113

AIX update

March 2005

In this issue

A xephon PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs $275.00 in the USA and Canada; £180.00 in the UK; £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 2000 issue, are available separately to subscribers for $24.00 (£16.00) 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/aix; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *AIX Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

AIX – has IBM got it wrong?

Last year IBM announced profits of $1.8 billion, but looking at the details showed that the pSeries Unix server revenue grew by only 1% year-to-year.

The big decision that IBM has made is to use the POWER5 chip, but it is the only vendor to use this chip. And its top of the range AIX system runs only on POWER chips.

Is it a good idea for IBM to limit itself to a single platform? Should it have chosen to run AIX on other platforms, like HP or Sun servers?

What's going to happen in the future? Will AIX sites continue to upgrade hardware and software as directed by IBM? Or will they want the flexibility that comes with Linux – the ability to run on a very large variety of hardware platforms? Being able to choose different hardware vendors could help a site keep down costs. So, by tying hardware to software, has IBM got it wrong?

But perhaps IBM doesn't have things completely wrong. Last year's IDC figures showed that IBM had 32% (virtually a third) of the server market and was showing revenue growth of 6%. Its nearest rival is Hewlett-Packard with 27% and a revenue growth rate of 3%. Interestingly, the biggest revenue growth rate was achieved by Dell with 14% – but it has only a 10% share of the server market. Similarly placed is Sun, whose 10% market share showed 0% revenue growth.

At the end of last year, Sun announced Solaris 10 – which will be given away free! Solaris 10 offers 'military-grade' security as standard, and provides 'container' technology – a virtualization tool, allowing one server to be divided so that hundreds of applications can be run. It also has performance analysis, diagnosis, and self-healing abilities, making the management of dynamic partitioning easier. This makes Solaris a definite alternative to AIX and HP-UX, although all three are proprietary versions of Unix.

The other clever part of Sun's plan is to allow Linux applications to run on Solaris through its Janus emulation software.

Has IBM got it wrong? Will Sun take away potential customers? Will locking AIX users to only one type of hardware seem like a good decision in five years' time? Should AIX support Linux like Solaris 10 does? Only time will tell.

Nick Nourse
Independent Consultant (UK)

AIX–Solaris differences

Because almost all system administrators work in multi-platform environments with AIX, Solaris, etc, I thought the following would be helpful.

I work intensively with AIX and Sun Solaris machines and wanted to share my experience. Many system administrators are mainly experienced in only one Unix operating system and know little about the other ones. This article will help AIX system administrators to know how to do the same things on Solaris, and, conversely, will help Solaris administrators to see how to administer AIX machines. It's like a two-way translator for administrators.

At the end you will see a very useful script, info.sh, that I wrote to collect information from Solaris machines.

THE TWO-WAY GUIDE

Here is the two-way guide (assuming Solaris with Volume Manager).

Backing up the operating system

- AIX

- Backing up to a tape:

```
# /usr/bin/mksysb -i /dev/rmt0
```

- Backing up to a file:

```
# /usr/bin/mksysb '-e' '-i' '-v' '-X' <filename>
```

where:

- -e = exclude file
- -i = generate new */image.data* file
- -v = verbose mode
- -X = expand */tmp* if needed.

- Solaris
  - Backing up to a tape:

```
# ufsdump 0ucf /dev/rmt/0cn /
```

  - Backing up to a file:

```
# ufsdump 0ucf <dump file> <filesystem name>
```

Restoring operating system

- AIX:

```
*Boot from mksysb image
```

- Solaris:

```
# ufsrestore ivf /dev/rmt/0cn  (interactive)
```

or:

```
# ufsrestore rvf /dev/rmt/0cn
```

Creating a volume group/disk group

- AIX:

```
# mkvg -s <PP-size> -y <new-vgname> <hdiskN ... >
```

- Solaris:

```
# vxdg init <dgname> <diskN>
```

Add a physical disk to a volume group/disk group

- AIX:

```
# extendvg <vg-name> hdiskN
```

- Solaris:

```
# vxdg -g <dg-name> adddisk <disk2>
```

Remove a physical disk from a volume group/disk

- AIX:

```
# reducevg <vg-name> hdiskN
```

- Solaris:

```
# vxdg -g <diskgroup> rmdisk <diskname>
```

Remove a logical volume from a volume group/disk group

- AIX:

```
# rmlv <lv-name>
```

- Solaris:

```
# vxedit -rf -g <dg-name> rm <volname>
```

Creating logical volumes

- AIX:

```
# mklv -y <lv-name> <vg-name> <size> <hdiskN ...>
```

(*size*= NxPP-size.)

  - for JFS2:

```
# mklv -y <lv-name> -t jfs2 <vg-name> <size> <hdiskN ...>
```

- Solaris:

```
# vxassist -g <dgname> make <volname> <size>
```

Removing logical volumes

- AIX:

```
# rmlv <lv-name>
```

- Solaris:
  - stop volume:

```
# vxvol -g <diskgrp> stop <volname>
```

  - remove volume:

```
# vxedit -rf -g <diskgrp> rm <volname>
```

Mirroring logical volumes

- AIX:

```
# mklvcopy <lv-name> Copies <hdiskN …>
```

- Solaris:

```
# vxassist -g <diskgroup> mirror <volname> <disk1><disk2> ...
```

Unmirror volumes

- AIX:

```
# rmlvcopy <lv-name> Copies hdiskX
```

- Solaris:

```
# vxplex -g <diskgrp> -o rm dis <disk-name>
```

Creating filesystems

- AIX:

```
# crfs -v jfs -g <vg-name> -m /<mount-point> -a <size> -a
frag=<fragment-size> nbpi=<number of bytes per i-node>
```

- Solaris:

```
# newfs /dev/vx/rdsk/<dgname>/<volname>
# mount /dev/vx/dsk/<dgname>/<volname> /<mount-point>
```

  - for VXFS:

```
# mkfs -F vxfs /dev/vx/rdsk/mydg/volname
# mount -F vxfs /dev/vx/dsk/<dgname>/<volname>  /<mount-point>
```

Add physical partitions to a logical volume

- AIX:

```
# extendlv <lv-name> <size>  (size= NxPP-size)
```

eg:

```
# extendlv test_lv 10
```

adds 10 PPs to the test_lv.

- Solaris:

```
# /etc/vx/bin/vxresize -g <dg-name> <volname> +<size>
```

eg:

```
# /etc/vx/bin/vxresize -g rootdg test_vol +2g
```

adds 2GB to test_vol in rootdg.

  - for vxfs:

```
/etc/vx/bin/vxresize -g <dg> -F vxfs <vol-name> <size>
```

Display volume group information

- AIX:

```
# lspv
# lsvg  <vg-name>
```

Listing information for each lv within the group:

```
# lsvg -l <vg-name>
```

- Solaris:

```
# vxdg list
# vxprint
```

Displaying all records in all diskgroups, with clearly displayed associations:

```
# vxprint -Ath
```

Move physical partitions from one disk to another

- AIX:

Move partitions from hdiskX to hdiskY:

```
# migratepv hdiskX hdiskY
```

- Solaris:

```
# vxevac -g <diskgroup> <disk-from> <disk-to>
```

Manage VM disks

- AIX:

```
# lspv
```

Displaying status and characteristics of physical volume hdiskN:

```
# lspv hdiskN
```

Displaying partition list of hdiskN:

```
# lspv -p hdiskN
```

Displaying logical volumes in hdiskN:

```
# lspv -l hdiskN
```

- Solaris:

```
# vxdisk list
```

Detailed information about the disk:

```
# vxdisk list <disk-name>
```

Removing disks from a volume group/disk group

- AIX:

```
# reducevg <volume-group> hdiskN
```

- Solaris:

```
# vxdg -g <diskgroup> rmdisk <diskname>
```

Displaying free space in disks

- AIX:

```
# lsvg -p <volume-group>
```

- Solaris:

```
# vxdg -g <disk-group> free
```

Set up sysboot information on VM disk

- AIX:

  Displaying boot sequence:

```
# bootlist -m normal -o
```

  Define boot sequence:

```
# bootlist -m normal hdiskX hdiskY ...
```

  Create a boot image for hdiskN:

```
# bosboot -ad /dev/hdiskN
```

- Solaris:

```
# vxbootsetup
```

  Define boot sequence in ok prompt:

```
ok setenv boot-device <root-disk> <root-mirror>
```

  Show/set EEPROM/NVRAM values:

```
# eeprom
```

  eg:

```
# eeprom auto-boot?=true
```

Reporting system error

- AIX:

```
# errpt
```

  To display a complete detailed report:

```
# errpt -a
```

- Solaris:

```
# cat /var/adm/messages
```

  and:

```
# prtdiag
```

List hardware configuration

- AIX:

```
# lscfg -v
```

- Solaris:

```
# /usr/platform/'uname -i'/sbin/prtdiag -v
```

and:

```
# prtconf -pv
```

Add a device without a reboot

- AIX:

```
# cfgmgr -v
```

- Solaris:

```
# devfsadm
```

  - Pre-Solaris 7 HW 11/99, use:

```
# drvconfig
# disks
# tapes
```

Show installed software

- AIX:

```
# lslpp -L
```

and:

```
# lslpp -l <Fileset-name>
```

- Solaris:

```
# pkginfo
```

  - to get information about a specific package:

```
# pkginfo -l <pkg-name>
```

Add software

- AIX:

```
# smitty installp
# smitty intall_all
```

or:

```
# /usr/lib/instl/sm_inst installp_cmd -a -Q -d <path> -f _all_latest -c
-N -g -X -G -Y
```

- Solaris:

```
# pkgadd -d <pkg-name>
```

Show installed patch level

- AIX:

```
# instfix -ivq
```

- Solaris:

```
# showrev -p
```

Install patches/fixes

- AIX:

```
# instfix -k <fix-name> -d <path>
```

- Solaris:

```
# patchadd <patch-name>
```

Encrypted passwords

- AIX:

```
/etc/security/passwd
```

- Solaris:

```
/etc/shadow
```

Check swap space

- AIX:

```
# lsps -a
```

- Solaris:

```
# swap -s
```

or:

```
#swap -l
```

NFS share definitions

- AIX:

```
/etc/exports
```

- Solaris:

```
/etc/dfs/dfstab
```

NFS share command

- AIX:

```
# exportfs -i  <directory>
```

  - unexport a directory:

```
# exportfs -u <dir>
```

  - export all directories in the */etc/exports* file:

```
# exportfs -a
```

  - show exported directories:

```
# exportfs -v
```

- Solaris:
  - share directory:

```
# share -F nfs -o ro <filesystem>
```

  - unshare:

```
# unshare <dir>
```

  - show shared directories:

```
# dfshares
```

Filesystem description

- AIX:

  ```
  /etc/filesystems
  ```

- Solaris:

  ```
  /etc/vfstab
  ```

Create non-0 length empty file

- AIX:

  ```
  # lmktemp <file> <size in bytes>
  ```

- Solaris:

  ```
  # mkfile <size> <file>
  ```

Determine which kernel is running (32- or 64-bit)

- AIX:

  ```
  # bootinfo -K
  ```

- Solaris:

  ```
  # isainfo -b
  ```

Reboot the server

- AIX:

  ```
  # shutdown -Fr
  ```

  where:

  – -r = reboot

  – -F = fast.

- Solaris:

  ```
  # shutdown -i6 -g0 -y
  ```

  where:

  – -i = init-state

- -g = grace-period
- -y = pre-answer the confirmation question.

or:

```
# init 6
```

- ok prompt boot:
  - start the system from the default boot device:

    ```
    ok boot
    ```

  - start the system in single user mode:

    ```
    ok boot -s
    ```

- Interactive boot:

  ```
  ok boot -a
  ```

  - start the sys while checking new devices and creating device entries in the */device* and */dev*:

    ```
    ok boot -r
    ```

  - start the system while displaying the details of installed devices:

    ```
    ok boot -v
    ```

Shut down the server

- AIX:

  ```
  # shutdown
  ```

- Solaris:

  ```
  # shutdown -h now
  ```

  or:

  ```
  # init 5
  ```

Run levels

- AIX:

  - 2: multiuser
  - 6: reboot
- Solaris:
  - 0: firmware monitor (ok prompt)
  - s: single user
  - 1: sys admin
  - 2: multiuser
  - 3: share NFS
  - 4: user defined
  - 5: power-down
  - 6: reboot.

Add a user

- AIX:

```
# mkuser id=<uid> groups=<groups> home=<home dir> shell=<shell>
gecos="<comment> umask=<umask> <user-name>
```

- Solaris:

```
# useradd -u <uid> -g <group-id> -c "<comment>" -d <home dir> -m -s
<shell> <user-name>
```

Delete a user

- AIX:

  ```
  # rmuser -p <user-name>
  ```

  where -p removes user account and all its attributes, including passwords and other user authentication information.

- Solaris:

  ```
  # userdel -r <user-name>
  ```

  where -r also removes the user's home directory.

List all users

- AIX:

```
# lsuser -a  ALL
```

- Solaris:

```
# logins
```

Mount CD-ROM

- AIX:

```
# mount /cdrom
```

or:

```
mount -v cdrfs -o ro /dev/cd0 /cdrom
```

- Solaris:

```
# mount -F hsfs /dev/sr0  /cdrom
```

or:

```
# mount -F hsfs -o ro /dev/dsk/c0t6d0s2 /cdrom
```

Show connected enclosures (storage devices)

- AIX:

```
# lsdev -C | grep enclosure
```

  - display information about the enclosure:

```
# lscfg -vpl enclosureX
```

- Solaris:

```
# vxdmpadm listenclosure all
```

or:

```
# luxadm probe
```

  - display information about the storage:

```
# luxadm display <storagearray name>
```

Display information about disks

- AIX:

```
# lqueryvg -p hdiskN -At
```

- Solaris:

```
# luxadm display /dev/rdsk/<disk>
```

Allow deny root logins

- AIX:

```
# /etc/security/user
```

- Solaris:

```
# /etc/default/login
```

Change IP address

- AIX:

```
# smitty chinet
```

or:

```
# chdev -a netaddr=<new-IP> -I <interface-name>
```

- Solaris:
  - permanent change:
    - write new IP in */etc/hosts*
    - write new default-router in */etc/defaultrouter*
    - reboot.
  - temporary change (without reboot):

```
# ifconfig <interface-name> down
# ifconfig <interface-name> <new-ip>
# ifconfig <interface-name> netmask <netmask>
# ifconfig <interface-name> broadcast +
# ifconfig <interface-name> up
```

Name resolution order

- AIX:

```
/etc/netsvc.conf
```

- Solaris:

```
/etc/nsswitch.conf
```

Administrative GUI

- AIX:

```
# smit
# smitty
# wsm (web smitty)
```

- Solaris:

```
# solstice
# admintool
# smc (solaris management console, 8 Ø1/Ø1+)
```

Partition a disk

- AIX:

  Partitions is done by creating lvs with mklv and displaying partitions in hdiskN:

```
# lspv -p hdiskN
```

- Solaris:

```
# format
```

Show/set kernel parameters

- AIX:

```
# /usr/samples/kernel/vmtune
```

  and:

```
# /usr/bin/no (network-related)
```

- Solaris:
  - show kernel parameters:

```
# sysdef
```

and:

```
# kstat
```

– Set kernel parameters in:

```
# vi /etc/system
```

and:

```
# adb -k
```

and:

```
# ndd
```

- eg to see which parameters are supported for TCP:

  ```
  # ndd /dev/tcp ?
  ```

- to see the current value for tcp_close_wait_interval:

  ```
  # ndd -get /dev/tcp tcp_close_wait_interval
  ```

- set the current value for tcp_close_wait_interval:

  ```
  # ndd -set /dev/tcp tcp_close_wait_interval <parameter>
  ```

Tape device

- AIX:

  ```
  /dev/rmt0
  ```

- Solaris:

  ```
  /dev/rmt/0
  ```

Automatic installation and recovery

- AIX:

  NIM – an excellent feature of the AIX operating system and very important for teams or companies that have a need to install or upgrade many RS/6000s.

- Solaris:

  Jumpstart – a system for automating the installation of new Solaris systems and a tool for disaster recovery of root diskgroup.

Check cluster member

- AIX:

```
# /usr/es/sbin/cluster/utilities/clnodename
```

- Solaris:

```
# /opt/SUNWcluster/bin/hastat
```

INFO.SH

Info.sh is a script that collects information from any Solaris machine. It enables the system administrator to see periodically whether something is wrong on the server.

```
#!/bin/ksh
#
# by Adnan Akbas Apr'2003
#

PATH=$PATH:/usr/bin:/usr/sbin
export PATH

#variables:
logfile=/tmp/info.'date +%d'
architec='uname -i'
sysname='uname -n'
clustermember=No

rm $logfile

#date:
print | tee -a $logfile
date | tee -a $logfile
print | tee -a $logfile

#output of showrev
/usr/bin/showrev | tee -a $logfile

#boot info
```

```
print Last 'who -b' - 'uptime | awk '{print $2 " " $3 " " $4}'' | tee -a
$logfile
print | tee -a $logfile

#hardware configuration:
print Num. of Sys. Boards ='/usr/platform/$architec/sbin/prtdiag -v |
grep POST | wc -l' | tee -a $logfile
print Num. of CPUs = 'uname -X | grep NumCPU | awk '{print $3}'' | tee -
a $logfile
print CPU Speed = 'psrinfo -v | grep MHz | tail -1 | awk '{print $6}''
MHz | tee -a $logfile
print Total Memory = '/usr/platform/$architec/sbin/prtdiag -v | grep
"Memory size" | awk '{print $3}'' | tee -a $logfile

z='pkginfo | grep "Sun Cluster" | wc -l'
if [ $z -ge 1 ]
then
        if [ -f /opt/SUNWcluster/bin/hastat ]
        then
                x='hastat | grep member | grep $sysname | head -1 | awk
'{print $1}''
                if [ $x = $sysname ]
                then
                clustermember=Yes
                fi
        fi
fi
print "Cluster Member ? =" $clustermember | tee -a $logfile
print | tee -a $logfile

#other info
eeprom | grep auto-boot | tee -a $logfile
eeprom | grep diag-level | tee -a $logfile
eeprom | grep scsi-initiator-id | tee -a $logfile
print | tee -a $logfile
set 'sar -k | grep Average'
kermem=$(($3 + $5 + $8))
kermem=$(($kermem / 1000000))
print "The kernel has grabbed $kermem Mbytes of the memory.(sar -k)" |
tee -a $logfile
sysdef | grep bufhwm | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#swapping
print "*SWAP events since boot: (vmstat -s)" | tee -a $logfile
print | tee -a $logfile
vmstat -s | head -4 | tee -a $logfile
print | tee -a $logfile
swap -s | tee -a $logfile
```

```
print | tee -a $logfile
print "*SWAP area status: (swap -l)" | tee -a $logfile
print | tee -a $logfile
swap -l | tee -a $logfile
print | tee -a $logfile
print "*SWAP-IN & SWAP-OUT now: (vmstat -S 1 5)" | tee -a $logfile
print | tee -a $logfile
vmstat -S 1 5 | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#network interfaces
print "*All UP network interfaces in the system : (ifconfig -au)" | tee
-a $logfile
print | tee -a $logfile
ifconfig -au | tee -a $logfile
print | tee -a $logfile
/bin/dmesg | grep Mbps | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#Collision rates:
print *Collision rates:
print | tee -a $logfile
x='netstat -i | grep -v Name | grep -v lo0 |  wc -l'
netstat -i | grep Name |awk '{print $1," ",$7," ",$9,"rate"}' | tee -a
$logfile
set -A interface 'netstat -i | grep -v Name | grep -v lo0 | awk '{print
$1}''
set -A opkts 'netstat -i | grep -v Name | grep -v lo0 | awk '{print
$7}''
set -A collis 'netstat -i | grep -v Name | grep -v lo0 | awk '{print
$9}''
i=0
while (($i<$x));do
opkts[i]=$((${opkts[i]} + 1))
rate=$((${collis[$i]} * 100 / ${opkts[$i]}))
echo "${interface[$i]} ${opkts[$i]}  ${collis[$i]}      %$rate" | tee -a
$logfile
i=$i+1
done
print | tee -a $logfile
print | tee -a $logfile

#Retransmission Rates
print *Retransmission rates: | tee -a $logfile
print | tee -a $logfile
netstat -s | grep tcpOutDataSegs | tee -a $logfile
netstat -s | grep tcpRetransSegs | tee -a $logfile
tcpOutDataSegs='netstat -s | grep tcpOutDataSegs | cut -d= -f2 | awk
```

```
'{print $1}''
tcpOutDataBytes='netstat -s | grep tcpOutDataSegs | cut -d= -f3 | awk
'{print $1}''
tcpRetransSegs='netstat -s | grep tcpRetransSegs | cut -d= -f2 | awk
'{print $1}''
tcpRetransBytes='netstat -s | grep tcpRetransSegs | cut -d= -f3 | awk
'{print $1}''
SegRate=$(($tcpRetransSegs * 100 / $tcpOutDataSegs))
ByteRate=$(($tcpRetransBytes * 100 / $tcpOutDataBytes))
print | tee -a $logfile
print "Retransmission Segs Rate: %$SegRate" | tee -a $logfile
print "Retransmission Bytes Rate: %$ByteRate" | tee -a $logfile
print | tee -a $logfile

#connection status:
print *Connection Status: | tee -a $logfile
print | tee -a $logfile
netstat -P tcp| awk '{print $7}' | sort| uniq -c | tail +3 | grep -v
Rwind | grep -v Remote | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#TCP tuning parameters
print *Some TCP tuning parameters: | tee -a $logfile
print | tee -a $logfile
close_wait='ndd /dev/tcp tcp_close_wait_interval'
print "tcp_close_wait_interval = $close_wait (Default 240 seconds)" |
tee -a $logfile
winsize_xmit='ndd /dev/tcp tcp_xmit_hiwat'
print "tcp_xmit_hiwat = $winsize_xmit (Default 8K)" | tee -a $logfile
winsize_recv='ndd /dev/tcp tcp_recv_hiwat'
print "tcp_recv_hiwat = $winsize_recv (Default 8K)" | tee -a $logfile
slow_start='ndd /dev/tcp tcp_slow_start_initial'
print "tcp_slow_start_initial = $slow_start (Default 1 package)" | tee -
a $logfile
print | tee -a $logfile
netstat -s | grep tcpListenDrop | tee -a $logfile
print | tee -a $logfile
pending_limit='ndd /dev/tcp tcp_conn_req_max_q'
print "tcp_conn_req_max_q = $pending_limit (Default 128)" | tee -a
$logfile
queue_limit='ndd /dev/tcp tcp_conn_req_max_q0'
print "tcp_conn_req_max_q0 = $queue_limit (Default 1024)" | tee -a
$logfile
print | tee -a $logfile

#free disks
print "*Free disk space in kbytes: (df -k)" | tee -a $logfile
print | tee -a $logfile
df -k | tee -a $logfile
```

```
print | tee -a $logfile
print | tee -a $logfile

#disk usage
print *Users disk usage in kbytes: | tee -a $logfile
print | tee -a $logfile
cut -d: -f6 /etc/passwd | sort | uniq | tail +2 | xargs du -sk | sort -
nr | head -1Ø | tee -a $logfile
print | tee -a $logfile

#disk status
vxdisk list > /dev/null 2>&1
if [ $? = Ø ]; then
print "*Disks Status: (vxdisk list)" | tee -a $logfile
print | tee -a $logfile
vxdisk list | tee -a $logfile
fi
print | tee -a $logfile
print | tee -a $logfile

#disk I/O load
print "*Disk I/O load:(iostat -xn) (must be %b < 5 , svc_t < 3Ø)" | tee
-a $logfile
print | tee -a $logfile
print "%b = %busy"
print "svc_t = average response times"
print | tee -a $logfile
iostat -xn | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#checking disk bottleneck
print "*Checking disk bottleneck:(vmstat) (must be: b <= r)" | tee -a
$logfile
print | tee -a $logfile
print "b = num. of blocked process" | tee -a $logfile
print "r = num. of process in the run queue" | tee -a $logfile
print | tee -a $logfile
vmstat 2 5 | awk '{print $1 " " $2}' | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#checking memory bottleneck
print "*Checking memory bottleneck:(vmstat) (must be: sr < 2ØØ , po =
Ø)" | tee -a $logfile
print | tee -a $logfile
print "sr = scan rate"
print "po = num. of Kbytes paged out per second"
vmstat 2 5 | awk '{print $9 " " $12}' | tee -a $logfile
print | tee -a $logfile
```

```
print | tee -a $logfile

#storage array enclosures and disks:
luxadm probe | grep SENA | cut -d: -f2 | awk '{print $1}' | xargs luxadm
display > /dev/null 2>&1
if [ $? = Ø ]; then
print *Storage Array enclosures and disks: | tee -a $logfile
print | tee -a $logfile
luxadm probe | grep SENA | cut -d: -f2 | awk '{print $1}' | xargs luxadm
display | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile
fi

#info about processors
print "*Info about preocessors: (psrinfo -v)" | tee -a $logfile
print | tee -a $logfile
psrinfo -v | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#per-processor performance statistics
print "*Per-processor performance statistics: (mpstat)" | tee -a
$logfile
print | tee -a $logfile
mpstat 2 3 | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#other
print "*DNLC cache hits:(vmstat) ( must be: cache hits > %9Ø)" | tee -a
$logfile
print | tee -a $logfile
vmstat -s | grep "cache hits" | tee -a $logfile
print | tee -a $logfile
print "*inode cache status: (netstat -k) (must be: maxsize > maxsize
reached)" | tee -a $logfile
print | tee -a $logfile
print "inode_cache:" | tee -a $logfile
netstat -k | grep maxsize | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#volumes
vxprint > /dev/null 2>&1
if [ $? = Ø ]; then
print "*Volume Status: (vxprint -Ath)" | tee -a $logfile
print | tee -a $logfile
vxprint -Ath | tee -a $logfile
fi
```

```
print | tee -a $logfile
print | tee -a $logfile

#system messages
print "*Alert System messages? : (/var/adm/messages)" | tee -a $logfile
print | tee -a $logfile
cat /var/adm/messages | grep NOTICE | tee -a $logfile
cat /var/adm/messages | grep WARNING | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#checking files
print *Files not accessed more than 3 months and greater than 100Mb : |
tee -a $logfile
print | tee -a $logfile
find / -type f -atime +90 -size +100000000c | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#checking sulog
print "*Last 15 su : (/var/adm/sulog)" | tee -a $logfile
print | tee -a $logfile
tail -15 /var/adm/sulog | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile


#Semaphores
print *IPC Semaphores and Shared Memory : | tee -a $logfile
print | tee -a $logfile
sysdef | grep SEM | tee -a $logfile
sysdef | grep SHM | tee -a $logfile
print | tee -a $logfile
print | tee -a $logfile

#prtdiag -v output:
print *prtdiag -v output:
print | tee -a $logfile
/usr/platform/$architec/sbin/prtdiag -v | tee -a $logfile
print | tee -a $logfile

print | tee -a $logfile
print by Adnan Akbas | tee -a $logfile
print | tee -a $logfile

#mail it!
mailx -s "DAILY INFO" adnan.akbas@turkcell.com.tr < $logfile
```

Adnan Akbas
Senior System Administrator
TURKCELL (Germany)

Recover a deleted file in AIX with JFS2 filesystems

In the article 'Recover a deleted file in AIX' published in *AIX Update* issue 111 (January 2005) we showed how to recover a deleted file in AIX JFS filesystems. For AIX JFS2 filesystems the procedure differs slightly. This article will provide a similar procedure, but which applies to JFS2 filesystems.

I will use an example to explain the procedure. In this example, we will recover a file called *myfile*, in a JFS2 filesystem called */test*. The full path to the file is */test/mydir/myfile*, and the logical volume for the */test* filesystem is */dev/lv08*. Please note that you will be using this procedure at your own risk and it is always recommended that you do a back-up of your current filesystem for restore purposes in case of mistakes.

Unmount the filesystem as soon as possible in order to prevent any major updates or changes to the filesystem meta data and to prevent the deleted file blocks from being overwritten:

```
# umount  /test
```

Run the **fsdb** command on the logical volume */dev/lv08*. The command **fsdb** is a filesystem debug utility and is part of the *bos.rte.filesystems* fileset. The **fsdb** command has different interfaces for JFS filesystems and JFS2 filesystems. This example shows the **fsdb** subcommands for a JFS2 filesystem only. For more information about the **fsdb** command please consult the man pages.

```
# fsdb /dev/lv08

File System:                    /dev/lv08

File System Size:               65120   (512 byte blocks)
Aggregate Block Size:           4096
Allocation Group Size:          8192    (aggregate blocks)
```

The root inode for any filesystem is always 2. Switch to that inode in **fsdb** by running the **i 2** subcommand:

```
> i 2
Inode 2 at block 26, offset 0x400:

[1] di_fileset:          16           [14] di_inostamp:        0x4192531f
[2] di_number:           2            [15] di_gen:             1
[3] di_size:     0x0000000000000100   [16] di_ixpxd.len:       4
[4] di_nblocks:  0x0000000000000000   [17] di_ixpxd.addr1:     0x00
[5] di_nlink:            4            [18] di_ixpxd.addr2:     0x0000001a
[6] di_mode:             0x000341ed              di_ixpxd.address:  26
                 0040755 drwxr-xr-x   [19] di_uid:             0
[7] di_ea.flag: 0x00                  [20] di_gid:             0
                                      [21]
di_atime.tj_sec:0x000000004192531f
[8] di_ea.nEntry:        0x00         [22] di_atime.tj_nsec:   0x00000000
[9] di_ea.len:           0            [23]
di_ctime.tj_sec:0x0000000041925339
[10] di_ea.addr1:        0x00         [24] di_ctime.tj_nsec:   0x25447f9e
[11] di_ea.addr2:        0x00000000   [25]
di_mtime.tj_sec:0x0000000041925339
     di_ea.address:      0            [26] di_mtime.tj_nsec:   0x25447f9e
[12] di_ea.type:         0x0000       [27]
di_otime.tj_sec:0x000000004192531f
                                      [28] di_otime.tj_nsec:   0x00000000
[13] di_ea.nblocks:      0            [29]
change_inode: [m]odify, [e]a, [t]ree, or e[x]it > x
```

You are in the inode subcommands now. You can exit by choosing **x**.

Get a directory listing on the filesystem by running the **dir** subcommand using the inode of the filesystem, ie **dir 2**:

```
> dir 2
 idotdot = 2

 3       lost+found
 32      mydir
```

The **mydir** subdirectory is at inode 32. Switch to inode 32 by running the **i 32** subcommand:

```
> i 32
Inode 32 at block 32, offset 0x0:

[1] di_fileset:          16           [14] di_inostamp:        0x4192531f
[2] di_number:          32            [15] di_gen:             1395618867
[3] di_size:     0x0000000000000100   [16] di_ixpxd.len:       4
[4] di_nblocks:  0x0000000000000000   [17] di_ixpxd.addr1:     0x00
[5] di_nlink:            2            [18] di_ixpxd.addr2:     0x00000020
```

```
[6] di_mode:            0x000141ed                  di_ixpxd.address:
32
                0040755 drwxr-xr-x   [19] di_uid:            0
[7] di_ea.flag: 0x00                 [20] di_gid:            0
                                     [21]
di_atime.tj_sec:0x0000000041925339
[8] di_ea.nEntry:       0x00         [22] di_atime.tj_nsec:  0x25447f9e
[9] di_ea.len:          0            [23]
di_ctime.tj_sec:0x000000004192535d
[10] di_ea.addr1:       0x00         [24] di_ctime.tj_nsec:  0x1e60ad2c
[11] di_ea.addr2:       0x00000000   [25]
di_mtime.tj_sec:0x000000004192535d
     di_ea.address:     0            [26] di_mtime.tj_nsec:  0x1e60ad2c
[12] di_ea.type:        0x0000       [27]
di_otime.tj_sec:0x0000000041925339
                                     [28] di_otime.tj_nsec:  0x25447f9e
[13] di_ea.nblocks:     0            [29]
change_inode: [m]odify, [e]a, [t]ree, or e[x]it > x
```

You are in the inode subcommands now. You can exit by choosing **x**.

In this directory, *myfile* can be found. We need to find out at what offset in the directory this file name occurs. The **display** subcommand can be used with directory inode to show this information in ASCII format, ie **display 32 a**:

```
> display 32 a
      …
000000ca:  00000000 00000000 00000000 00000000   |................|
000000da:  00000000 00000000 00000000 00020000   |................|
000000ea:  00000000 00008300 08010000 00000100   |................|
000000fa:  00000000 00000200 00000000 0021FF06   |.............!..|
-hit enter for more-
0000010a:  6D796669 6C650000 00000000 00000000   |myfile..........|
0000011a:  00000000 00000300 00000000 00000000   |................|
0000012a:  00000000 00000000 00000000 00000000   |................|
0000013a:  00000000 00000400 00000000 00000000   |................|
0000014a:  00000000 00000000 00000000 00000000   |................|
0000015a:  00000000 00000500 00000000 00000000   |................|
0000016a:  00000000 00000000 00000000 00000000   |................|
0000017a:  00000000 00000600 00000000 00000000   |................|
0000018a:  00000000 00000000 00000000 00000000   |................|
…
```

You might need to press *Enter* a few times to scroll down to the specific page where the deleted file is located because the **display** subcommand provides a pausing mechanism. The

output above is just part of the actual full output. This pausing feature prevents the use of the **grep** command on the file name from the command line, because **grep** will search only the first page of the display subcommand output.

The previous output shows that the string ‘myfile’ starts at 0000010a in this particular inode. The name of the file is also displayed in hex, 6D7966696C65, at the same offset 0000010a. The previous bytes (0021FF06 at offset 000000fa on the previous line) contain vital information about the file including the inode. Look at the first two bytes (0021), which represent the inode number of this file in hex. Hex 21 in decimal is 33.

Switch to the inode that we discovered in the previous step, which is 33, by entering the **i 33** subcommand in **fsdb**:

```
> i 33
Inode 33 at block 32, offset 0x200:

[1] di_fileset:          16           [14] di_inostamp:        0x4192531f
[2] di_number:           33           [15] di_gen:             1395618868
[3] di_size:     0x0000000000005e69   [16] di_ixpxd.len:       4
[4] di_nblocks: 0x0000000000000006    [17] di_ixpxd.addr1:     0x00
[5] di_nlink:            0            [18] di_ixpxd.addr2:     0x00000020
[6] di_mode:             0x000281a4                 di_ixpxd.address:  32
                         0100644 -rw-r--r--          [19] di_uid:
0
[7] di_ea.flag: 0x00                  [20] di_gid:             0
                                      [21]
di_atime.tj_sec:0x0000000041925347
[8] di_ea.nEntry:        0x00         [22] di_atime.tj_nsec:   0x12ca843d
[9] di_ea.len:           0            [23]
di_ctime.tj_sec:0x000000004192535d
[10] di_ea.addr1:        0x00         [24] di_ctime.tj_nsec:   0x1e60ad2c
[11] di_ea.addr2:        0x00000000   [25]
di_mtime.tj_sec:0x0000000041925347
     di_ea.address:      0            [26] di_mtime.tj_nsec:   0x1493fbd5
[12] di_ea.type:         0x0000       [27]
di_otime.tj_sec:0x0000000041925347
                                      [28] di_otime.tj_nsec:   0x12ca843d
[13] di_ea.nblocks:      0            [29]
change_inode: [m]odify, [e]a, [t]ree, or e[x]it > m
```

We can see the original permissions (-rw-r—r—), the owner (root uid 0), the group (system gid 0), and the size of the file in hex (5e69 = 24169 bytes). Notice the fifth field, *di_nlink: 0*.

This is the link count, and it is zero. To recover this file we need to set the link count to 1 by choosing **m** to modify, then choosing the field number 5, and specifying the new value of 1:

```
change_inode: [m]odify, [e]a, [t]ree, or e[x]it > m
Please enter: field-number value > 5 1
Inode 33 at block 32, offset 0x200:

[1] di_fileset:          16           [14] di_inostamp:        0x4192531f
[2] di_number:           33           [15] di_gen:             1644800058
[3] di_size:     0x0000000000005e69   [16] di_ixpxd.len:       4
[4] di_nblocks: 0x0000000000000006    [17] di_ixpxd.addr1:     0x00
[5] di_nlink:            1            [18] di_ixpxd.addr2:     0x00000020
[6] di_mode:             0x000281a4           di_ixpxd.address:  32
                 0100644 -rw-r--r--   [19] di_uid:             0
[7] di_ea.flag: 0x00                  [20] di_gid:             0
                                      [21]
di_atime.tj_sec:0x0000000041926227
[8] di_ea.nEntry:        0x00         [22] di_atime.tj_nsec:   0x3045d7a8
[9] di_ea.len:           0            [23]
di_ctime.tj_sec:0x000000004192627f
[10] di_ea.addr1:        0x00         [24] di_ctime.tj_nsec:   0x202430d8
[11] di_ea.addr2:        0x00000000   [25]
di_mtime.tj_sec:0x0000000041926227
     di_ea.address:      0            [26] di_mtime.tj_nsec:   0x30d4b3b0
[12] di_ea.type:         0x0000       [27]
di_otime.tj_sec:0x0000000041926227
                                      [28] di_otime.tj_nsec:   0x3045d7a8
[13] di_ea.nblocks:      0            [29]
change_inode: [m]odify, [e]a, [t]ree, or e[x]it > x
```

Notice how the *di_nlink* field changed to '1'. Now you can exit by choosing **x**.

Running the **fsck** command on */dev/lv08* should recover the file into the *lost+found* directory; therefore, make sure that the *lost+found* directory exists in the filesystem before running the **fsck** command:

```
# fsck /dev/lv08

****************
The current volume is: /dev/lv08
**Phase 1 - Check Blocks, Files/Directories, and Directory Entries
**Phase 2 - Count links
**Phase 3 - Duplicate Block Rescan and Directory Connectedness
**Phase 4 - Report Problems
```

```
**Phase 5 - Check Connectivity
**Phase 6 - Perform Corrections
1 file reconnected to /lost+found/.
**Phase 7 - Rebuild File/Directory Allocation Maps
Errors detected in the file system inode allocation
         map control information.
Errors detected in the file system inode allocation map.
**Phase 8 - Rebuild Disk Allocation Maps
    32768 kilobytes total disk space.
        3 kilobytes in 6 directories.
       25 kilobytes in 2 user files.
    32392 kilobytes are available for use.
File system is clean.
All observed inconsistencies have been repaired.
```

Mount */test* filesystem (*/dev/lv08*), and the file *myfile* is recovered into a file in */test/lost+found* called by the name of the inode, which in our case is 33:

```
# mount /test
# ls -l /test/lost+found
total 48
-rw-r—r—   1 root     system        24169 Nov 1Ø 13:47 33
```

Finally, the file can now be renamed to its original name */test/mydir/myfile*:

```
# mv /test/lost+found/33  /test/mydir/myfile
```

If you deleted more than one file, you will have to repeat this procedure for each file you need to recover.

Basim Chafik
Senior Systems Analyst
IBM Certified Advanced Technical Expert (CATE)
Plexus (Division of BancTec) (Canada)

The Andrew Filesystem (AFS) – part 2

This month we conclude the article looking at how to use AFS (the Andrew Filesystem).

CREATE_INITIAL_VOLS.SH

The next step is to create the initial volumes and mountpoints.

```
#!/bin/ksh
#
# Adnan Akbas, Turkcell, Ø6.Ø4.2ØØ4
#
# This script creates the wanted initial volumes and mountpoints
# one time execution

# variables ###################################################################

version=2.Ø
afs_path=/usr/afs/bin
scr_path=/usr/afs/scripts
afs_param=$scr_path/afs_vars_act.dat
root_quota=5ØØØ  # quota of the root directories ( 5MB )

# main ########################################################################

echo "\n\nINFO: $(date +"%H:%M %d.%m.%Y") starting script $Ø Version
$version ..."

## get token

if [ ! -f $scr_path/get_token.sh ]
then
   echo "ERROR: can't find the file $scr_path/get_token.sh , exit 1"
   exit 1
fi

$scr_path/get_token.sh
if [ $? != Ø ]
then
   echo "ERROR: wrong return code from get_token.sh, exit 2 "
   exit 2
fi

for myvol in $(grep afsquota $scr_path/create_vars.sh|grep "in kB" | cut
-d")" -f1 )
do
   echo "INFO: myvol -> $myvol"
   $scr_path/create_vars.sh $myvol
   if [ $? != Ø ]
   then
      echo "ERROR: wrong return code from create_vars.sh !!"
      exit 3
   fi
   if [ ! -f $afs_param ]
```

```
   then
      echo "ERROR: parameterfile -> $afs_param not found!!!"
      exit 4
   fi

# Load Parameters
   echo "INFO: load parameterfile param.dat"
   . $afs_param

   # check RO mountpoint is ok
   ls -l $afsro1 > /dev/null 2>&1
   if [ $? -eq 0 ]
   then
      echo "INFO: mountpoint $afsro1 already excists ."
   else
      echo "INFO: create vol $myvol ..."
      echo "INFO: $afs_path/vos create ${afsserver[0]} $afspart $myvol"
      $afs_path/vos create ${afsserver[0]} $afspart $myvol
      if [ $? != 0 ]
      then
         echo "ERROR: cannot create $myvol !!"
         exit 5
      fi

      # make mount
      echo "INFO: $afs_path/fs mkmount $afsrw1 $myvol"
      $afs_path/fs mkmount $afsrw1 $myvol
      if [ $? != 0 ]
      then
         echo "ERROR: cannot mkmount $myvol !!"
         exit 6
      fi

      # set access rights
      echo "INFO: $afs_path/fs setacl $afsrw1 $other_user $other_perm"
      echo "INFO: $afs_path/fs setacl $afsrw1 $adm_user $adm_perm"
      $afs_path/fs setacl $afsrw1 $other_user $other_perm
      if [ $? != 0 ]
      then
         echo "ERROR: cannot setacls for $afsrw1 !!"
         exit 7
      fi
      $afs_path/fs setacl $afsrw1 $adm_user $adm_perm
      if [ $? != 0 ]
      then
         echo "ERROR: cannot setacl $afsrw1/$afsvol1mnt !!"
         exit 8
      fi

      # set AFS volume size (Quota=0 -> unlimited)
```

```
      echo "INFO: $afs_path/fs setquota $afsrw1 $root_quota"
      $afs_path/fs setquota $afsrw1 $root_quota
      if [ $? != Ø ]
      then
         echo "ERROR: cannot setquota for $afsrw1 !!"
         exit 9
      fi

      # set replication sides: loop over all servers
      # the assumption is that all correlated RØ are on the same
partition
      for myserver in $(echo ${afsserver[*]})
      do
         echo "INFO: $afs_path/vos addsite $myserver $afspart $myvol"
         $afs_path/vos addsite $myserver $afspart $myvol
         if [ $? != Ø ]
         then
            echo "ERROR: cannot addsite RO volume $myvol !!"
            exit 1Ø
         fi
      done

      # write new Volume to Volume-history File
      echo "INFO: new volume created : $myvol" >> $afsrw1/Volume_history

      # release new replicas
      echo "INFO: $(date) $afs_path/vos $myvol release..."
      $afs_path/vos release $myvol
      if [ $? != Ø ]
      then
         echo "ERROR: cannot release $myvol !!"
         exit 11
      fi
      echo "INFO: $(date) $afs_path/vos $myvol released !"
      echo "INFO: $(date) $afs_path/vos root.cell release..."
      $afs_path/vos release root.cell
      if [ $? != Ø ]
      then
         echo "ERROR: cannot release root.cell !!"
         exit 12
      fi
      echo "INFO: $(date) $afs_path/vos root.cell releaseed !"

      # check for success in /afs : RO
      if [ $(ls $afsro 2>&1 | grep -cw $myvol) -ne 1 ]
      then
         echo "ERROR: new mountpoint not available:  $afsro/$myvol"
         exit 13
      else
         echo "INFO: new Mountpoint created: $afsro/$myvol"
```

```
      fi

      # check for success in /afs : RW
      if [ $(ls $afsrw 2>&1 | grep -cw $myvol) -ne 1 ]
      then
         echo "ERROR: new Mountpoint not available: $afsrw/$myvol"
         exit 14
      else
         echo "INFO: new Mountpoint created: $afsrw/$myvol"
      fi

      # check for success in vldb
      $afs_path/vos listvldb $myvol
      if [ $? != Ø ]
      then
         echo "ERROR: new volume not available in vldb: $myvol"
         exit 15
      fi

      # get entries from vos examine command
      index_rw=$(vos examine $myvol | grep server | grep "RW Site" | awk
' { print $2 } ')
      set -A index_ro $(vos examine $myvol | grep server | grep "RO
Site" | awk ' { print $2 } ')

      ## Server hosting RW volume
      if [ $($afs_path/vos listvol $index_rw | grep $myvol | grep RW |
grep -c "On-line") -ne 1 ]
      then
         echo "ERROR: new RW volume $myvol not available on : $index_rw"
         exit 16
      else
         echo "INFO: new RW volume $afsvol1 available on : $index_rw"
      fi

      # check for correct number and location of RW and RO volumes
      # loop over all RO hosting servers
      for ros in $(echo ${index_ro[*]})
      do
         if [ $($afs_path/vos listvol $ros | grep $myvol | grep RO |
grep -c "On-line" ) -ne 1 ]
         then
            echo "ERROR: new RO volume $myvol not available on : $ros"
            exit 17
         else
            echo "INFO: new RO volume $myvol available on : $ros"
         fi
      done

      # check acl for user
```

```
      if [ $($afs_path/fs listacl $afsrw1 | grep -c "${adm_user}
${adm_perm}" ) -ne 1 ]
      then
         echo "ERROR: AFS acl for new volume  $myvol, user ${adm_user}
not correct. "
         exit 18
      else
         echo "INFO: AFS acl for new volume $myvol , user ${adm_user} is
ok. "
      fi

      # check acl for other_user
      if [ $($afs_path/fs listacl $afsrw1 | grep -c "${other_user}
${other_perm}" ) -ne 1 ]
      then
         echo "ERROR: AFS acl for new volume $myvol , user ${other_user}
not correct. "
         exit 19
      else
         echo "INFO: AFS acl for new volume  $myvol , user ${other_user}
is ok."
      fi
   fi
done

# Kill AFS token
if [ ! -f $scr_path/kill_token.sh ]
then
   echo "ERROR: can't find the file $scr_path/kill_token.sh , exit 20"
   exit 20
fi
$scr_path/kill_token.sh
if [ $? != 0 ]
then
   echo "ERROR: wrong return code from kill_token.sh, exit 2 "
   exit 2
fi

echo "INFO: $(date +"%H:%M %d.%m.%Y")  ending script $0 "
```

In this way, if we create a new volume, we keep the last two old volumes (with a timestamp) as back-up. This allows us to get back very quickly to the old version if something goes wrong with the new one.

RM_VOLS.SH

We can remove old volumes with rm_vols.sh.

```
#!/bin/ksh
#
# Adnan Akbas, Turkcell, 08.04.2004
#
# This script removes the old AFS Volumes and mountpoints
#

# variables ###################################################

version=1.0
volinput=$(cat /usr/afs/scripts/volinput.dat)
afs_path=/usr/afs/bin
scr_path=/usr/afs/scripts
afs_param=$scr_path/param.dat

# main ########################################################

echo "\n\nINFO: $(date +"%H:%M %d.%m.%Y") starting script $0 Version
$version ..."

if [ ! -f $afs_param ]
then
   echo "ERROR: parameterfile -> $afs_param not found!!!"
   exit 1
fi

# Load Parameters
. $afs_param

# find oldest versions of example2xy_yymmdd_HHMM files/volumes
typeset -i no_vols=$(vos listvldb | grep -c ${myvol}_[0-9])
(( afsbackupmin = $afsbackupmin - 1 ))
if [ $no_vols -lt $afsbackupmin ]
then
   echo "INFO: less ($no_vols) than $afsbackupmin Files of type ${myvol}
found."
   echo "INFO: no file/volume will be deleted."
   echo "INFO: ... exit 0"
   exit 0
fi

typeset -i no_del_vol=${no_vols}-${afsbackupmin}

echo "INFO: $no_del_vol old volumes will be deleted !"

while [ $no_del_vol -gt 0 ]
do
   no_del_vol=$no_del_vol-1
   afsrmvol=$($afs_path/vos listvldb | grep ${myvol}_[0-9] | sort -r |
tail -1 | awk '{print $1}')
```

```
   afsrmnt=$afsrmvol
   echo "INFO: delete vol $afsrmvol ..."

# remove mountpoints for ex_abcdefg_yymmdd_HHMM files/volumes
   echo "INFO: $afs_path/fs rmmount $afsrw1/$afsrmnt"
   $afs_path/fs rmmount $afsrw1/$afsrmnt
   if [ $? != 0 ]
   then
      echo "ERROR: command fs rmmount $afsrw1/$afsrmnt failed ,exit 2
!!"
      exit 2
   fi

   # release related RO Volume one level higher
   # Volume name is myvol !!!
   echo "INFO: $afs_path/vos release $myvol"
   $afs_path/vos release $myvol

# check for success in /afs
   if [ $(ls $afsro1 | grep -c $afsrmnt) -ne 0 ]
   then
      echo "ERROR: mountpoint still available: $afsro1/$afsrmnt"
      exit 3
   else
      echo "INFO: mountpoint removed: $afsro1/$afsrmnt"
   fi

# delete RO Volume(s): loop over all servers
   vos listvldb $afsrmvol|grep "RO Site" | awk '{ print $2 }'|while read
server_ro
   do
      echo "INFO: $afs_path/vos remove -id $afsrmvol.readonly -server
$server_ro -verbose  (RO-Volume)"
      $afs_path/vos remove -id $afsrmvol.readonly -server $server_ro -
verbose
   done

# delete RW Volume
   echo "INFO: $afs_path/vos remove -id $afsrmvol -verbose (RW-Volume)"
   $afs_path/vos remove -id $afsrmvol -verbose

# check for success in vldb
   if [ $($afs_path/vos listvldb | grep -c $afsrmvol) -ne 0 ]
   then
      echo "ERROR: volume still available in vldb: $afsrmvol"
      exit 4
   else
      echo "INFO: volume removed in vldb: $afsrmvol"
   fi
done
```

```
echo "INFO: $(date +"%H:%M %d.%m.%Y")  ending script $Ø "
```

CREATE_TIMESTAMP_VOLS.SH

We can create volumes with timestamps (example_yymmdd_HHMM) using create_timestamp_vols.sh.

```
#!/bin/ksh
#
# Adnan Akbas, Turkcell, Ø5.Ø4.2ØØ4
#
# This script creates the example2xy_yymmdd_HHMM volume,
# with correct mountpoint and acl
#

# variables ###########################################################

version=1.Ø
volinput=$(cat  /usr/afs/scripts/volinput.dat)
afs_path=/usr/afs/bin
scr_path=/usr/afs/scripts
afs_param=$scr_path/param.dat

# main ################################################################

echo "\n\nINFO: $(date +"%H:%M %d.%m.%Y") starting script $Ø Version
$version ..."

if [ ! -f $afs_param ]
then
   echo "ERROR: parameterfile -> $afs_param not found!!!"
   exit 1
fi

echo "INFO: Load Parameterfile para.dat"

# Load Parameters
. $afs_param

# create extra timestamp volume
echo "INFO: $afs_path/vos create ${afsserver[Ø]} $afspart $afsvol1"
$afs_path/vos create ${afsserver[Ø]} $afspart $afsvol1
if [ $? != Ø ]
then
   echo "ERROR: cannot create $afsvol1 !!"
   exit 2
fi
```

```
# mounting
echo "INFO: $afs_path/fs mkmount $afsrw1/$afsvol1mnt $afsvol1"
$afs_path/fs mkmount $afsrw1/$afsvol1mnt $afsvol1
if [ $? != Ø ]
then
   echo "ERROR: command failed –> fs mkmount $afsrw1/$afsvol1mnt
$afsvol1 !!"
   exit 3
fi

# set access rights
echo "INFO: $afs_path/fs setacl $afsrw1/$afsvol1mnt $other_user
$other_perm"
echo "INFO: $afs_path/fs setacl $afsrw1/$afsvol1mnt $adm_user $adm_perm"
$afs_path/fs setacl $afsrw1/$afsvol1mnt $other_user $other_perm
if [ $? != Ø ]
then
   echo "ERROR: cannot setacls for $afsrw1/$afsvol1mnt !!"
   exit 4
fi

$afs_path/fs setacl $afsrw1/$afsvol1mnt $adm_user $adm_perm
if [ $? != Ø ]
then
   echo "ERROR: cannot setacl $afsrw1/$afsvol1mnt !!"
   exit 5
fi

# set AFS volume size (Quota=Ø -> unlimited)
echo "INFO: $afs_path/fs setquota $afsrw1/$afsvol1mnt $afsquota"
$afs_path/fs setquota $afsrw1/$afsvol1mnt $afsquota
if [ $? != Ø ]
then
   echo "ERROR: cannot setquota for $afsrw1/$afsvol1mnt !!"
   exit 6
fi

# set replication sides: loop over all servers
# the assumption is that all correlated RØ are on the same partition
for serv in $(echo ${afsserver[*]})
do
   echo "INFO: $afs_path/vos addsite $serv $afspart $afsvol1"
   $afs_path/vos addsite $serv $afspart $afsvol1
   if [ $? != Ø ]
   then
      echo "ERROR: cannot addsite RO volume $afsvol1 !!"
      exit 7
   fi
done
```

```
# write new Volume to Volume-history File
echo "new volume created : $afsvol1" >> $afsrw1/Volume_history
if [ $? != 0 ]
then
   echo "ERROR: cannot write to $afsrw1/Volume_history !!"
   exit 8
fi

# release new replicas
echo "INFO: $(date) $afs_path/vos $afsvol1 release..."
$afs_path/vos release $afsvol1
if [ $? != 0 ]
then
   echo "ERROR: cannot release $afsvol1 !!"
   exit 9
fi
echo "INFO: $(date) $afs_path/vos $afsvol1 released !"
echo "INFO: $(date) $afs_path/vos $myvol release..."
$afs_path/vos release $myvol
if [ $? != 0 ]
then
   echo "ERROR: cannot release $myvol !!"
   exit 10
fi
echo "INFO: $(date) $afs_path/vos $myvol releaseed !"

sleep 10

# check for success in /afs : RO
if [ $(ls 2>&1 $afsro1 | grep $afsvol1mnt | grep -vc "does not exist" )
-ne 1 ]
then
   echo "ERROR: new mountpoint not available:  $afsro1/$afsvol1mnt"
   exit 11
fi

# check for success in /afs : RW
if [ $(ls 2>&1 $afsrw1 | grep $afsvol1mnt | grep -vc "does not exist") -
ne 1 ]
then
   echo "ERROR: new Mountpoint not available: $afsrw1/$afsvol1mnt"
   exit 12
fi

# check for success in vldb
if [ $($afs_path/vos listvldb | grep -c $afsvol1 ) -ne 1 ]
then
   echo "ERROR: new volume not available in vldb: $afsvol1"
   exit 13
fi
```

```
# get entries from vos examine command
index_rw=$(vos examine $afsvol1 | grep server | grep "RW Site" | awk ' {
print $2 } ')
set -A index_ro $(vos examine $afsvol1 | grep server | grep "RO Site" |
awk ' { print $2 } ')

# Server hosting RW volume
if [ $($afs_path/vos listvol $index_rw | grep $afsvol1 | grep RW | grep
-c "On-line") -ne 1 ]
then
  echo "ERROR: new RW volume $afsvol1 not available on : $index_rw"
   exit 14
fi

# check for correct number and location of RW and RO volumes
# loop over all RO hosting servers
for ros in $(echo ${index_ro[*]})
do
   if [ $($afs_path/vos listvol $ros | grep $afsvol1 | grep RO | grep -c
"On-line" ) -ne 1 ]
   then
      echo "ERROR: new RO volume $afsvol1 not available on : $ros"
      exit 15
   fi
done

# check acl for user
if [ $($afs_path/fs listacl $afsrw1/$afsvol1mnt | grep -c "${adm_user}
${adm_perm}" ) -ne 1 ]
then
   echo "ERROR: AFS acl for new volume  $afsvol1mnt, user ${adm_user}
not correct. "
   exit 16
fi

# check acl for user $other_user
if [ $($afs_path/fs listacl $afsrw1/$afsvol1mnt | grep -c "${other_user}
${other_perm}" ) -ne 1 ]
then
   echo "ERROR: AFS acl for new volume $afsvol1mnt , user ${other_user}
not correct. "
   exit 17
fi

echo "INFO: $(date +"%H:%M %d.%m.%Y")  ending script $Ø "
```

CREATE_NEW_VOLS.SH

The main script to create new volumes is also the main script

that calls all the other scripts, and it is called create_new_vols.sh.

```
#!/bin/ksh
#
# Adnan Akbas, Turkcell, 08.04.2004
#
# This script is the main script, which calls other scripts
# to create/delete new AFS volumes
#
# variables
##########################################################################

version=1.0
afs_path=/usr/afs/bin
scr_path=/usr/afs/scripts
afs_param=$scr_path/param.dat

# structure
# AFS_Volume_name               Mountpoint
# root.afs                      /afs
# root.cell                     /afs/.prodcell_rw
# example                       /afs/.prodcell_rw/example
# example                       /afs/.prodcell_rw/example
# example_yymmdd_HHMM           /afs/.prodcell_rw/example/
example_i_yymmdd_HHMM
# abcdefgh_1_yymmdd_HHMM        10 characters are available
# 123456789012345678912         not more than 22 characters to name a
volume

# main
##########################################################################

echo "\n\nINFO: $(date +"%H:%M %d.%m.%Y") starting script $0 Version
$version ..."

## check the number of arguments
if [ $# -ne 1 ]
then
   echo "ERROR: The number of arguments is not correct ! , exit 1 "
   exit 1
fi

# Assign argument to variable

myvol=$1

echo "INFO: create the TIMESTAMP file (/usr/afs/scripts/time_stamp.dat)"
echo $(date +"%y%m%d_%H%M") > /usr/afs/scripts/time_stamp.dat
```

```
# get the input argument and write it to an temporary file
echo ${myvol} > /usr/afs/scripts/volinput.dat

# create variable file
${scr_path}/create_vars.sh ${myvol}
rc=$?
if [ $rc -ne Ø ]
then
   echo "ERROR: create_vars.sh has wrong returncode"
   exit $rc
fi

if [ ! -f $afs_param ]
then
   echo "ERROR: parameterfile -> $afs_param not found!!!"
   exit 2
fi

# Load Parameters
echo "INFO: load parameterfile param.dat"
. $afs_param

# get token
echo "INFO: call $scr_path/get_token.sh ... "
$scr_path/get_token.sh
rc=$?
if [ $rc -ne Ø ]
then
   echo "ERROR: ... exit with RC $rc "
   exit $rc
fi

# remove oldest volumes example_yymmdd_HHMM
echo "INFO: call $scr_path/rm_vols.sh"
$scr_path/rm_vols.sh
rc=$?
if [ $rc -ne Ø ]
then
   echo "ERROR: rm_vols.sh exit with RC $rc "
   exit $rc
fi

# create new example2xy_yymmdd_HHMM volume
echo "INFO: call $scr_path/create_timestamp_vols.sh"
$scr_path/create_timestamp_vols.sh
rc=$?
echo "INFO: Returncode from script: $scr_path/create_timestamp_vols.sh =
$rc"
if [ $rc -ne Ø ]
then
```

```
   echo "ERROR: create_timestamp_vols.sh exit with RC $rc "
   exit $rc
fi

# Kill token
echo "INFO: call $scr_path/kill_token.sh"
$scr_path/kill_token.sh
rc=$?
if [ $rc -ne Ø ]
then
   echo "ERROR: kill_token.sh exit with RC $rc "
   exit $rc
fi

echo "INFO: $(date +"%H:%M %d.%m.%Y")  ending script $Ø "
```

Adnan Akbas
Senior System Administrator
TURKCELL (Germany)

The *Update* family

In addition to *AIX Update*, the Xephon family of *Update* publications now includes *CICS Update*, *DB2 Update*, *MQ Update*, *MVS Update*, *RACF Update,* and *TCP/SNA Update*.

Details of all of these can be found on the Xephon Web site at www.xephon.com, or you can contact Xephon Inc at PO Box 550547, Dallas, TX 75355, USA. Our phone number is (214) 340 5690, the fax number is (214) 341 7081, and the e-mail address to use is info@xephon.com.

Articles should be sent to the editor, Trevor Eddolls, at trevore@xephon.com.

AIX news

XOsoft has announced an enhanced version of its WANSync Server application availability software platform optimized specifically for AIX-based systems.

The new WANSync Server for AIX is a real-time data synchronization and replication software solution that enables system administrators to protect the integrity and availability of entire AIX-based corporate information repositories across multiple servers around the world, without interrupting their organizations' operations, claims the company. WANSync Server ensures seamless availability by synchronizing entire servers and constantly replicating them to either local or remote replicas in real time. Should an AIX server fail for any reason, a WANSync Server replica will take its place.

For further information contact:
URL: www.xosoft.com/press/f_pr011005.shtml.

* * *

Xenos Group has announced Version 2.0 of d2e Vision, a document enhancement solution that can manage, archive, and repurpose documents for use in print or Web applications. New components allow for processing Adobe PDF 1.4 and TIFF input files and generating indexes for electronic content management, archive, and database applications. The Java-based enterprise solution is available for AIX, HP-UX, Linux, Solaris, z/OS, and Windows NT/2000/XP, and has multi-language support.

For further information contact:
URL: www.xenos.com/solutions/prod_d2e_vision_overview.htm.

* * *

TenFold has announced EnterpriseTenFold MarketForce, its new platform for building and implementing enterprise applications

The product includes XML BrowserClient, which separates static, transaction user-interface descriptions from transaction data and attributes using XML. It caches static transaction definition XML so that browsers request it once for the lifetime of a transaction definition, and reduces network traffic to mostly transaction data and attributes.

MultiLanguage allows administrators to automatically translate a TenFold-powered application into another language to make the application available to end-users in that language.

For further information contact:
URL: www.tenfold.com/.

* * *

Four new vulnerabilities have been reported in AIX. A boundary error in the paginit utility when handling the first command line argument can be exploited to cause a stack-based buffer overflow by passing an overly long string. When invoking the **/bin/Dctrl** utility, the diag script prefixes the path with the content of the DIAGNOSTICS environment variable. When invoking the **uname** utility, the lsvpd program uses the path information in the PATH environment variable. This can be exploited via the invscout program to execute an arbitrary program by manipulating the PATH environment variable. When invoking the **grep** utility, the chcod program uses the path information in the PATH environment variable. This can be exploited by users in the system group to execute an arbitrary program by manipulating the PATH environment variable.

For further information contact:
URL: /secunia.com/advisories/13589.