- f) select file

  This option will allow the selection of files for FTP. It will enable the number selection to appear on the right of the screen. In the example above, to select file 23 enter the following:

  ```
  f

  15
  ```

  As a result an asterisk will appear next to the selected file. If the file is selected by mistake, type the number again and the asterisk will be removed.

- z) record file for FTP transfer

  This option records the selected files for later FTP transmission. Always select this option after selecting the files because no transfer will take place otherwise. It is essential to select z before leaving a directory or the selected file number will remain but no record of the transfer will be recorded or actioned. Therefore always select and record the files required in a single directory before changing directory, otherwise the transfer will not work as intended.

- t) perform FTP transfer

  This option will execute the FTP transfer. It will display the log file after the transfer has taken place. Also one minor check is performed for the size of the file in the host directory. This checks

that the file exists and is greater than 0 bytes. This option also removes the lock file to allow FTPMENU to be called again.

- b) select binary FTP transfer (only selected once)

  This option allows a binary transfer. It is a once-only selection: files cannot be mixed with ASCII transfer in one FTPMENU session.

- a) select ASCII FTP transfer (only selected once)

  This option allows an ASCII transfer. It is a once-only selection: files cannot be mixed with binary transfer in one FTPMENU session.

- j) scroll down screen for more files/directories

  This allows the selection of directories or files that are not currently displayed on the FTPMENU selection screen. If MORE appears on the bottom of either the directory or the files side of the menu then there are more files to choose from. (The range is 00 to 99.) There is no wrap-around if there are more than 100 files or directories in a single directory – any above 100 will be displayed but will not be selected.

- k) scroll up screen for previous files/directoriess

  00 represents the top of a directory or the first file in a directory. This option allows the selection to move back up the selection tree until 00 is reached.

- r) right-justify text

  This option allows the file or directory name to be right-justified. If the name of the file or directory is greater than 30 characters then this option allows the end of the name to be displayed.

- l) left-justify text

  This option allows the file or directory name to be left-justified.

- v) view files for transfer

  This option displays the current file names selected for FTP transfer. To exit this option press the *Enter* key.

- q) quit program

  Quit the program. This option will leave the lock file (/tmp/.lockmenu currently) and not allow FTPMENU to be called again. It is useful if you want to keep the batch files generated, to transfer files at a later time.

Robert Russell (UK)

Implementing AIX start-up/shutdown scripts in Unix SystemV-style

Installers of Maintenance Level 6 of AIX 4.3.3 were surprised to discover that the /etc/inittab file of their computers had been changed. The following is the relevant fragment from the /etc/inittab file on one of my own servers:

```
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
l7:7:wait:/etc/rc.d/rc 7
l8:8:wait:/etc/rc.d/rc 8
l9:9:wait:/etc/rc.d/rc 9
```

Additional related changes included the creation of the following nine directories:

```
/etc/rc.d
/etc/rc.d/rc2.d
/etc/rc.d/rc3.d
/etc/rc.d/rc4.d
/etc/rc.d/rc5.d
/etc/rc.d/rc6.d
/etc/rc.d/rc7.d
/etc/rc.d/rc8.d
/etc/rc.d/rc9.d
```

as well as installation of the /etc/rc.d/rc script.

The objective of this new feature is to allow system administrators to start and stop selected applications when the number running on the

operating system changes. The directories are provided for customers to place their own stop and start scripts.

The /etc/rc.d/rc script, supplied by IBM, is designed to use the run level input to look at the appropriate /etc/rc.d/rc<runlevel>.d and then execute scripts in this directory that start with ‘K’ to stop the applications. Then execute scripts starting with ‘S’ to start the applications.

The start and stop scripts are executed with a single parameter of ‘stop’ or ‘start’. No other values are valid. By doing it this way, the same script can be used for both starting and stopping an application by linking the start and stop scripts to a common file in /etc/rc.d. For example, a script named ‘samba’ could be in /etc/rc.d and the files /etc/rc.d/rc2.d/S80samba and /etc/rc.d//K80samba can be linked back to /etc/rc.d/samba.

The files will be executed in the order that they are sorted in.

To verify the order, execute **ls S* | sort** from the correct rc directory.

Sample /etc/rc.d/samba script:

```
#!/bin/sh
# This file should have uid root, gid sys and chmod 744
#
killproc() {            # kill the named process(es)
       pid=`/usr/bin/ps -e |  /usr/bin/grep -w $1 | /usr/bin/sed -e 's/^
*//' -e 's/ .*//'`
        [ "$pid" != "" ] && kill $pid
}

# Start/stop processes required for samba server

case "$1" in
'start')
#
# Edit these lines to suit your installation

  echo "Starting smbd..."
  /usr/local/samba/bin/smbd -D
  echo "Starting nmbd..."
  /usr/local/samba/bin/nmbd -D
;;
'stop')
   echo "Stopping smbd and nmbd..."
```

```
   killproc nmbd
   killproc smbd
 rm -f /usr/local/samba/var/locks/smbd.pid
 rm -f /usr/local/samba/var/locks/nmbd.pid

;;
*)
   echo "Usage: /etc/rc.d/samba  { start | stop }"
   ;;
esac
```

The support for automatic start-up and shutdown of application, using /etc/rc.d/rc script, is more consistent with the way the same function is implemented in modern Unix operating systems such as Solaris and HP-UX. It replaces the old, BSD-like, fashion of application start-up utilizing /etc/rc.tcpip, /etc/rc.nfs, and /etc/rc.local start-up scripts.

Alex Polak
System Engineer
APS (Israel)

Contributing to *AIX Update*

If you have ever experienced any difficulties with AIX, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon's wide range of products and services, simply by telling us all about it.

If you're interested in writing an article, but not sure what on, then visit the *AIX Update* Web site, http://www.xephon.com/aixupdate.html, and follow the link to *Opportunities for AIX specialists*. Here you'll find a list of topics that readers have asked us for more articles about.

Articles can be e-mailed to Trevor Eddolls at trevore@xephon.com. A copy of our *Notes for Contributors* is available from www.xephon.com/contnote.html.

Quick reference: Solaris to AIX

Though Sun Solaris specialists often first shake their heads in confusion when confronted with AIX, they will grudgingly accept its strengths after getting used to it. Later they will often praise its advantages. The two operating systems often use different paths to reach the same goal. They also usually differ in terms of terminology (and commands).

This reference contrasts the IBM AIX Version 4.3.3 and Sun Solaris 8 operating systems. The following Figures contrast the structure of these two Unix-based operating systems. Tasks are grouped according to major categories that are listed below:

- Software packaging
- Installing and upgrading tasks
- Booting and shutting down
- User management tasks
- Device management and configuration
- Network management and configuration
- Printer management and configuration
- File system management
- Virtual disk management
- Logical volume management
- Troubleshooting and additional location information.

SOFTWARE PACKAGING

Figure 1 contrasts AIX and Solaris software packaging details.

Units	*AIX Version 4.3.3*	*Solaris 8*
Smallest installable unit	fileset	package
Single installable image; distributed and installed as a unit	package	package
Logical grouping of packages	bundle	software cluster

Figure 1: Software packaging

INSTALLING AND UPGRADING TASKS

Figure 2 contrasts AIX and Solaris installing and upgrading tasks.

Tasks	*AIX Version 4.3.3*	*Solaris 8*
Install packages	installp -a or fast path: smitty install_latest	pkgadd
Display installed packages	lslpp -L or fast path: smitty list_installed_sw	pkginfo or pkgparam
Remove software package	installp -r or fast path: smitty reject installp -u or fast path: smitty remove	pkgrm
Verify correct installation	lppchk or fast path: smitty check_files	pkgchk
Install a patch	instfix or fast path: smitty update_by_fix	patchadd
Remove a patch	installp -r or fast path: smitty reject	patchrm

Figure 2a: Installing and upgrading tasks

Display installed patches	instfix -ia	showrev -p
Install OS on another disk (Alternate disk installation)	alt_disk_install	Live Upgrade
Create an installation server for network installation	nimconfig	setup_install_server install_dir_path
Create a boot server for network installation	smitty nim_config_env	setup_install_ server -b bootdirpath
Set up a client for network installation	nim -o bos_inst	add_install_client

Figure 2b: Installing and upgrading tasks

BOOTING AND SHUTTING DOWN

Figure 3 displays processes and locations of items that are involved in booting and shutting down a system in AIX and Solaris.

Tasks/Locations	*AIX Version 4.3.3*	*Solaris 8*
Boot process	Phases: • Read Only Storage (ROS): Check the system mother board, perform Power-On Self-Test (POST), locate the boot image, load the boot image into memory, begin system initialization and execute phase 1 of the /etc/rc.boot script • Base Device Configuration: Start	Phases: • Boot PROM: Display system information, run POST, load bootblk, locate ufsboot • Boot Programs: bootblk loads and executes the ufsboot • Kernel Initialization: ufsboot loads and executes the core kernel, initializes core kernel data structures, loads other kernel

Figure 3a: Booting and shutting down

	Configuration Manager to configure base devices • System Boot: Start init process phase 2, switch to hard-disk root file system, start other processes defined by records in the /etc/inittab file and execute phase 3 of the /etc/rc.boot script	modules based on the /etc/system file, starts /sbin/init program • init: Starts other processes based on the /etc/inittab file
Kernel modules directory	Kernel and kernel extension modules are stored in two directories: • /usr/lib/boot • /usr/lib/drivers	Kernel modules are stored in three directories: • /platform/sparc/kernel • /kernel • /usr/kernel
Create and stop processes and services for a current system run level based on the /etc/inittab file.	Set the default environment variables as defined in /etc/rc.	Set the default environment variables as defined in /etc/default/init.
System run levels	Defined run levels: • 0-1: Reserved for future use • 2: Multiuser state with NFS resources shared (default run level) • 3-9: Defined according to the user's preferences • m,M,s,S: Single-user state (maintenance level)	Eight run levels: • 0: Power-down state • s or S: Single-user state • 1: Administrative state • 2: Multiuser state • 3: Multiuser state with NFS resources shared (default run level) • 4: Alternative multiuser (not in use) • 5: Power-down state • 6: Reboot state

Figure 3b: Booting and shutting down

	• a,b,c: Starts processes assigned to the new run levels while leaving the existing processes at the current level running • Q,q: init command to reexamine the /etc/inittab file	
Determine a system's run level	who -r	who -r
Change a system's run level	telinit level number	Choose one of the following: halt init poweroff reboot shutdown telinit uadmin
Startup script	/etc/rc	/sbin/rc run-level number
Display boot information	bootinfo	N/A
Display or alter the list of boot devices	bootlist	boot

Figure 3c: Booting and shutting down

USER MANAGEMENT TASKS

Figure 4 displays tasks and location of files or information that are needed to perform user management in AIX and Solaris.

Tasks/Locations	*AIX Version 4.3.3*	*Solaris 8*
Run multiple tasks in a GUI environment	smit or wsm	admintool
Add a user	mkuser	useradd
Remove a user	rmuser	userdel
Change a user	chuser	usermod
List users	lsuser	listusers
Password files	/etc/passwd and/etc/ security/passwd	/etc/passwd and/etc/ shadow
Group files	/etc/group and/etc/ security/group	/etc/group
Process resource limits for users	/etc/security/limits	/etc/system
Systemwide environment file	/etc/environment	N/A
Environment attributes for users	/etc/security/environ	N/A
Configuration information for logging into system	/etc/security/login.cfg	/etc/default/login
Configuration information for user authentication	/etc/security/login.cfg	/etc/pam.conf
Profile template	/etc/security/.profile	/etc/skel/local.profile

Figure 4: User management tasks

Figure 5 is a list of tasks that are used for device management and configuration in AIX and Solaris.

Tasks	*AIX Version 4.3.3*	*Solaris 8*
Run multiple tasks in a GUI environment	smit or wsm	admintool
Configure a device	cfgmgr	Choose one of the following: drvconfig devlinks disks tapes ports
Define a device	mkdev	Choose one of the following: drvconfig devlinks disks tapes ports
Remove a device	rmdev	rem_drv
Change a device	chdev	N/A
List devices	lsdev	sysdef
Display device	lscfg	prtconf

Figure 5: Device management and configuration

NETWORK MANAGEMENT AND CONFIGURATION

Figure 6 is a list of tasks that are employed when performing network management and configuration in AIX and Solaris.

Tasks	*AIX Version 4.3.3*	*Solaris 8*
Run multiple tasks in a GUI environment	smit or wsm	N/A
Configure TCP/IP	mktcpip	ifconfig or /etc/ nsswitch.conf
Display interface settings	ifconfig	ifconfig
Configure interface	ifconfig	ifconfig
Change name service	chnamsv	/etc/nsswitch.conf
Unconfigure name service	rmnamsv	/etc/nsswitch.conf
Display name service	lsnamsv	/etc/nsswitch.conf

Figure 6: Network management and configuration

PRINTER MANAGEMENT AND CONFIGURATION

Figure 7 displays tasks that are involved in printer management and configuration in AIX and Solaris.

Tasks	*AIX Version 4.3.3*	*Solaris 8*
Run multiple tasks in a GUI environment	smit or wsm	admintool
Add a printer	mkdev	lpadmin
Start a printer	Start a queue: qadm	enable
Stop a printer	Stop a queue: qadm	disable
Add a printer class	N/A	lpadmin
Display print queue status	lpstat	lpstat
Cancel a print job	qcan	cancel
Add printer queue	Choose one of the following: mkque mkquedev mkvirprt	lpadmin
Change printer queue	Choose one of the following: chque chquedev chvirprt	lpadmin
Remove printer queue	Choose one of the following: rmque rmquedev rmvirprt	lpadmin
Display settings of printer queue	Choose one of the following: lsque lsquedev lsvirprt	lpadmin

Figure 7: Printer management and configuration

This article will be concluded in the next issue.

Werner Klauser
Klauser Informatik (Switzerland)

Candle has announced the CandleMonitor message processing plug-in node, part of CandleNet Command Center (CCC) for MQSeries Integrator Version 2.

It provides message flow performance and event monitoring for MQSI plug-in nodes. Each node is deployed in message flows using standard MQSI control-centre facilities, and performance statistics are gathered by placing the node at various points in a message flow, depending on the user's requirement for statistics.

The CandleMonitor node can help debug message flows by deploying it liberally during message flow development to detect failures and areas in the flow that perform badly.

Available for MQSeries Integrator Version 2 on AIX, Windows NT, and Sun Solaris.

For further information contact:
Candle, 201 N Douglas St, El Segundo, CA 90245, USA.
Tel: (310) 535 3600.
URL: http://www.candle.com.

* * *

Computer Associates has launched its CA-LPD output management tool, which enables secure management and delivery of reports from distributed platforms including AIX, OS/400, Unix, and Windows NT, and using high-speed mainframe printers and large storage devices.

It integrates with CA's report management products including CA-View, CA-Deliver, CA-Dispatch, and CA-Bundle, for automated report processing, storage, and delivery, as well as with other mainframe-based report management applications. Its functionality will also be incorporated in the next release of CA-Spool, the mainframe-based spooling and print management system that allows reports to be printed on any VTAM, PSF, or TCP/IP-attached printer.

For further information contact:
Computer Associates, 1 CA Plaza, Hauppauge, NY 11749, USA.
Tel: (516) 342 5224.
URL: http://www.ca.com.

* * *

Open Market has announced Version 3.6 of both Content Server and Content Centre, providing integration with additional operating system platforms, language support, and search capabilities.

Content Server 3.6 introduces support for AIX and Windows 2000, and there are new capabilities in the area of internationalization with certification for most European languages as well as Japanese, Chinese, and the UTF-8 character set.

Content Centre 3.6 is the CSEE browser-based application that enables non-technical users to create, manage, manipulate, and deliver content to numerous targets, including Web sites and wireless devices. It also provides automated workflow and version control.

For further information contact:
Open Market, 1 Wayside Rd, Burlington, MA 01803, USA.
Tel: (781) 359 3000
URL: http://www.openmarket.com.