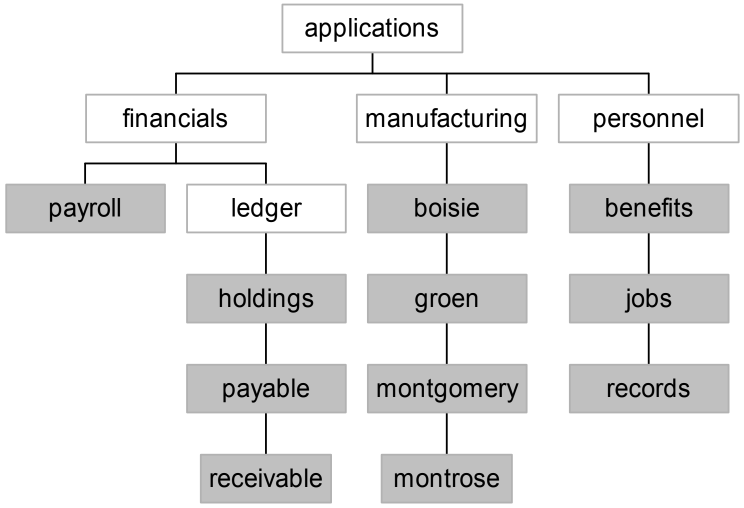
Standard UNIX Tools Hands-On Session

David Totsch
Account Support Consultant
Hewlett-Packard Company

HP WORLD 2003
Solutions and Technology Conference & Expo

CDPATH

- Syntax is just like PATH
  - Colon delimited
  - List of directories to find directories in
  - Searching current working directory is allowable (and probably desired)
- In your $HOME you will find the directory "applications"



CDPATH

- CDPATH=${HOME}:${HOME}/applications
  - cd
  - cd manufacturing # Schweet!
  - cd financials # Works as expected
  - cd ledger # What do you mean "not found"!
- CDPATH=${CDPATH}:${HOME}/applications/financials
  - Now cd(1) will find the directory 'ledger'
  - You may want to allow a local directory search
    - Do you want it in front, or at the end?

Functions In Your Shell Environment

- I usually don't recommend this, but I have been asked how to cd into a directory you just created so many times that I hereby relinquish…
- In your $HOME, create the file "mymkdir" and type in the script from the next page.
- Make the script executable and test
  - chmod +x mymkdir
  - ./mymkdir
  - You may want to try testing if a file or directory of the same name exists

mymkdir script

```
mymkdir(){
if [[ $# != 1 ]]
then
   print "usage: mymkdir new_path"
   exit 1
fi
if [[ -e $1 ]]
then
   print -n j "$1 already exists "
   if [[ -d $1 ]]
   then
       print "as a directory."
   else
       print "as some type of
   file."
   fi
   exit 1
fi
if mkdir -p $1
then
   cd $1
else
   print -n "[$?] mymkdir "
   print "of $1 failed."
   exit 1
fi

exit 0
}

mymkdir test_dir
```

mymkdir

- Once you are happy with how mymkdir behaves
  - Remove the last line (so that only the function remains)
  - mv mymkdir .env
  - Add the following environmental variable to your .profile
    - ENV=$HOME/.env
  - Logout and login again
  - Try using your new command…
- Never export ENV
  - Makes it execute for every shell
  - Your functionality could negatively impact otherwise healthy scripts
- How might the "-p" option to mkdir cause problems?

Logfile Manager

- Modular Program
- Scan list of logs
- Logs over threshold size are:
  - Reported, or
  - Truncated, or
  - Archived

Logfile Manager
The Data File

- OK, the data file ($HOME/bin/trimlogs_data) will look like this…

```
#/absolute/path/to/log          threshold   action
/home/<user>/logs/syslog        100         r
/home/<user>/logs/trial_log     50          a
/home/<user>/logs/testlog       75          t
```

Now is probably a good time to tell you that

your $HOME sports the following directories:

logs	holds example logs for you to work with
archives	where you will place your log archives
bin	script files we will use

Logfile Manager Main Script

- $HOME/bin/trimlogs.sh
- Lets design it to accept exactly one argument, datafile

```
PROG=${0##*/}          ####### The name of the exectuting shell script...
PROG=${PROG%.*}        ####### ...w/o extension

USAGE()
{
   print "USAGE: ${PROG} data_file"
}

if (( ${#} != 1 ))
then
   USAGE
   if (( ${#} > 1 ))
   then
        print "${PROG}: Only one argument accepted."
   fi
   USAGE
   exit 1
fi
```

Logfile Manager $HOME/bin/trimlogs.sh

- Now, train your script to check that the datafile exists…

```
if [[ -e ${1} ]]
then
   if [[ ! -r ${1} ]]
   then
       print "${PROG}: Data file >${1}< not readable!"
       exit 1
   fi
else
   print "${PROG}: Data file >${1}< does not exist."
   exit 1
fi
```

Logfile Management $HOME/bin/trimlogs.sh

- We have a valid data file, process it…

```
grep -v "^#" ${1} | while read LOG MLINES ACTION
do
  if [[ -f ${LOG} ]]
  then
     CLINES=$(wc -l < ${LOG})
     if (( ${CLINES} > ${MLINES} ))
     then
          case ${ACTION} in
               t)   ######## TRUNCATE
                    ;;
               r)   ######## REPORT
                    ;;
               *)   ######## UNKNOWN ACTION!
                    ;;
          esac
     fi
  else
     print "Skipping ${LOG}: File does not exist!"
  fi
done
```

Logfile Management
$HOME/bin/trimlogs.sh

- Actions
  - Truncate
    - Discard top lines until the file is below threshold
  - Report
    - Simply print a message on standard out
  - Archive
    - Copy the report to an archive directory and compress

Logfile Management ACTION: truncate

- In the case statement, add:

```
# this will bomb if insufficient
# space in /tmp to hold the log file.
ed - ${LOG} <<- =EOI=
    1,$(( ${CLINES}-${MLINES} )) d
    w
    q
    =EOI=
```

- You may want to calculate a bigger number, otherwise, once the threshold is reached, you will truncate every time the script executes…

Logfile Management ACTION: Report

- Just print a simple message on stderr.

```
print -nu2 "${LOG} has ${CLINES} lines, "
print -nu2 "which exceeds the maximum "
print -nu2 "recommended length of "
print -u2 "${MLINES} lines."
```

- You could always choose to accumulate these message in a file and send an e-mail at the end…

Logfile Management ACTION: Archive

- We will use $HOME/archives, but you might want to put them under /var/adm…

```
ARCHIVEDIR=/home/<user>/archives
DAYNUM=$(date +%d)



cp -p ${LOG} ${ARCHIVEDIR}/${LOG##*/}/${DAYNUM}
gzip ${ARCHIVEDIR}/${LOG##*/}/${DAYNUM}
> ${LOG}
```

- Later, you will want to surround cp(1) and gzip(1) with if-then-else statements for quality assurance…and to see if you have already archived!

What If I Haven't Read All Of The Messages In Syslog.log?

- Logger(1) allows any user to put a message into syslog

```
logger -t LogManager "MARK -- Logfile Viewed"
```

- If we put such a mark in the file just before we view it each time, we can use them as "from" and "to" expressions in a sed(1) [timestamps make them unique]

```
MSGS=$(grep -F "MARK -- Logfile Viewed" \
   /var/adm/syslog/syslog.log | tail -2)
```

- Gives us the last two messages (did we get two?)

```
FROM=$(print ${MSGS} | head -1)
TO=$(print ${MSGS} | tail -1)
sed -n "/${FROM}/,/${TO}/p" syslog.log
```

- The messages we have not seen!

Shall We Monitor bdf(1)?

- Keep it simple; check for % full greater than 80.
- How do we handle an entry that spans two lines?
- Sounds like a job for awk(1)…

Awk Filesystem Monitor fsw.awk

```
{ #read lines with the expected number of fields
  #(and ignore header line).
     if ( NF == 6 )
         {
         cur_pct[$1]=$5; cur_mp[$1]=$6
         }
     # now, worry about wrapped lines.
     if ( NF == 1 )
         {
         cur_pct[$1]="x"; holder=$1
         }
     if ( NF == 5 )
         {
         cur_pct[holder]=$4; cur_mp[holder]=$5
         }
}
```

fsw.awk

- Could just as easily have been:

```
NF == 6 {
        cur_pct[$1]=$5; cur_mp[$1]=$6
        }
NF == 1 {
        cur_pct[$1]="x"; holder=$1
        }
NF == 5 {
        cur_pct[holder]=$4
         cur_mp[holder]=$5
        }
```

fsw.awk

- Lets check the output, before we make decisions…

```
END {
    for (FS in cur_pct)
        {
        print FS,cur_pct[FS],cur_mp[FS]
        }
    }
```

- Test and make sure your awk merely re-formats…

```
            bdf | awk -f fsw.awk
```

fsw.awk

- Make the script self-executing…
- First line should be:

```
#!/usr/bin/awk -f
```

- Make the file executable

```
chmod +x ./fsw.awk
```

- Now, you can

```
bdf | ./fsw.awk
```

- We will eliminate the bdf(1) later…

fsw.awk

- Change the END statement to only output when a filesystem is over threshold…
- Um, wait, we have a problem with testing a string by an integer value (cur_pct has a percent-sign in it)…

```
for (FS in cur_pct)
     {
     split(cur_pct[FS],scratch,"%")
     cur_pct[FS]=scratch[1]
     if (cur_pct[FS] > 5 )
          print FS,cur_pct[FS],cur_mp[FS]
     }
```

MAJOR CODING VIOLATION! Our threshold is hard-coded.

- Lets make it so that the threshold is given as an arg…

```
for (FS in cur_pct)
    {
    split(cur_pct[FS],scratch,"%")
    cur_pct[FS]=scratch[1]
    if (cur_pct[FS] > thresh )
        print FS,cur_pct[FS],cur_mp[FS]
    }
```

- Now we are set to execute with:

```
        bdf | ./fsw.awk thresh=75
```

I'm Lazy
Lets Get Rid of Typing bdf

- Awk(1) can run bdf(1), but we will end up with one big begin statement…[remove the "end {"]
- Since we will not be reading stdin, we will have to go back to hard-coding the threshold value.
- Surround the "if" statements with curly braces, and add the while at the top…

```
while ( "bdf -l" | getline > 0 )
         {
         if ( NF == 6 )
              .
              .
              .
              cur_pct[holder]=$4; cur_mp[holder]=$5
              }
         }
```

OK, I Really Don't Like Hardcoding Variables...

- Awk(1) can read an environmental variable.
- Lets call it FSWTHRESH

```
thresh=ENVIRON["FSWTHRESH"]
if ( length(ENVIRON["FSWTHRESH"]) == 0 )
     {
     printf("%s", "Required env var FSWTHRESH ")
     printf("%s\n","is unset, or not exported")
     exit 2
     }
if (thresh < 0 || thresh > 100 )
     {
     printf("FSWTHRESH=")
     printf("%s unrealistic.\n",thresh)
     exit 2
     }
```

fsw.awk

- If we can pass in the threshold value as an environmental variable, what if an environmental variable contained the name of a configuration file?
- And what if that configuration file contained the filesystem, threshold size and mount point?
- We could read the file to load a pair of arrays just like we do when we read bdf(1) output…
- We could even test for filesystems mounted in the wrong place, extra mounts and missing mounts!

Thank You

- I wish we had more than two hours together…
- Recap
  - CDPATH
  - Environmental Function (mymkdir)
  - Logfile management script
  - Syslog.log trick
  - Filesystem monitor

HP WORLD 2003
Solutions and Technology Conference & Expo

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

interex
shared knowledge • shared power

encompass
AN HP USER GROUP

hp
invent