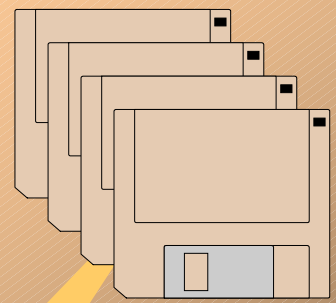
Sysadmin Script Writing

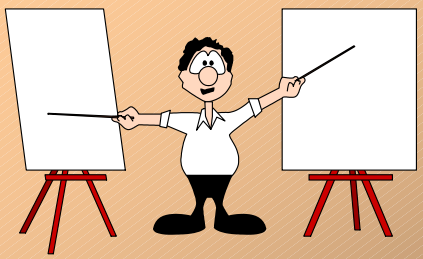
A SysAdmin's Guide to Automation



Bill Hassell

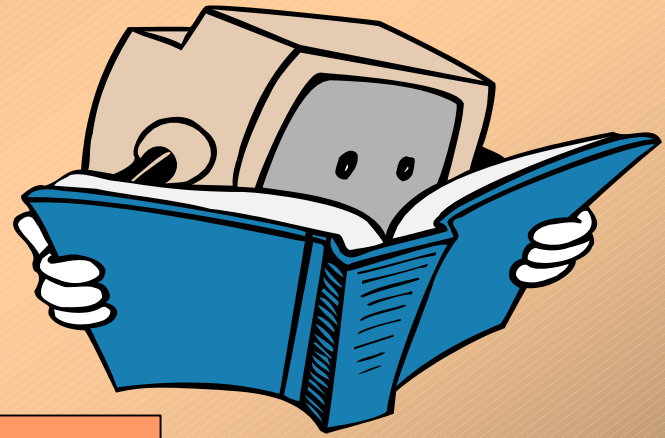
Course Outline

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



Bill Hassell
Bill Hassell Consulting, Inc

Scripting for SysAdmins



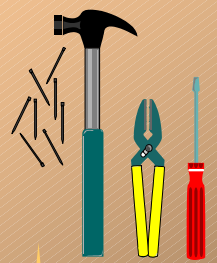
- ***Shells***
  - Bourne
  - Korn
  - POSIX
  - C-shell
  - (bash, tcsh, …)
- ***Batch files***
  - Automate multiple steps
  - cron jobs

Security!

Bill Hassell
Bill Hassell Consulting, Inc

Pipes and Redirection

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

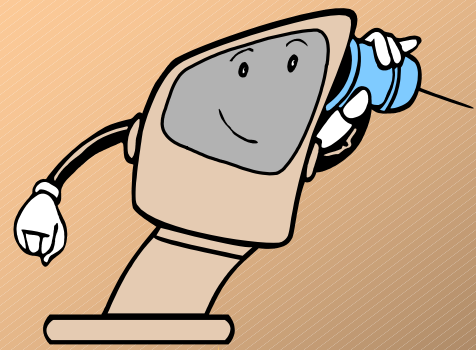


Bill Hassell
Bill Hassell Consulting, Inc

'here' document

- *inline data for commands:*

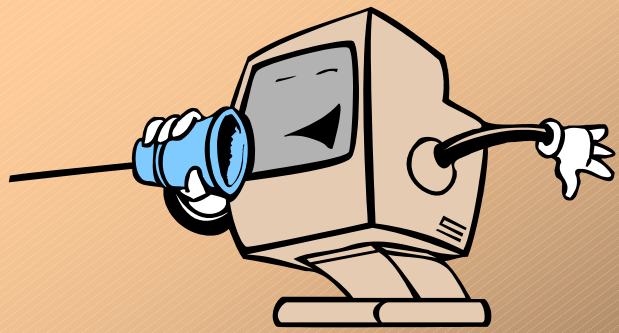
```
MYCOMPUTER=bambam.atl.hp.cm
ftp -n << EOF
open $MYCOMPUTER
user root rootpw
binary
get /etc/somefile
put /tmp/file /var/tmp/xfile
bye
EOF
```



'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
```

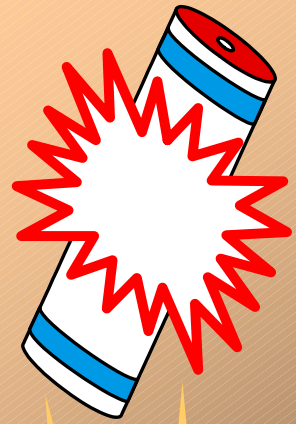
- ***Command results***

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

Built-in Tools (cont)

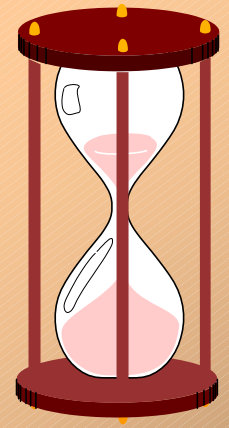
- ***for - do - done***

```
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
```

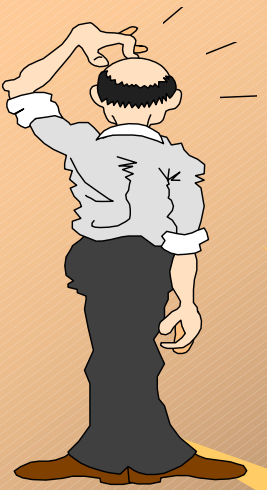
Arithmetic computation

Built-in Tools (cont)

- ***until - do - done***

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

alternate arithmetic



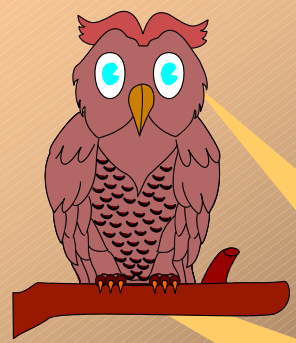
Bill Hassell
Bill Hassell Consulting, Inc

Built-in Tools (cont)

- *case - in - esac*

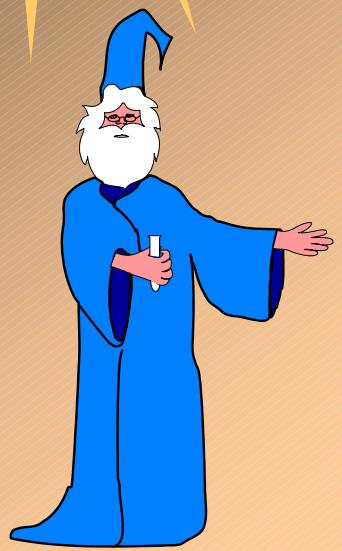
```
case MYVAR in $(cut -f 1 -d : /etc/passwd)
   blh ) echo "Found blh";;
   abc ) touch /tmp/x5
         rm /var/tmp/abc
         ;;
[A-Z]* ) echo "Found UPPERCASE letter"
         ;;
     * ) let COUNTER=$COUNTER+1
         ;;
esac
```

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
}
```

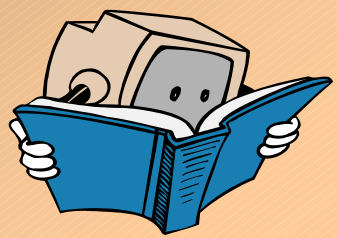
```
DeBugMesg "line 34, var5=$VAR5"
...
DeBugMesg "routine A575, var1=$VAR1"
```

Bill Hassell
Bill Hassell Consulting, Inc

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

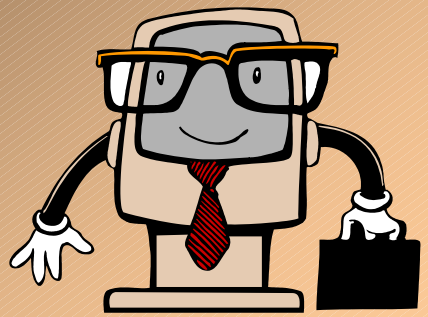
```
bdf /var
Filesystem          kbytes    used   avail %used Mounted
/dev/vg00/lvol8     480341  217863  214443   50% /var

bdf /var | tail -1 | read DEV KSIZ KUSED KAVAIL DUMMY
echo "Size = $KSIZ, 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:

Syntax error

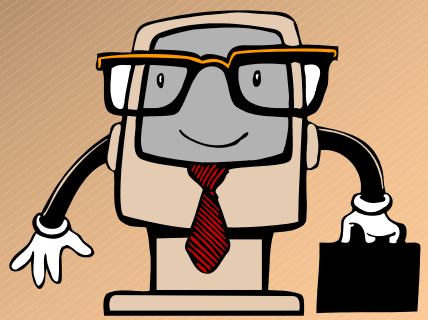
```
if [ = X ]
```

(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
```

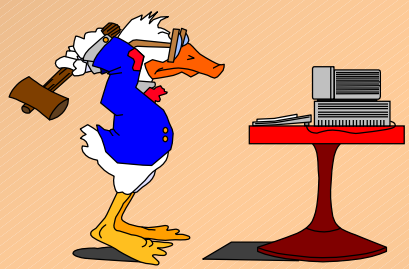
Debugging

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

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

Bill Hassell
Bill Hassell Consulting, Inc

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:

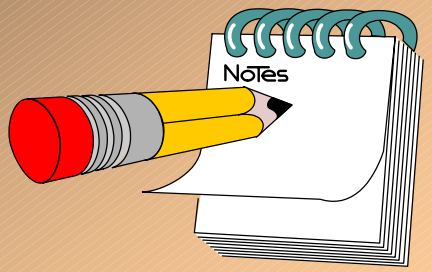
```
MHZ=$(echo itick_per_tick/D \
     | adb -k $HPUX /dev/kmem \
     | $TAIL -1 \
     | awk '{print $2/10000}')
```

  - ***not this:***

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

Text handling

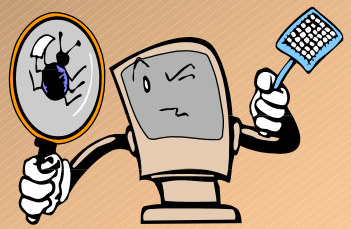
- ***Parsing words:***
  - `cut -d : -f 1,3,4 /etc/passwd`
  - `awk '{print $1 someTEXT $3}'`
- ***Counting lines, words, chars:***
  - `wc -l /etc/passwd`
- ***Finding files***
  - `find /home -name core`
- ***Finding strings***
  - `grep -i abc /etc/passwd`
- ***Date handling (Y2K)***
  - `date +%H:%M:%S (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
```

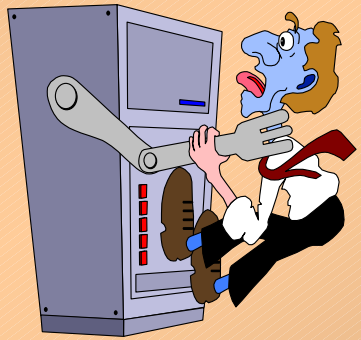
$REPLY

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
```

Handling traps



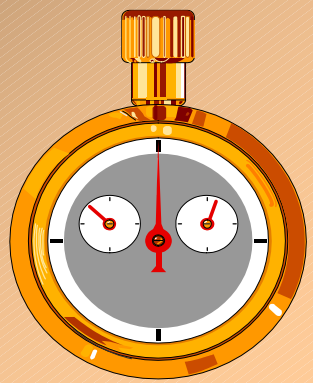
- ***trap:***

```
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

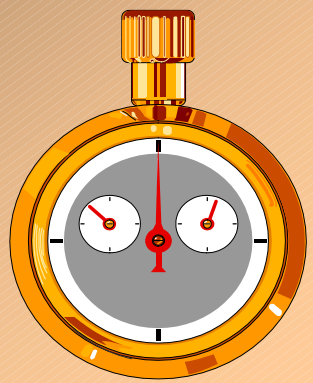
```
GREP=/usr/bin/grep
$GREP sometext SomeFile
```

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                    testing
```

result



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

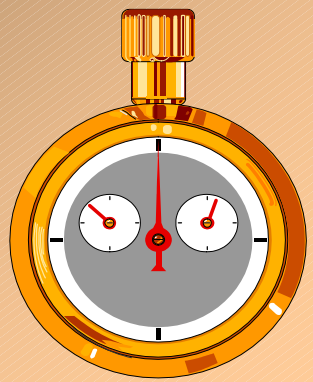
Setting local variables

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

dot

```
#!/usr/bin/sh
# change settings
LOCAL=testing
. /var/tmp/mysettings
echo $LOCAL                    not testing
```

result



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

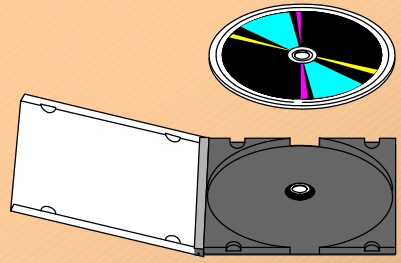
Examples: core files

- ***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: disk space

Monitor disk space

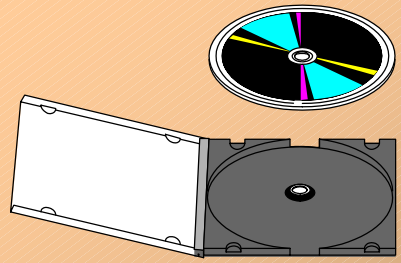


```
#!/usr/bin/sh
# Global settings
#
if [ "$#" -lt "1" ]
then
   echo
   echo "$0 Usage:"
   echo "  $0 <email-address list>, comma separated"
   echo
   echo "Example:  $0 sysadmin1@mycompany.com,sysadmin3@mycompany.com"
   exit 1
fi
if [ "$1" = "debug" ]
then
   DEBUG=/usr/bin/true
else
   DEBUG=/usr/bin/false
   NOTIFY=$1
fi
MYNAME=$(/usr/bin/hostname)
TEMPFILE=/VAR/tmp/diskspace.$$
```

Email address

Examples: disk space

Monitor disk space

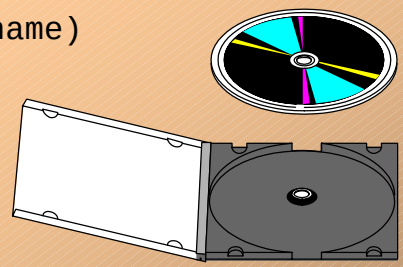


```
# Get a bdf of the filesystem(s) into a temp file
#    If JFS (vxfs) exists, use the example:
#    FSTYPES="hfs vxfs"
#
FSTYPES="hfs vxfs"
for FSYS in $(echo $FSTYPES)
do
   bdf -i -t $FSYS >> $TEMPFILE
done
trap "rm $TEMPFILE 2> /dev/null " 0
exec 7< $TEMPFILE
#
# Read through the temp file, skipping header line(s)
#
while read -u7 BDFLINE
do
#
# skip the title line(s)
#
  echo "$BDFLINE" | grep -qv ^Filesystem
```

Examples: disk space

Monitor disk space

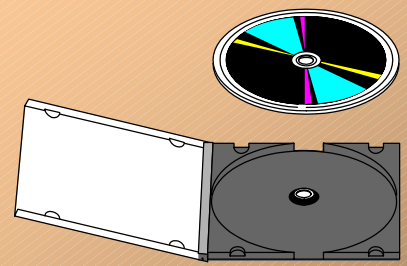
```
if [ $? -eq 0 ]
  then
#
# Check if this is a 2-line bdf (ie, long device filename)
#    Concatenate the next line if so
#
     LEN=${#BDFLINE}
     if [ $LEN -lt 60 ]
     then
        read -u7 LINE2
        BDFLINE=$(echo "$BDFLINE $LINE2")
     fi
#
# translate % to a space...large filesystems will run numbers together
# at HP-UX 9.x so we may need spaces to separate params.
#
     echo "$BDFLINE" | tr "%" "\ " \
          | read DEVFILE MAXMEGS USEDMEGS FREEMEGS \
                 PERCENT IUSED IFREE IPERCENT MNTPNT
```



Examples: disk space

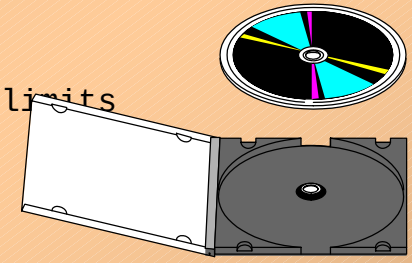
Monitor disk space

```
# Special cases for some filesystems that need different limits
#
    case $MNTPNT in
                /LaserROM  ) LIMIT=99  ;;
                /teno      ) LIMIT=100 ;;
                /usr       ) LIMIT=85  ;;
                /home/uas  ) LIMIT=90  ;;
                         * ) LIMIT=91  ;;
    esac
    if [ $PERCENT -gt $LIMIT ]
    then
        if $DEBUG
        then
           echo "$MYNAME: $MNTPNT is ${PERCENT}% full"
        else
           echo "$MYNAME: $MNTPNT is ${PERCENT}% full" \
           | mailx $NOTIFY

        fi
    fi
```



Examples: disk space

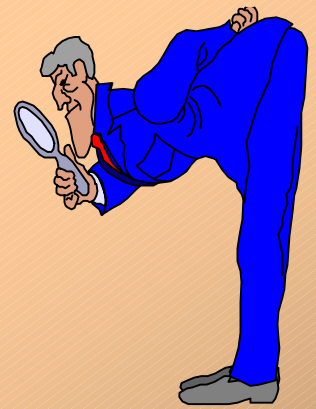
Monitor disk space



```
#
# Same test for inode usage – can specify special limits
#
    case $MNTPNT in
         /home ) ILIMIT=85 ;;
             * ) ILIMIT=90 ;;
    esac
    if [ $IPERCENT -gt $ILIMIT ]
    then
        if $DEBUG
        then
           echo "\n\n$MYNAME: $MNTPNT $IFREE inodes left)"
        else
           echo "$MYNAME: $MNTPNT $IFREE inodes left)" \
                | mailx $NOTIFY

        fi
    fi

  fi
done
rm $TEMPFILE
```

Examples: swap space

Monitor swap space



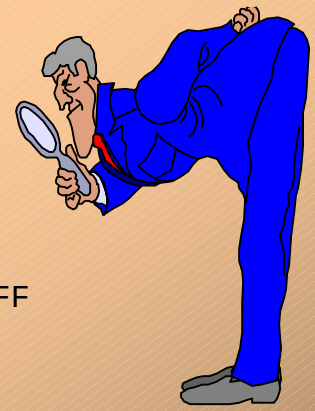
```
#!/usr/bin/sh
# Monitor swap space
#
if [ "$#" -lt "1" ]
then
   echo
   echo "$0 Usage:"
   echo "  $0 <email-address list>, comma separated"
   echo
   echo "Example:  $0 sysadmin1@mycompany.com,sysadmin3@mycompany.com"
   exit 1
fi
if [ "$1" = "debug" ]
then
   DEBUG=/usr/bin/true
else
   DEBUG=/usr/bin/false
   NOTIFY=$1
fi
MYHOST=$(/usr/bin/hostname)
MAXPERCENT=80
```

Email address

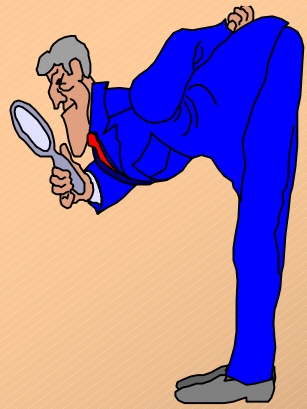
Examples: swap space

Monitor swap space

```
#
# customize swapinfo command
#
SWAPINFO="/usr/sbin/swapinfo -tm"
#
# Get the summary line
#
$SWAPINFO | tail -1 | read TOT AVAIL USED FREE FULL OTHERSTUFF
#
# then extract % full and strip the % off
#
FULLVAL=${FULL%\%*}
#
# Test for more than $MAXOERCENT% used
#
if [ "$FULLVAL" -gt "$MAXPERCENT" ]
then
   if $DEBUG
   then
      echo “$MYHOST--swap space warning: $FULL used \
         ($AVAIL Mb max, $FREE Mb left)"
```



Examples: swap space

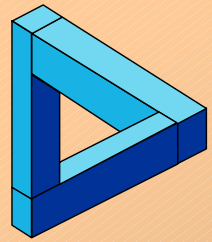


- ***Monitor swap space***

```
    else
       echo “$MYHOST--swap space warning: $FULL used \
         ($AVAIL Mb max, $FREE Mb left)"\
         | mailx $NOTIFY
    fi
fi
```

- ***Check space via cron***
  - every 10 mins? 30 mins?
- ***Notification***
  - via email (can be pager)
  - email into archive

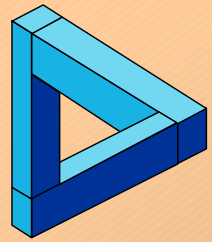
Examples: mkmount



- ***mkmount***
  - makes a mountpoint, sets ownership, permissions and a documentation file

```
#!/usr/bin/sh
#
# Make a mountpoint, take away file storage capability and
#   document that a mountpoint isn't mounted
#
# Usage:
# ------
#   mkmount <mountpoint> [ other_mountpoints ]
#
#  (Will create intermediate directories
#   as needed in a long pathname)
#
#  mkmount /mnt1/second/third:
#    /mnt1/second/third/IamNOTmounted
#
```

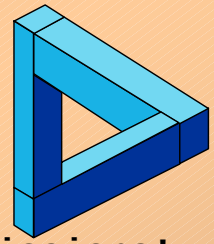
Examples: mkmount



```
# Usage check
#
MYNAME=$(basename $0)
if [ "$(whoami)" != "root" ]
then
   echo
   echo "root user is required to use $MYNAME"
   echo
   exit 1
fi
PARAMS=${#}
if [ $PARAMS -lt 1 ]
then
   echo
   echo "Usage:"
   echo "   $MYNAME <mountpoint> [ other_mountpoints ]"
   exit 1
fi
```

$#

Examples: mkmount



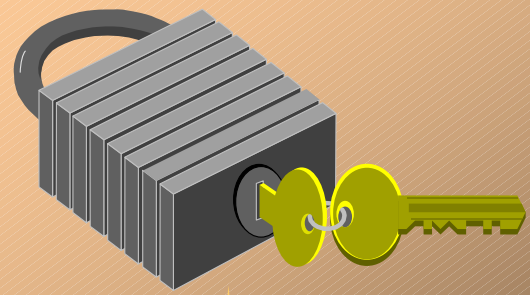
```
# Go through the list of params
do
#
# Make the directory recursively if needed - no permissions!
#
    mkdir -m 000 -p $MNTPNT
    RESULT=${?}
    if [ $RESULT = 0 ]
    then
#
# Create a dummy file to state mounpoint is not active
#    and remove write privileges to the directory
#
       touch $MNTPNT/IamNOTmounted
       chmod 444 $MNTPNT/IamNOTmounted
    else
       echo
       echo "Error:  mkdir $MNTPNT failed -- skipping"
       echo
    fi
done
```

$?

Self documenting

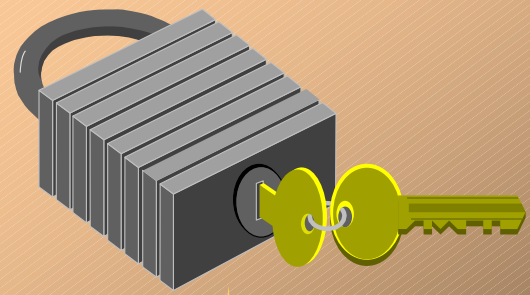
Examples: mx records

```
#!/bin/ksh
#
# Script to report MX records for specific host(s)
#
# Usage: mx hostname [ hostname ... ]
#
set -u
MYNAME=$(/usr/bin/hostname)
DNSHOST=j3107at.ssr.hp.com
TEMPFILE=/tmp/$MYNAME.$$
trap "rm $TEMPFILE 2> /dev/null " 1 2 3 15
PARMS=$#
if [ “$PARMS” -lt "1" ]
then
   echo
   echo "Usage: $(basename $0) hostname [ hostname ... ]"
   echo
   exit 1
fi
```

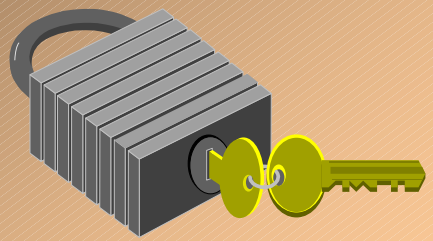


Examples: mx records

```
# Local function: use a 'here document' to run nslookup with
#   a series of commands.  This allows the function to be
#   filtered with grep when it is called.
#
LookupMX ()
{
  nslookup <<- EOF
  server $DNSHOST
  set q=mx
  $1
  EOF
}
#
# Loop thru all host(s) specified
#
while [ $PARMS -gt 0 ]
do
  echo
  echo "Mail eXchange (MX) records for $1 (from $DNSHOST)"
  LookupMX $1 | grep "mail exchanger" | sort -nk 4 > $TEMPFILE
```



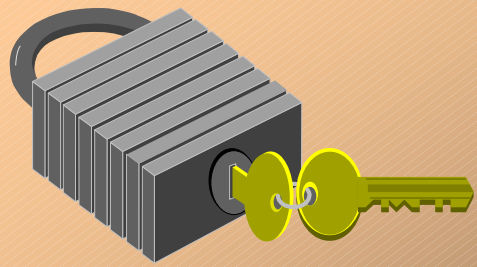
Examples: mx records



```
if [ "$(cat $TEMPFILE | wc -l )" -gt "0" ]
    then
        cat $TEMPFILE | while read MXRECORD
        do
           echo "    $MXRECORD"
        done
        grep -q "10," $TEMPFILE
        if [ $? -ne 0 ]
        then
           FQDN=$(head -1 $TEMPFILE | awk '{print $1}')
           echo
           echo " -> ERROR: No level 10 MX record found."
           echo " -> Should look like this:"
           echo "    $FQDN preference = 10, mail exchanger = $FQDN"
           echo
        fi
    fi
    shift
    PARMS=$(($PARMS - 1))
done
```

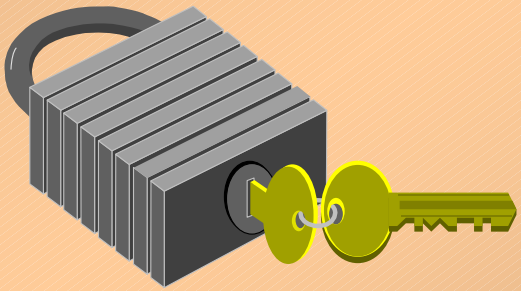
Examples: mx records

```
nslookup
Using /etc/hosts on:  rc

> server j3107at.ssr.hp.com
Specifying a server has overridden the switch policy order.
The reset command will reinstate the order specified by the switch policy.
Default Name Server:  i3107atl.ssr.hp.com
Address:  15.41.144.99

> set q=mx
> bambam.atl.hp.com
Name Server:  j3107at.ssr.hp.com
Address:  15.41.144.99

Trying DNS
bambam.atl.hp.com       preference = 100, mail exchanger = palsmtp.hp.com
bambam.atl.hp.com       preference = 100, mail exchanger = atlsmtp.hp.com
bambam.atl.hp.com       preference = 10, mail exchanger = bambam.atl.hp.com
bambam.atl.hp.com       preference = 40, mail exchanger = i3107sdm.atl.hp.com
atl.hp.com      nameserver = i3107atl.ssr.hp.com
atl.hp.com      nameserver = i3107dns.atl.hp.com
atl.hp.com      nameserver = atlrel1.hp.com
atl.hp.com      nameserver = palrel1.hp.com
palsmtp.hp.com  internet address = 156.153.255.226
```



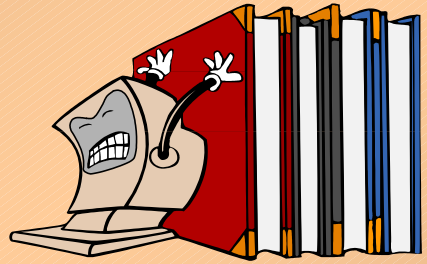
Examples: mx records

```
mx bambam.atl.hp.com
Mail eXchange (MX) records for bambam.atl (from i3107atl.ssr.hp.com)
     bambam.atl.hp.com   preference = 10, mail exchanger = bambam.atl.hp.com
     bambam.atl.hp.com   preference = 40, mail exchanger = i3107sdm.atl.hp.com
     bambam.atl.hp.com   preference = 100, mail exchanger = atlsmtp.hp.com
     bambam.atl.hp.com   preference = 100, mail exchanger = palsmtp.hp.com
```



Bill Hassell
Bill Hassell Consulting, Inc

Quickies:

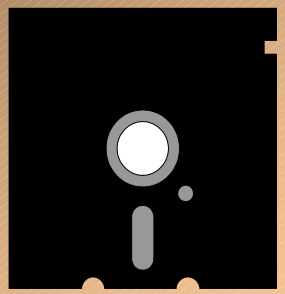


pwgrep

```
#!/bin/sh
#
# Does case-insensitive pattern matches of /etc/passwd
#
if [ "$#" -lt "1" ]
then
  echo "usage: $0 <some_string> [<more_strings>]" >&2
  echo "..strings are case-insensitive and look at /etc/passwd"
  echo
  exit 1
fi
for PARAM in $*
do
  echo
  echo "Search $PARAM:"
  grep -Fi $PARAM /etc/passwd
done
```

Quickies

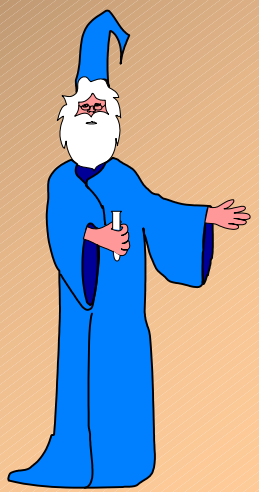
psgrep



```
#!/bin/sh
# Does a simple pattern match of ps -ef, along with a header
# With kudos to Marty Poniatowsky's book on
# HP-UX System Administration
if [ "$#" -lt "1" ]
then
  echo "usage: $0 <some_string>>" >&2
  echo "..some_string is case-insensitive for ps lines"
  echo
  exit 1
fi
AllProcs=$(ps –ef)
Header=$(echo "$AllProcs" | line)
echo "$Header"
#
# Run thru all the process lines...find matches and
# exclude this script by looking for our own name
#
echo "$AllProcs" | grep -Fi $1 | grep -Fv $0
```

Conclusions

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



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

Bill Hassell
Bill Hassell Consulting, Inc