Tru64 UNIX Backup & System Maintenance Scripts

Jenny Butler

Senior System Analyst

University of Tennessee Memphis

HP WORLD 2003
Solutions and Technology Conference & Expo

Tru64 UNIX Tools for Backups & Device Handling

- dump, vdump, rvdump
- restore, vrestore, rvrestore
- cpio, pax
- tar
- dd
- mt
- robot (Media Robot Utility)

dump, vdump, rvdump

Used for file system backups

dump – for non-AdvFS file systems – restore

vdump – for AdvFS file systems – vrestore

rvdump – remote AdvFS – rvrestore

Supports full and multilevel incremental backups

-0 – full backup of file system

-1-9 – incremental backup since last backup of lower level

-u updates /etc/vdumpdates with last backup date for file system at that level (number)

restore, vrestore, rvrestore

Used to restore dump, vdump backups

vrestore –D restores files to different location than where they were at backup

vrestore –i enter interactive mode where user may choose specific files to restore using add

vrestore –v (verify mode) lists files as they are restored

cpio, pax, tar

- cpio – useful for archiving small number of files
  - "legacy" utility, compatible with pax
- pax – extracts, writes, and lists archive files
  - use –a to append to archive on supported devices
  - may specify cpio or extended tar format for output archive file
- tar – builds archive file – can zip a tar file to compress it further
  - need extra space to untar and extract files

dd

- Can use dd to move files to tape
- Can specify blocking factor
- Can erase (zero) a tape using dd

  dd if=/dev/zero of=/dev/tape/tape0_d0 bs=64
- Can convert between ascii, ebcdic, fixed or

  variable length records, upper or lower case

mt

Magnetic tape manipulation program

mt –f /dev/ntape/tape0 rewind

mt –f /dev/tape/tape0 status

mt –f /dev/tape/tape1 unload

mt –f /dev/ntape/tape1 rdpos s - show position of tape

mt –f /dev/ntape/tape1 fsf 2 - skip forward 2 files

Media Robot Utility

MRU commands

robot show robot – shows model, # drives, # slots

robot show drives – shows drives and whether empty or tape label

robot show slots – lists tapes in slots (reads barcode labels on tape cartridges)

robot initialize robot /dev/changer/mc0 – inventories drives and slots for device

robot load drive 0 slot 3

robot unload drive 1 slot 6

Current version at www.support.compaq.com/sms/mru/

Script choices

- sh ($) – Bourne shell is default for root
  - Simple, easy to program
  - Missing some more complex constructs
- csh (%) – C shell
  - Designed for interactive users
- ksh ($) – Korn shell
  - Combines ease of C shell and programming features of Bourne shell – superset of Bourne shell
  - Any Bourne shell script will run with Korn shell
- bash shell – freeware shell

More script choices

- perl – "Practical Extraction and Report Language"
  - More like programming
  - Portable
  - Resources at *www.cpan.org*
- expect – "a program to control interactive applications"
  - Can download from *www.cpan.org*
  - Complex constructs to emulate interactive responses to utilities for errors or end of tape
  - Download and install tcl and tk as well – "tool command language"

Getting started

- Choose the shell script or programming tool
- First line of script should indicate choice:

  #!/bin/ksh – Korn shell

  #!/usr/bin/perl – perl

  #!/usr/local/bin/expect –f - expect

  -f flag means read commands from the file
- When script is ready to run, remember to

  chmod +x ./myscript to make it executable
- Test expect using debugger with

  expect –D script arguments

Simple vdump script

```
#!/bin/ksh
TODAY=`date+"%Y_%b_%d"`
LOG=/home/backup/test-$TODAY.log
print "\nBegin backup - $(/bin/date)" > $LOG
vdump –0 –NUu –f /dev/ntape/tape0_d1 / >> $LOG 2>&1
vdump –0 –NUu –f /dev/ntape/tape0_d1 /usr >> $LOG 2>&1
vdump –0 –NUu –f /dev/ntape/tape0_d1 /var >> $LOG 2>&1
print "\nEnd backup - $(/bin/date)" >> $LOG
mt –f /dev/ntape/tape0_d1 rewind >> $LOG
mt –f /dev/ntape/tape0_d1 unload >> $LOG
mailx –s "Backup done" –r root@mysys jb@mysys < $LOG
```

Expect script

```
#!/bin/expect –f
set timeout –1
spawn $env(SHELL)
match_max 100000
log_file –noappend /home/tools/test-file.log
send_log "starting expect script now – [exec date] \r\n"
set mdir [lindex $argv 0]
send_log "[lindex $argv 0] is the directory \r\n"
set xlev [lindex $argv 1]
send_log "[lindex $argv 1] is the backup level \r\n"
#
```

Expect script continued

```
send_log "begin vdump process – [exec date] \r\n"
send -- "/sbin/vdump -${xlev} –NUu –f /dev/ntape/tape0_d1
          ${mdir}\r"
while {1} {
  expect {
    -re "(vdump: Dump completed)" {
      send_log "finished – [exec date] \r\n"
      exit 0
     }
    -re "(vdump: do you want to retry?)" {
       send -- "no\r"
```

Expect script continued

```
        send_log "vdump error – [exec date] \r\n"
        exit 1
      }
      default {
        exp_continue
      }
    }
}
```

Calling expect script

```
#!/bin/ksh
logfile=/home/tools/backup.log
print "Starting backup script $(/bin/date)" > $logfile
#
/home/tools/test-bkup /usr 0
cat /home/tools/test-file.log >> /home/tools/backup.log
/home/tools/test-bkup /var 2
cat /home/tools/test-file.log >> /home/tools/backup.log
#
print "Finished backup script $(/bin/date)" >> $logfile
#
```

MRU example

```
#!/bin/ksh
MRU_ROBOT=/dev/changer/mc0
export MRU_ROBOT
#  Defined tape functions in another script file
. /home/tools/tape_functions.sh
select_tape
if [ "${TAPE}" = "" ]
  then
        exit 1
fi
```

MRU example continued

```
load_tape $TAPESLOT $TAPEDRIVE
if [ "${TAPEID}" != "${DRIVEID}" ]
  then
        echo "Unable to load tape. Exiting"
        exit 1
fi
vdump –0 –NUu –f $TAPE /home
unload_tape $TAPEDRIVE $TAPESLOT
exit
```

tape_functions.sh

```
#!/bin/ksh
select_tape() {
TAPEDRIVE=`robot show drives | grep "Empty$" | head –1 |
        awk '{print $2}'`
if [ "$TAPEDRIVE" = "" ]
  then
    echo "No empty tape drive found"
    $TAPE=""
    $TAPESLOT=""
    $TAPEID=""
  else
```

tape_functions.sh continued

```
TAPE="/dev/ntape/tape${TAPEDRIVE}_d1"
WEEKDAY=`date +%Ou`
case $WEEKDAY in
 1|2|3|4|5|6|7)
    TAPESLOT=$((${WEEKDAY}-1))
    ;;
 *)
    TAPESLOT=6
    ;;
esac
TAPEID=`robot show slot ${TAPESLOT} | awk '{print $3}'`
```

tape_functions.sh continued

```
   echo "Using ${TAPESLOT} (${TAPEID}) in ${TAPEDRIVE}
fi
}
load_tape() {
robot load drive ${2} slot ${1}
DRIVEID=`robot show drive ${2} | awk '{print $3}'`
#
unload_tape() {
robot unload drive ${1} slot ${2}
DRIVEID=""
}
```

Remote backup

```
#!/bin/ksh
DEST_NODE=xyz.abc.edu
DEST_DIR=/develop
BKUP_DATE=`date+"%Y_%b_%d"`
BKUP_LOG=/home/tools/dev-$BKUP_DATE.log
DEST_FILE=$DEST_DIR/dev-$BKUP_DATE.tar.gz
EMAIL="jenny@abc.edu,operator@abc.edu"
#
cd /home/develop
(gtar czf - | rsh $DEST_NODE dd of=$DEST_FILE) >
        $BKUP_LOG 2>&1
```

Remote backup continued

```
if [ $? = 0 ]
 then
   mailx –s "Success" –r root@abc.edu $EMAIL < $BKUP_LOG
 else
   mailx – s "Failure" –r root@abc.edu $EMAIL < $BKUP_LOG
fi
#
exit
```

Remote backup to tape

```
#!/bin/ksh
BKUP-DATE=`date+"%Y_%b_%d"`
LOG=/home/bkup/remote-$BKUP-DATE.log
rsh cub robot show robot /dev/changer/mc0 slots > $LOG
rsh cub robot load slot 5 drive 1 robot /dev/changer/mc0
rsh cub robot show robot /dev/changer/mc0 drives >> $LOG
rvdump –0 –NUu –f cub:/dev/ntape/tape1_d1 /home >>
                $LOG 2>&1
rsh cub mt –f /dev/ntape/tape1 rdpos s >> $LOG
rsh cub robot unload slot 5 drive 1 robot /dev/changer/mc0
#
```

Back up most recent files

```
mkdir /usr/local/Content/bkup

cd /usr/local/Content

touch –t `/usr/local/bin/date --date='60 days ago'
   +%C%y%m%d%H%M` marker.txt

ln `find . –type f –newer ./marker.txt` /usr/local/Content/bkup

ls –l /usr/local/Content/bkup

#

vdump –0 –NUu –f /dev/ntape/tape0_d1 –D
   /usr/local/Content/bkup >> $BKUP_LOG 2>&1

#

rm –r /usr/local/Content/bkup

rm /usr/local/Content/marker.txt
```

Clone AdvFS fileset

```
MOUNT=/mnt
BDOMAIN=mydata1
BFILESET=data1
BCLONE=data1_clone
#
clonefset $BDOMAIN $BFILESET $BCLONE
mount $BDOMAIN#BCLONE $MOUNT
vdump –0 –f /dev/tape/tape1_d1 /mnt
umount $MOUNT
rmfset –f $BDOMAIN $BCLONE
#
```

Defragment AdvFS fileset

```
#!/bin/ksh
now=$(date'+%m%d%H%M')
logfile=/home/info/defrag-$now.log
date > $logfile
#
df –k –t advfs | grep '#' | awk '{print substr($1,1,index($1,"#")-1)}' | \
while read fset
do
#
/usr/sbin/defragment –n –v ${fset} | grep Aggregate | grep "100%"
   >> $logfile
```

Defragment AdvFS fileset continued

```
if [[ $? != 0 ]]; then
  /usr/sbin/defragment –v ${fset} >> $logfile
  err=$?
if [$err != 0 ]
  then
    print "Error $err defragmenting ${fset}" >> $logfile
  fi
fi
#
done
```

Tru64 UNIX v5.1B vfast

- Introduced with v5.1B
- Runs in background
- May be activated, deactivated, suspended
- Display hot files and extents
- Balance across multivolume domains
- Replaces defragment utility
- Initially defaults to deactivated

Check disk storage

```
#!/bin/ksh
THRESH=85
EMAIL="jenny@mysite.edu,7011234567@msg.servc.com"
DF="/sbin/df –k –t advfs"
#
if [ -f /home/tools/disk_alert.${PPID} ]
then
    rm –f /home/tools/disk_alert.${PPID}
fi
#
${DF} | grep '%' | awk '{print $5 " " $1 " " $6}' | sort –n | \
while read usage device mountpoint
```

Check disk storage continued

```
do
  if [ ${usage%\%} –gt ${THRESH} ]
    then
     echo "Filesystem ${mountpoint} (${device})" \
          "at ${usage} utilization" \
           >> /home/tools/disk_alert.${PPID}
  fi
  done
if [ -f /home/tools/disk_alert.${PPID} ]
  then
```

Check disk storage continued

```
    mailx –s "Disks over Utilization on $(hostname –s)" \
        ${EMAIL} < /home/tools/disk_alert.${PPID}
        rm –f /home/tools/disk_alert.${PPID}
fi
```

Check & roll system logs

```
#!/bin/ksh
RUNDATE=`date +"%Y_%b_%d"`
RUNLOG=/home/backup/syslogs-$RUNDATE.log
EMAIL="jenny@mysys.edu"
cd /var/adm/syslog.dated/current
date > $RUNLOG
ls –l >> $RUNLOG
if [[ -s auth.log ]]; then
  grep authenticated auth.log >> $RUNLOG
  grep denied auth.log >> $RUNLOG
fi
```

Check & roll system logs continued

```
if [[ -s daemon.log ]]; then
  cat daemon.log >> $RUNLOG
fi
     (include your choice for kern.log, syslog.log, etc)
if [[ -s mail.log ]]; then
  grep "from=" mail.log >> $RUNLOG
fi
kill –HUP `cat /var/run/syslog.pid`
mailx –s "Syslogs" –r root@mysys.edu $EMAIL < $RUNLOG
find /home/backup –type f –name 'syslog*log' –ctime +29 –exec rm {} \;
exit
```

Run sys_check

```
#!/bin/ksh
now=$(date '+%b%d')
nname="`hostname –s`"
/usr/sbin/sys_check –nohtml –perf > ./$nname-perf-$now.log
find . –type f –name '*-perf-*log' –ctime +21 –exec rm {} \;
exit
```

Cluster check script

```
# Check section only
'check')
  PIDFILE=/usr/local/apache/logs/httpd.pid
  PROCFILE=/proc/`<$PIDFILE`
#
if [ -f $PIDFILE ]; then
  if [ -f $PROCFILE ]; then
   RPROC=$(ps –o pid,command,user –p $(cat $PIDFILE))
    echo "\"$RPROC\" is running"
  else
```

Cluster check script continued

```
    print "\"$PROBE_PROCS\" is not running" >> ${LOG}
    mailx –s "Apache check on truclus" ${EMAIL} <${LOG}
    exit 1
  fi
else
    print "\"$PROBE_PROCS\" may not be running" >> ${LOG}
    mailx –s "Apache check on truclus" ${EMAIL} <${LOG}
    exit 1
fi
exit 0
```

Printing menu

```
#!/usr/bin/perl –w
$SIG{INT} = \&trapit;
use Term::ReadKey;
while (1) {
  print "\n       Operator Print Menu\n\n";
  print " A – Check printer queue status\n";
  print " B – Show jobs in printer queue\n";
     (note other choices may be added)
  print " P – Reset password for operator account\n";
  print " X – Exit from account\n";
#
```

Printing menu continued

```
ReadMode 'cbreak';
 print "\n Enter choice: ";
 while (not defined ($input = ReadKey(900))) {
   print "\n";  printf("          %s\n", `date`);
   system("/home/tools/prlist | grep '[0-9] entr' | sort –k 2");
   print "\n Enter choice: ";
  }
$CHOICE = $input; print "$CHOICE\n\n";
 ReadMode 'normal';
#
if ($CHOICE =~ /[aA]/) {
```

Printing menu continued

```
 printf("         %s\n", `date`);
 system("/home/tools/prlist | sort –k 2");
 print "\n Enter <RETURN> to continue: ";
  chomp ($OK = <STDIN>);
} elsif ($CHOICE =~ /[bB]/ {
  {
   print "\n Enter queue to show jobs: ";
      chomp ($QUE = <STDIN>);
    last if $QUE ne "";
   print "\a\n You must enter a printer queue to show!\n";
     redo;
```

Printing menu continued

```
    print "\n";
    system("/usr/local/bin/sudo /usr/bin/lpq –P$QUE");
    print "\n Enter <RETURN> to continue: ";
      chomp ($OK = <STDIN>);
} elsif ($CHOICE =~ /[pP]/ {
   system("/bin/passwd");
   print "\n Enter <RETURN> to continue: ";
      chomp ($OK = <STDIN>);
} elsif ($CHOICE =~ /[xX]/ {
  exit
} else {
```

Printing menu continued

```
   print " Please enter valid choice – try again!\n\n"
   }
#
}
#
sub trapit {
  $SIG{INT} = \&trapit;
  warn "\aPlease do not interrupt with Ctrl-C\n";
  }
```

prlist script

```
#!/usr/bin/perl –w
my (@printers, @prdef, $lpque, $lpnam, $lpfrm);
open(PRINTCAP,"grep '|' /etc/printcap |");
@printers = <PRINTCAP>;  close PRINTCAP;
#
foreach (@printers) {
    @prdef = split(/\|/,$_);
    $lpque = $prdef[0];
    $lpfrm = $prdef[-2];
    $lpfrm =~ tr/A-Z/a-z/;
    $lpnam = $prdef[-1];
```

prlist script continued

```
    chomp($lpnam);
    $lpnam =~ s/:\\$//;
    open(LPSTATUS,
        "/usr/sbin/lpc status $lpque | grep -E 'entry|entries'|");
    $lpstatus = <LPSTATUS>;
    close LPSTATUS;
    chomp $lpstatus;
    $lpstatus =~ s/\t//g;
    printf("%-6s %-14s %-35s %s\n", $lpque, $lpfrm, $lpnam,
            $lpstatus);
};
```

Resources

- Tru64 UNIX Documentation Set
- O'Reilly Books:
  - Learning the Korn Shell
  - Exploring Expect
  - Learning Perl
  - Perl Cookbook
  - Programming Perl
- Documentation on cpan web site – www.cpan.org
- Tru64 Managers list – Tru64-UNIX-Managers@ornl.gov - (subscribe at majordomo@ornl.gov)

Interex, Encompass and HP bring you a powerful new HP World.