Progress Indicator
File system monitoring

Tutor: Lưu Thanh Trà
Email: lttra@hoasen.edu.vn

Outline

- Progress indicator
- File system monitoring

Using dots

```
while true
do
   echo ".\c"
   sleep 10
done
```

Example

```
1. while true
2. do
3. echo ".\c"
4. done &
5. BG_PID=$!
6. /usr/local/bin/my_backup.ksh
7. kill $BG_PID
```

```
function dots
{
while true
do
echo ".\c"
done
}
##########################################
######### Begin of Main #############
##########################################
dots &
BG_PID=$!
/usr/local/bin/my_backup.ksh
kill $BG_PID
```

Using rotate line

```
1. function rotate
2. INTERVAL=1 # Sleep time between “twirls”
3. TCOUNT=”0” # For each TCOUNT the line twirls one increment
4. while : # Loop forever...until this function is killed
5. do
6. TCOUNT=`expr $TCOUNT + 1` # Increment the TCOUNT
7. case $TCOUNT in
8. “1”) echo ‘-’”\b\c”
9. sleep $INTERVAL
10. ;;
11. “2”) echo ‘\\’”\b\c”
12. sleep $INTERVAL
13. ;;
14. “3”) echo “|\b\c”
15. sleep $INTERVAL
16. ;;
17. “4”) echo “/\b\c”
18. sleep $INTERVAL
19. ;;
20. *) TCOUNT=”0” ;; # Reset the TCOUNT to “0”, zero.
21. esac
22. done
23. } # End of Function - rotate
```

- Example
  - rotate &
  - ROTATE_PID=$!
  - /usr/local/bin/my_time_consuming_task.ksh
  - kill -9 $ROTATE_PID
  - # Cleanup...
  - echo “\b\b “
  - # End of Example

Count down indicator

- Book at page 95

Outline

- Progress indicator
- File system monitoring

df command

- df –k
- Filesystem 1024-blocks Free %Used Iused %Iused Mounted on
- /dev/hd4 32768 16376 51% 1663 11% /
- /dev/hd2 1212416 57592 96% 36386 13% /usr
- /dev/hd9var 53248 30824 43% 540 5% /var

- df -k | tail +2 | **grep -v '/dev/cd[0-9]' | grep -v '/proc'**
- df -k | tail +2 | egrep -v '/dev/cd[0-9]|/proc' | awk '{print $1, $4, $7}'

Notification of disk usage

1. **FSMAX=”85” # Max. FS percentage value**
2. WORKFILE=”/tmp/df.work” # Holds filesystem data
3. >$WORKFILE # Initialize to empty
4. OUTFILE=”/tmp/df.outfile” # Output display file
5. >$OUTFILE # Initialize to empty
6. **THISHOST=`hostname` # Hostname of this machine**

7. ######### START OF MAIN ###############
8. # Get the data of interest by stripping out /dev/cd#,
9. # /proc rows and keeping columns 1, 4 and 7
10. **df -k | tail +2 | egrep -v ‘/dev/cd[0-9] | /proc’ \| awk ‘{print $1, $4, $7}’ > $WORKFILE**
11. # Loop through each line of the file and compare column 2

```
12. while read FSDEVICE FSVALUE FSMOUNT
    12. do
    13. FSVALUE=$(echo $FSVALUE | sed s/\%//g) # Remove the % sign
    14. if [ $FSVALUE -gt $FSMAX ]
    15. then
    16. echo “$FSDEVICE mounted on $FSMOUNT is ${FSVALUE}%” \
    17. >> $OUTFILE
    18. fi
13. done < $WORKFILE # Feed the while loop from the bottom!!
14. if [[ -s $OUTFILE ]]
15. then
    12. echo “\nFull Filesystem(s) on $THISHOST\n”
    13. cat $OUTFILE
16. print
17. fi
```

- Output
  - Full Filesystem(s) on yogi
  - /dev/hd2 mounted on /usr is 96%
  - /dev/hd10opt mounted on /opt is 97%
- Note for “sed”
  - *command* | ***sed s/current_string/new_string/g***
- Example
  - df -k | tail +2 | egrep -v ‘/dev/cd[0-9]|/proc’ \ | awk ‘{print $1, $4, $7}’ | **sed s/\%//g**

Exceptions Capability

- Problem: for some kinds of file system, the ratio can be different from the default value.
- The exception values are written in a file
- Example: /bin/usr/exceptions
  - /usr 80%
  - /var 70%

And a wrong way with grep

- while read FSDEVICE FSVALUE FSMOUNT
- do
  - # Strip out the % sign if it exists
  - FSVALUE=$(echo $FSVALUE | sed s/\%//g) # Remove the % sign
  - if [[-s $EXCEPTIONS]] # Do we have a non-empty file?
  - then # Found it!
    - # Look for the current $FSMOUNT value in the file
    - ***#WRONG CODE, DON'T MAKE THIS MISTAKE USING grep!!***
    - ***cat $EXCEPTIONS | grep -v "^#" | grep $FSMOUNT ***
    - ***| read FSNAME NEW_MAX***
  - if [$? -eq 0] # Found it!
  - then

- if [[$FSNAME = $FSMOUNT]] # Sanity check
- then
  - NEW_MAX=$(echo $NEW_MAX | sed s/\%//g)
  - **if [$FSVALUE -gt $NEW_MAX] # Use the new $NEW_MAX**
  - then
    - echo “$FSDEVICE mount on $FSMOUNT is ${FSVALUE}%” \
  - >> $OUTFILE
  - fi
- **elif [$FSVALUE -gt $FSMAX] # Not in $EXCEPTIONS file**
- then

- echo “$FSDEVICE mount on $FSMOUNT is ${FSVALUE}%” \
- >> $OUTFILE
- fi
- fi
- **else # No exceptions file...use script default**
- **if [$FSVALUE -gt $FSMAX]**
- then
- echo “$FSDEVICE mount on $FSMOUNT is ${FSVALUE}%” >> $OUTFILE
- fi
- fi
- done < $WORKFILE

Correct way

- function load_EXCEPTIONS_file
- {
- # Ignore any line that begins with a pound sign, #
- # and also remove all blank lines
- cat $EXCEPTIONS | **grep -v “^#” | sed /^$/d > DATA_EXCEPTIONS**
- }

- function check_exceptions{
- # set -x # Uncomment to debug this function
- **while read FSNAME NEW_MAX**
- do
  - **if [[$FSNAME = $FSMOUNT]] # Correct /mount_point?**
  - then # Get rid of the % sign, if it exists!
    - **NEW_MAX=$(echo $NEW_MAX | sed s/\%//g)**
    - **if [$FSVALUE -gt $NEW_MAX]**
    - then # Over Limit...Return a “0”, zero
      - **return 0 # FOUND OVER LIMIT - Return 0**
    - else # Found in the file but is within limits
      - **return 2 # Found OK**
    - fi
  - fi
- **done < $DATA_EXCEPTIONS # Feed from the bottom of the loop!!**
- **return 1 # Not found in File**
- }

main program

- **FSMAX=”85” # Max. FS percentage value**
- WORKFILE=”/tmp/df.work” # Holds filesystem data
- >$WORKFILE # Initialize to empty
- OUTFILE=”/tmp/df.outfile” # Output display file
- >$OUTFILE # Initialize to empty
- BINDIR=”/usr/local/bin” # Local bin directory
- THISHOST=`hostname` # Hostname of this machine
- EXCEPTIONS=”${BINDIR}/exceptions” # Overrides $FSMAX
- **DATA_EXCEPTIONS=”/tmp/dfdata.out” # Exceptions file w/o #, comments**

- df -k | tail +2 | egrep -v ‘/dev/cd[0-9]|/proc’ \
- | awk ‘{print $1, $4, $7}’ > $WORKFILE
- **# Loop through each line of the file and compare column 2**
- **while read FSDEVICE FSVALUE FSMOUNT**
- do # **Feeding the while loop from the BOTTOM!!**
- **# Strip out the % sign if it exists**
- **FSVALUE=$(echo $FSVALUE | sed s/\%//g) # Remove the % sign**
- if [[-s $EXCEPTIONS]] # Do we have a non-empty file?
- then # Found it!
- # Look for the current $FSMOUNT value in the file
- # using the check_exceptions function defined above.

- **check_exceptions**
- **RC=$? # Get the return code from the function**
- **if [$RC -eq 0] # Found Exceeded in Exceptions File!!**
- then
- echo “$FSDEVICE mount on $FSMOUNT is ${FSVALUE}%” \
- >> $OUTFILE
- **elif [$RC -eq 1] # Not found in exceptions, use defaults**
- then
- if [$FSVALUE -gt $FSMAX] # Use Script Default
- then
- echo “$FSDEVICE mount on $FSMOUNT is ${FSVALUE}%” \
- >> $OUTFILE
- fi
- fi
- **else # No exceptions file use the script default**
- if [$FSVALUE -gt $FSMAX] # Use Script Default
- then
- echo “$FSDEVICE mount on $FSMOUNT is ${FSVALUE}%” >> $OUTFILE

- fi
- fi
- **done < $WORKFILE # Feed the while loop from the bottom...**
- # Display output if anything is exceeded...
- if [[-s $OUTFILE]]
- then
- echo “\nFull Filesystem(s) on ${THISHOST}\n”
- cat $OUTFILE
- print
- fi

Exception with in MB

- MIN_MB_FREE=”50MB”
- ((MIN_MB_FREE = $(echo $MIN_MB_FREE | sed s/MB//g) * 1024))

Uptime

- # uptime
- 12:17pm **up 20 min, 4 users, load average: 2.29, 2.17, 1.51**

- # uptime
- 1:04pm **up 1:07, 4 users, load average: 1.74, 2.10, 2.09**

- # uptime
- 4:40pm **up 12 days, 19:03, 4 users, load average: 1.52, 0.47, 0.16**

- # uptime
- 9:16pm **up 14 days, 17 mins, 9 users, load average: 1.31, 1.82, 1.61**

- # uptime
- 9:16pm **up 14 days, 5 hr, 2 users, load average: 1.01, 1.69, 1.84**

Minutes	$9
Hours	$8
Day(s)	$10
Day(s) on the exact reboot hour anniversary	$11
Day(s) on the first 59 minutes of the reboot hour anniversary	$11

```
uptime
# Find the correct field based on how long the system has been up.
if $(uptime | grep day | grep min >/dev/null)
then
FIELD=11
elif $(uptime | grep day | grep hr >/dev/null)
then
FIELD=11
elif $(uptime | grep day >/dev/null)
then
FIELD=10
elif $(uptime | grep min >/dev/null)
then
FIELD=9
else # The machine has been up for 1 to 23 hours.
FIELD=8
fi
# Display the correct field.
echo “\nField is $FIELD \n”
```

- **LOAD=$(uptime | sed s/,//g | awk ‘{print $’$FIELD’}’)**
- **((INT_LOAD >= INT_MAXLOAD)) && echo “\nWARNING: System load has reached ${LOAD}\n”**

Uptime from the end

- function get_max {
  - (($# == 0)) && return -1
  - echo $#
- }
- MAX=$(get_max $(uptime)) # Get the total number of fields in uptime
- ((MAX == -1)) && echo “ERROR: Function Error...EXITING...” && exit 2
- TARGET_FIELD=$(((MAX - 2))) # Subtract 2 from the total
- **CPU_LOAD=$(uptime | sed s/,//g | awk ‘{print ’$TARGET_FIELD’}’)**
- echo $CPU_LOAD

iostat

- iostat -c 10 2

```
Linux 2.4.2-2 (bambam) 07/29/2002
avg-cpu:    %user  %nice    %sys  %idle
            0.69   0.00     0.48  98.83
avg-cpu:    %user %nice     %sys %idle
            62.80  0.00     37.20  0.00
```

- iostat shows:
  - The load average statistics since the last system reboot on the first line of data
  - the most current data on the last line.

vmstat

- vmstat [*delay] [count]*

```
# vmstat 30 2
 procs     memory            page            disk        faults      cpu
 r b w   swap  free  re  mf pi po fr de sr cd f0 s0 --  in   sy  cs us sy id
 0 0 0 558316 33036  57 433  2  0  0  0  0  0  0  0  0 111  500  77  2  8 90
 0 0 0 556192 29992 387 2928 0  0  0  0  0  1  0  0  0 155 2711 273 14 60 26
```

- us: Time spent running non-kernel code (user time)
- sy: Time spent running kernel code (system time)
- id : idle time
- wa: waiting time for IO
- st: time stolen from a virtual machine