



Practical Scripting for HP-UX System Administrators

Bill Hassell

Director of IT

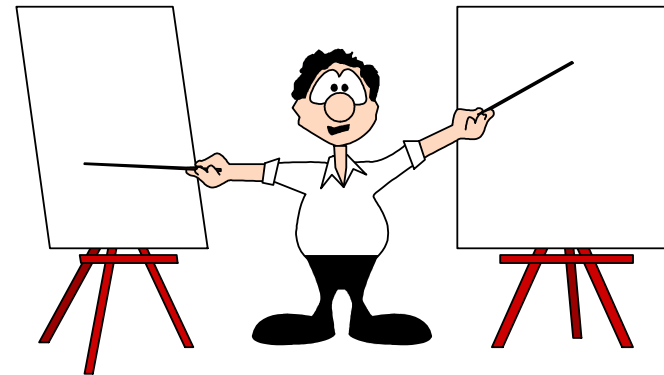
Systems and Methods, Inc.

Titles are Arial 36 point up to two lines if necessary

- Bulleted text is Arial 28 pt
 - Sub-bullets are Arial 24 pt
- Cap style is initial cap first word on titles and bulleted text
- **Emphasize keywords** as shown here
- Subdued text should be treated as such...
R141, G142, B145 (including the bullet)

Course Outline

- Shells
- Built-in tools
- Useful Commands
- Debugging
- Handling traps
- cron tips
- Script Examples



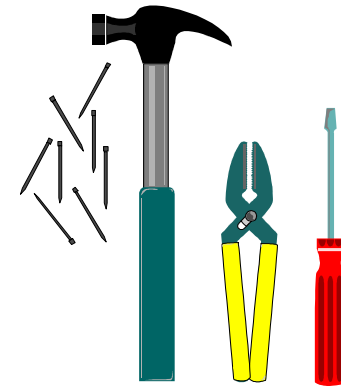
Scripting for SysAdmins

- Shells
 - Bourne
 - Korn
 - POSIX
 - C-shell
 - (bash, tcsh, ...)
- Batch files
 - Automate multiple steps
 - cron jobs



Pipes and Redirection

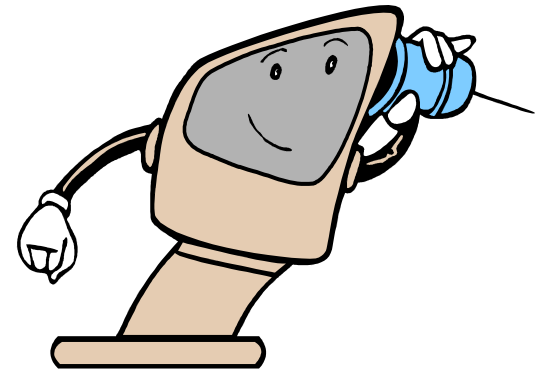
- | to feed output into another command
- > to redirect stdout
- < to redirect stdin
- >> to append
- << inline 'here' document
- <<- here-document with indents



'here' document

- inline data for commands:

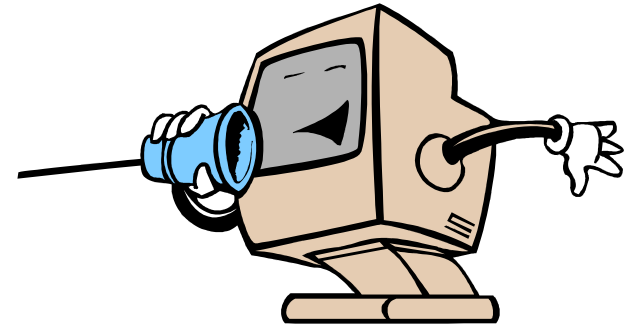
```
MYCOMPUTER=nermal.mydomain.com
ftp -n << EOF
open $MYCOMPUTER
user root rootpw
binary
get /etc/somefile
put /tmp/file /var/tmp/xfile
bye
EOF
```



'here' document (cont)

- Indented inline data:

```
MYCOMPUTER=freddie
MYFILE1=/tmp/test1
MYFILE2=/var/tmp/test2
DEST=$(pwd)
ftp -n <<- EOF
    open $MYCOMPUTER
    user root rootpw
    binary
    get /etc/$MYFILE1
    put $MYFILE2 $DEST
    bye
    EOF
echo "Done"
```

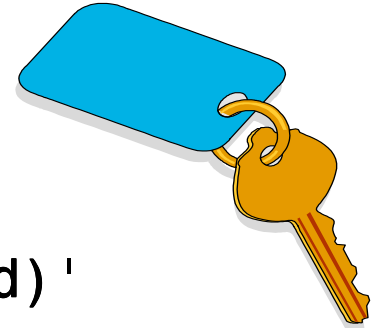


Built-in tools

- Quoting
 - single (no expansion)
 - 'single quotes: \$SPECIAL \$(pwd) '
 - double (env expanded)
 - "double quotes: \$SPECIAL \$(pwd) "
- Courtesy loader

```
#!/usr/bin/sh
#!/opt/perl/bin/perl
```
- Command results

```
$(some_command) (preferred)
`some_command` (obsolete form)
OPTLINE=$(grep opt /etc/fstab)
```

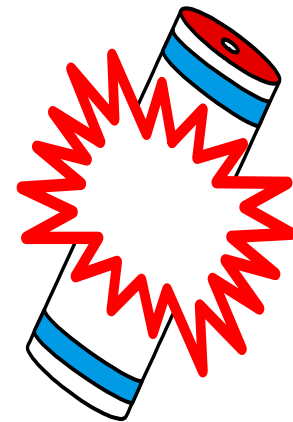


Built-in Tools (cont)

- **for - do - done**

```
for MYVAR in 1 2 46 -77889
do
    echo "$MYVAR"
done
```

```
for MYVAR in $(echo *)
do
    echo "$MYVAR"
done
```

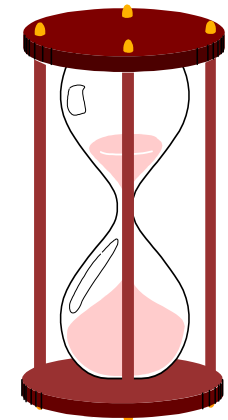


Built-in Tools (cont)

- **while - do - done**

```
COUNTER=1
while [ $COUNTER -lt 10 ]
do
    echo $COUNTER
    COUNTER=$((expr $COUNTER + 1))
done
```

```
cat some_file | while read VAR1 VAR2 VAR3
do
    echo "$VAR2 -- $VAR3 and $VAR1"
done
```



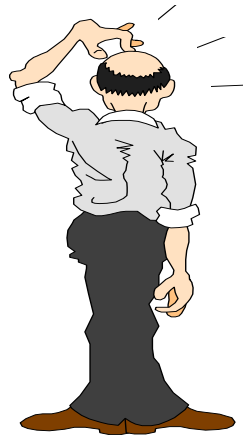
Arithmetic
computation

Built-in Tools (cont)

- **until - do - done**

```
COUNTER=1
until [ $COUNTER -gt 10 ]
do
    echo $COUNTER
    COUNTER=$(( $COUNTER + 1 ))
done
```

alternate
arithmetic



Built-in Tools (cont)

- case - in - esac

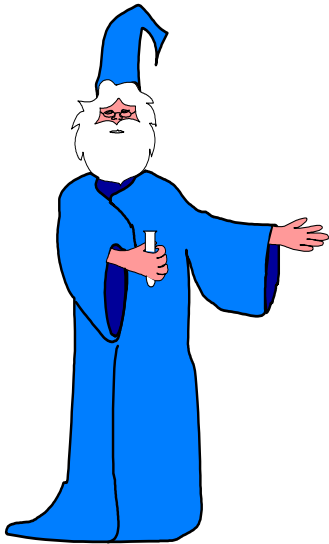
```
case MYVAR in $(cut -f 1 -d : /etc/passwd)
blh ) echo "Found blh";;
abc ) touch /tmp/x5
      rm /var/tmp/abc
      ;;
[A-Z]* ) echo "Found UPPERCASE letter"
        ;;
* ) let COUNTER=$COUNTER+1
      ;;
esac
```



alternate
arithmetic

Built-in Tools (cont)

- Functions (Like subroutines)
 - must appear prior to usage
 - can pass values
 - ideal for multiple usage
 - POSIX: `name() { body ;}`
 - KSH: `function name { body ;}`



```
function DeBugMesg
{
    if [ $DEBUG ]
    then
        echo $*
    fi
    return 0
}

DeBugMesg "line 34, var5=$VAR5"
...
DeBugMesg "routine A575, var1=$VAR1"
```

Built-in Tools (cont)

- Read



- reads 1 line at a time into REPLY or named variable(s)
- Values are space separated
- last variable gets all remaining values

```
bdf /var
```

Filesystem	kbytes	used	avail	%used	Mounted
/dev/vg00/lvol8	480341	217863	214443	50%	/var

```
bdf /var | tail -1 | read DEV KSIZE KUSED KAVAIL DUMMY  
echo "Size = $KSIZE, Left = $KAVAIL"
```

```
Size = 480341, Left = 21443
```

Built-in Tools (cont)

- test

- example: `test -d /dev`
- shorthand: `[ -d /dev ]`
- result is true or false

- Typical:

```
if [ "$MYTEXT" -eq "1234" ]
```

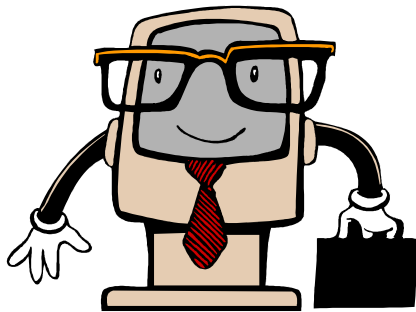
- Hint (for null parameters)

```
if [ X"$y" = X]
```

- without `X"$y"`, the test would be:

```
if [ = X ]    (syntax error)
```

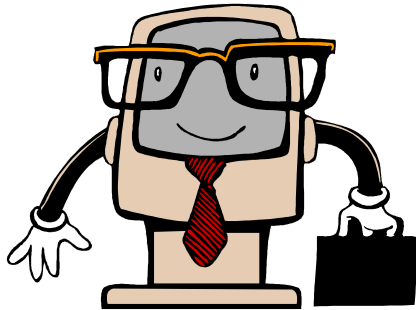
Syntax
error



Built-in Tools (cont)

- `set -u` becomes a problem for unset variables

```
set -u
if [ "$MYTEXT" -eq "1234" ]
```



- Which fails if `$MYTEXT` is unset.
Use the ksh/POSIX auto-assignment:

```
set -u
MYTEXT=${MYTEXT:-NOTset}
if [ "$MYTEXT" -eq "NOTset" ]
```

- The test string: `NOTset` becomes a flag for an unset variable

Command line

- \$1 \$2 \$3 ...

```
myfile aaa bbb $(pwd)
```

```
#!/usr/bin/sh  
echo "Param 1 = $1"  
echo "Param 2 = $2"  
echo "Param 3 = $3"
```

- \$# (quantity of params)

```
if [ "$#" -lt "2" ]  
then  
    echo "Wrong parameter number"  
    echo "Usage ..."  
    exit 1  
fi
```



Debugging

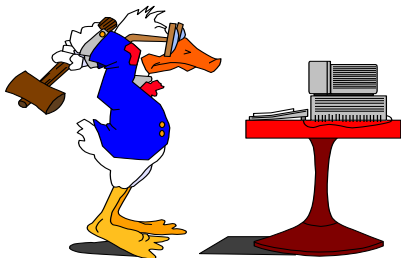
- `set -u`
 - Error if an unset variable is referenced
 - Always good SysAdmin practice

```
MYVAR=preserve
rm -rf /var/$MYVARIABLE
(actualy: rm -rf /var/)
```

- `set -x`
 - Traces execution:

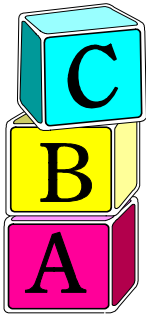
```
#!/usr/bin/sh
echo Testing
if [ $(pwd) = /opt ]
then
    echo Yes
else
    echo no
fi
```

```
# sh -x myscript
+ echo Testing
Testing
+ pwd
+ [ /root/bin = /opt ]
+ echo no
no
```



Formatting

- Indenting
 - tabs usually too wide, typically 3-4 spaces
 - Using vi, turn on autoindent (:set ai)
 - next line starts at previous line start
 - CTRL-D to backup one tab
 - :set noai to turn off



- Long lines
 - break up with \ at the end
 - stack multiple commands on separate lines like this:

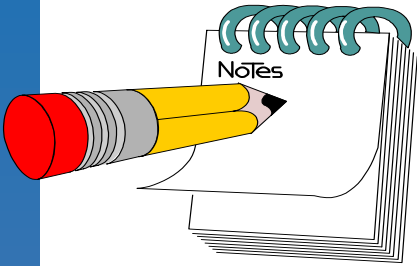
```
MHZ=$(echo itick_per_tick/D \  
      | adb -k $HPUX /dev/kmem \  
      | $TAIL -1 \  
      | awk '{print $2/10000}')
```

– ***not this:***

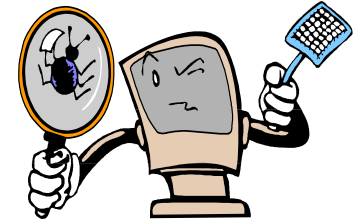
```
MHZ=$(echo itick_per_tick/D | adb -k $HPUX /dev/kmem | $TAIL -1 | awk '{print $2/10000}')
```

Text handling

- Parsing words:
 - `cut -d : -f 1,3,4 /etc/passwd`
 - `awk '{print $1 someTEXT $3}'`
- Counting lines, words, chars:
 - `wc -l /etc/passwd`
- Finding files
 - `find /home -name core`
- Finding strings
 - `grep -i abc /etc/passwd`
- Date handling
 - `date +%H:%M:%S` (see man page)
- Sorting
 - `du -x /opt | sort -rn > /var/tmp/du.opt`



Finding files



- find has many features
 - -type (for regular, device, dirs, etc)
 - -mtime (for modification time)
 - + is > as in -mtime +3 (more than 3 days)
 - - is < as in -mtime -3 (less than 3 days)
 - -perm (to find 777 or 666, etc)
 - -xdev (don't follow mountpoints)
 - -name (regexp for a filename)
 - -fstype (to limit searches, ie no cdfs)
 - -user (to find user-owned files)
 - -size (-size +9000c (otherwise blocks))
 - -exec (to run a command)

```
find /usr/local -type d -exec chmod 755 {} \;
```

Counting

- Expr

```
COUNTER=1
while [ COUNTER -lt 9 ]
do
    COUNTER=$(expr $COUNTER + 1)
done

BYTES=0
cat /var/adm/syslog/mail.log | while read
do
    BYTES=$(expr $BYTES + $(echo $REPLY) | wc -c)
done
```



\$REPLY

Counting (cont)

- ((expr))

```
COUNTER=1
while [ COUNTER -lt 9 ]
do
    COUNTER=$(( $COUNTER + 1 ))
done
```

```
BYTES=0
cat /var/adm/syslog/mail.log | while read
do
    BYTES=$(( $BYTES + $(echo $REPLY) | wc -
c))
done
```



\$REPLY

Handling traps

- trap:

```
trap "rm $TMP1 $TMP2" 1 2 3 15
```

```
trap "COUNTER=0" 16      (ie, kill -USR1  
#####)
```

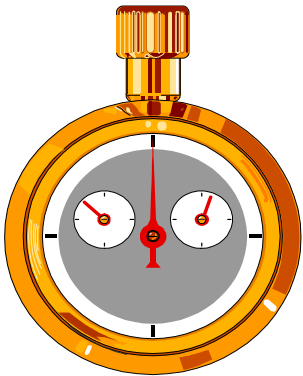
`kill -l` (to list signals)

0	SIGNULL	Null	Check access to pid
1	SIGHUP	Hangup	Terminate; can be trapped
2	SIGINT	Interrupt	Terminate; can be trapped
3	SIGQUIT	Quit	Terminate with core dump; can be trapped
9	SIGKILL	Kill	Forced termination; cannot be trapped
15	SIGTERM	Terminate	Terminate; can be trapped
16	SIGUSR1	User signal	User-defined; can be trapped
24	SIGSTOP	Stop	Pause the process; cannot be trapped
25	SIGTSTP	Terminal stop	Pause the process; can be trapped
26	SIGCONT	Continue	Run a stopped processtrap



cron Tips

- Regular execution
 - by minute, hourly, daily, weekly
 - maximum of 26 jobs each minute
 - watch time-serial scripts
- environment
 - use full pathnames or redefine \$PATH
 - assume nothing from your login env:

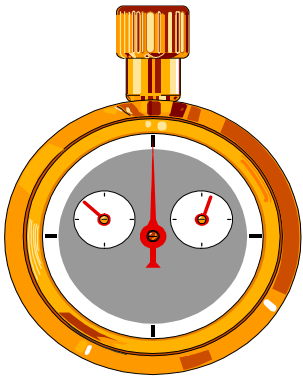


```
HOME=user's-home-directory  
LOGNAME=user's-login-id  
PATH=/usr/bin:/usr/sbin:..  
SHELL=/usr/bin/sh
```

Note `..` = PWD

Setting local variables

- Dot execution
 - *Running a subshell/program does not return or set internal values*



```
#!/usr/bin/sh
# change settings
LOCAL=testing
/var/tmp/mysettings
echo $LOCAL
```

result

testing

```
(contents of: /var/tmp/mysettings)
#!/usr/bin/sh
LOCAL="not testing"
```

Setting local variables

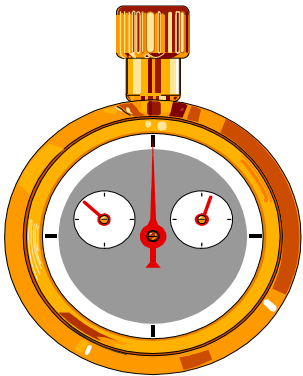
- Dot execution
 - *Use the dot command (.):*

dot

```
#!/usr/bin/sh
# change settings
LOCAL=testing
. /var/tmp/mysettings
echo $LOCAL
```

result

not testing



(contents of: /var/tmp/mysettings)

```
#!/usr/bin/sh
LOCAL="not testing"
```

Standard template

```
#!/usr/bin/sh
set -u
umask 077
export PATH=/usr/bin:/usr/sbin
MYNAME=${0##*/}
MYHOST=$(hostname)
TMPDIR=/var/tmp/$MYNAME$$
rm -rf $TMPDIR
mkdir $TMPDIR
trap "rm -rf $TMPDIR;exit" 0 1 2 3 15
. . .
ll > $TMPDIR/ll-results
exit
```

Examples: core files

- Remove core files on certain disks

```
for MYDIR in / \  
             /home \  
             /tmp \  
             /var/tmp \  
             /var/spool/news \  
             /CSCinfo \  
             /extra \  
             /tac4  
do  
    find $MYDIR -xdev -fsonly hfs -type f \  
        -name "core" -exec rm {} \  
done
```



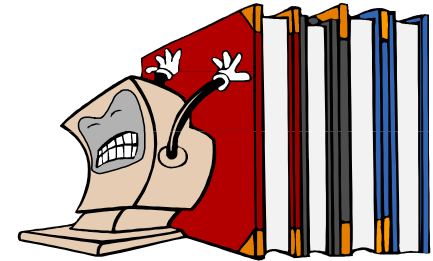
Examples:

- chkuid0

```
#!/usr/bin/sh
set -u
export PATH=/usr/bin

# Check for all UID 0's in the passwd file

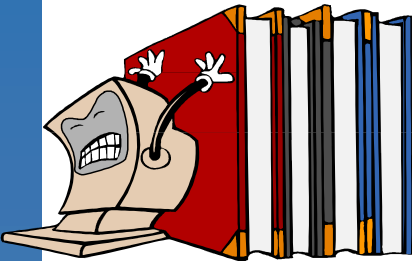
cut -f1,3 -d: /etc/passwd \
| tr ":" " " \
| while read USER UID
do
    if [ $UID = 0 ]
    then
        echo "user $USER is $UID"
    fi
done
```



Examples:

- **du**

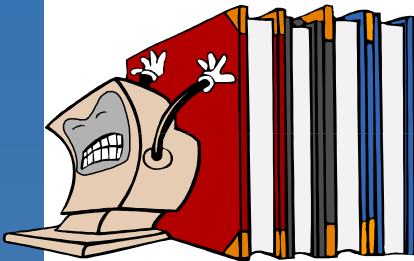
```
#!/usr/bin/sh
# Simple script to track down directory utilization
#   on a single volume and sort largest first
#
# USAGE:
#   du [ optional du options] starting-point-for-du
#
# Accepts du options, set -x automatically
#   Also set -k if the opsystem rev supports it
#
set -u
```



Examples:

- **dus – cont.**

```
TITLE="(disk usage is in"
MYREV=`uname -r | cut -f 2-3 -d .`
if [ $MYREV -lt "10.00" ]
then
    MYPATH=/bin
    TITLE="${TITLE} 512-byte blocks"
else
    MYPATH=/usr/bin
    if [ $MYREV -ge "10.20" ]
    then
        OPTIONS="-k"
        TITLE="${TITLE} K-byte blocks"
    else
        OPTIONS=""
        TITLE="${TITLE} 512-byte blocks"
    fi
fi
$MYPATH/du -x $OPTIONS ${@+"$@"} | sort -n -r | more
```

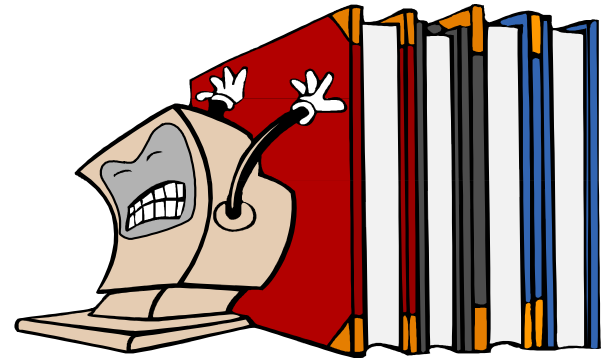


Examples:

- **lls**

```
#!/usr/bin/sh
set -u
export PATH=/usr/bin
typeset -L10 PERM
typeset -R3 LINKS
typeset -L10 UID
typeset -L8 GID
typeset -i SIZE
typeset -R6 VARSIZE
```

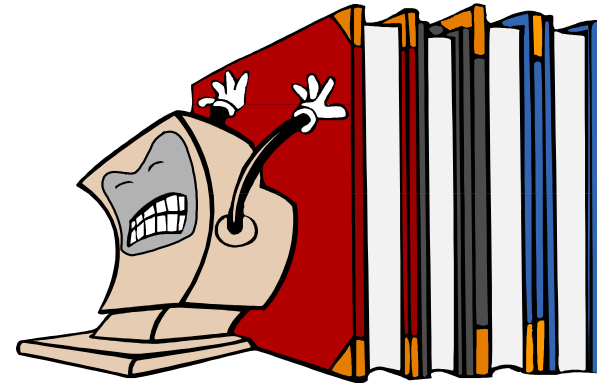
```
BINARY=/usr/bin/true
BINARY=/usr/bin/false
if $BINARY
then
    KB=1024
else
    KB=1000
fi
```



Examples:

- lls - cont

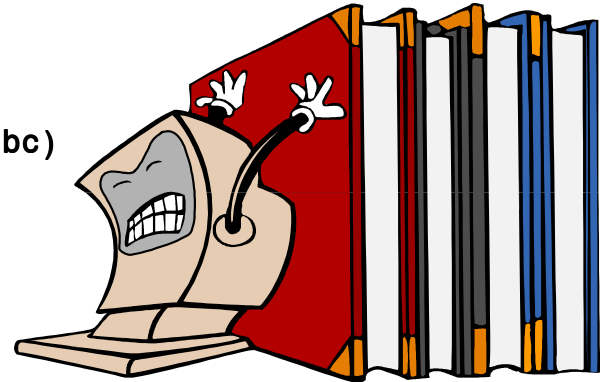
```
# Setup test and divisor values
let KB1=$KB-1
let MB=$KB*$KB
let MB1=$KB*$KB-1
let GB=$KB*$KB*$KB
let GB1=$KB*$KB*$KB-1
ls -laF ${@+"$@"} \
    | sort -rnk5 \
    | while read PERM LINKS UID GID SIZE REST
do
# find overall magnitude
    if [ "$PERM" != "total" ]
    then
        if [ $SIZE -gt $KB1 ]
        then
            if [ $SIZE -gt $MB1 ]
            then
                if [ $SIZE -gt $GB1 ]
                then
# Gigabytes
```



Examples:

- lls - cont

```
# Gigabytes
    UNITS="Gb"
    RSIZE=$(echo "scale=1 \n $SIZE/$GB" | bc)
    else
# Megabytes
    UNITS="Mb"
    RSIZE=$(echo "scale=1 \n $SIZE/$MB" | bc)
fi
    else
# Kilobytes
    UNITS="Kb"
    RSIZE=$(echo "scale=1 \n $SIZE/$KB" | bc)
fi
    VARSIZE=$(printf "%4.1f" $RSIZE)
    else
# bytes
    UNITS=" b"
    VARSIZE=$(printf "%6i" $SIZE)
fi
    echo "$PERM $LINKS $UID $GID $VARSIZE $UNITS $REST"
fi
done \
| more -e
```

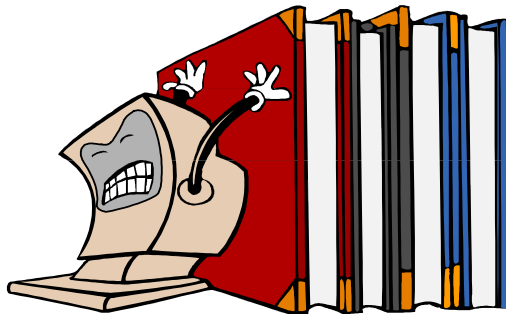


Examples:

- **lls** - *sample*

```
-rw-r-x--x   1 root      sys
-rw-r-----  1 root      sys
-rw-----   1 root      sys
-rwxr-xr-x   1 root      sys
-rw-----   1 root      sys
-rw-----   1 root      sys
-rw-----   1 root      sys
-rw-r--r--   1 bin       bin
drwxr-xr-x  20 root      root
drwxr-xr-x  10 root      sys
drwxr-xr-x   6 root      sys
```

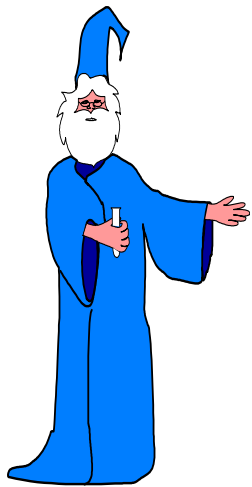
```
1.9 Mb Jan 18 2004 security_catalog*
45.2 Kb Feb  2 09:21 httpconfSample.txt
33.1 Kb Jan 28 16:13 .gpmhp-yoda
24.5 Kb May 12 14:10 pw*
12.6 Kb Jul  9 15:09 history.yoda
11.7 Kb Mar 26 13:24 .lsof_yoda
10.6 Kb Jul 17 16:39 .sh_history
 9.6 Kb Mar 11 14:29 .profile
 8.1 Kb May  3 13:32 ../
 8.1 Kb Jan 28 09:20 rcscripts/
 8.1 Kb Nov 18 2003 .sw/
```



Conclusions

- Scripting is just shell commands
- There are lots of tools
- References:
 - Start with Shells manual
 - 'The New Kornshell' by Bolsky and Korn
- Use debug tools
- Always simplify or use snippets
- cron is not a login env
- and remember:

*"Computers should work for people,
not the other way around."*



HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

