Operating Systems EE5012 - Laboratory

Graham Claffey

18296661

27/Feburary/2019

SDip in Embedded Systems Engineering

Laboratory Assignment #3

Assignment objectives

1. Learn some of the basic bash shell commands
2. Learn how to write shell script programs to manage processes and files
3. Learn how to write a simple Bash shell function
4. Learn how to use “signal” and “trap” commands

Description of solution

1) file_stat.sh

My solution to the file_stat script was to create three global variables, availableBlocks, usagePercent and remainingPercent.

The avaiableBlocks variable was the df command in the home directory piped to the tail command and finally piped to the awk command. The awk command printed the 4th column, while tail took the last line in that column, then the result will be assigned to the variable.

The usagePercent was similar only with the awk printing the 5th column and using the sed command to replace the % symbol with emptiness(not even empty space). After this, the result is assigned to the variable.

The last variable was the remainingPercent which was just the number 100 to represent 100% disk usage minus the usagePercent variable, then it assigns the result to the variable.

After this comes the conditional if-else-fi statements, I created two of these just for fun. The first if-else-fi statement just checks if the availableBlocks is less than 500k, if not print the WARNING message, else print the OK MESSAGE.

In a similar fashion the next statement does the same thing, only it checks if the disk usagePercent is greater than 90(90%).

If it is, print the WARNING, else print the OK MESSAGE.

2) load_reduce.sh

For this script, I started off giving all users execute permissions using chmod for the busy_wait.sh script. Then with the next step, I ran the busy_wait.sh script (located in the same directory as the load_reduce.sh script) in the background with the & symbol appended to the end, sleep for 1 second.

In a similar fashion to the last script, I created three global variables, highestProcessPID, highestProcessUsage and highestProcessName.

These variables made use of the ps command to get the snapshot of the currently running processes and then I used awk, sort and tail to manipulate the output for these variables.

When I had the desired values, names and PIDs for these variables I used them to find the process with the highest CPU usage (which is our busy_wait.sh script).

I printed out the NAME, PID and CPU USAGE of this process, then using the kill command I sent the SIGTERM signal to this process (kill -15).

3) signal_trap.sh

This script was very similar to the second script, in that it finds the process with the highest CPU usage and prints the details to the screen. The difference here is that it loops while it waits for a signal.

Inside the while loop as before, we got the process with the highest CPU usage (busy_wait.sh is used here again) and print the PID and USAGE variables to the screen with the echo command. The sleep command is used here and the loop updates the display every second. I also added a counter here that displays how long we have been in the loop (or how long the busy_wait.sh script has been running)

I used the trap command and created a function called trap_function(), the trap command in this script was to check for the SIGINT signal (Signal interrupt, Ctrl+c) to be pressed by the user.

Once it received this signal, the trap_fucntion() function would execute what was inside, which was a final print to the output of the highest process details and using the kill command again, but this time with the kill command I sent the SIGKILL signal (Signal Kill, kill -9) to the highest process PID (busy_wait.sh).

After this, we exit the script.

Testing and results

```
endian@endian-laptop:~/University/Semester1/OperatingSystemsET4725/Assignments/Assignment3/Scripts$ ls
busy_wait.sh  file_stat.sh  load_reduce.sh  signal_trap.sh
endian@endian-laptop:~/University/Semester1/OperatingSystemsET4725/Assignments/Assignment3/Scripts$ bash
 file_stat.sh

MESSAGE: There is a suffiecnt amount of diskspace, at 95% left

MESSAGE: You are currently at 5% with 860406164 1K-blocks remaining
endian@endian-laptop:~/University/Semester1/OperatingSystemsET4725/Assignments/Assignment3/Scripts$
```

This screenshot shows the results of the file_stat.sh script. Looks like I don't have to worry about the disk usage just yet.

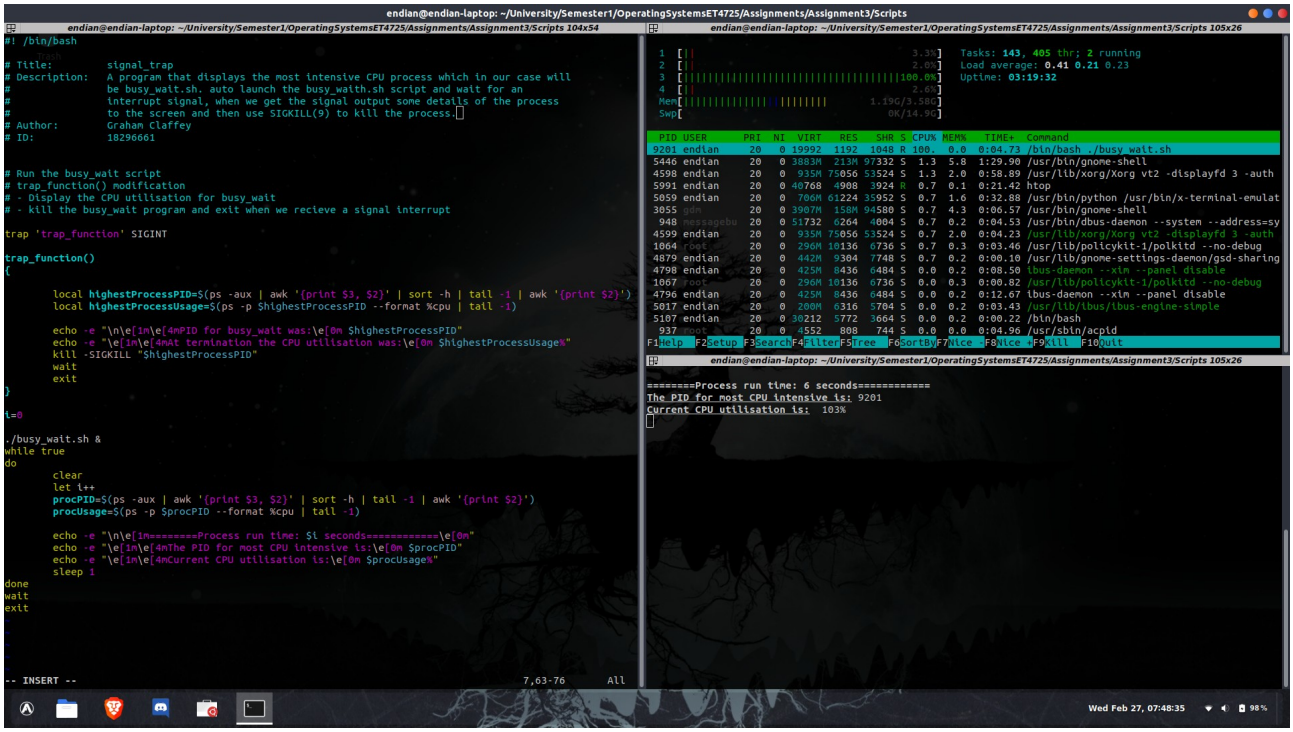
```
endian@endian-laptop:~/University/Semester1/OperatingSystemsET4725/Assignments/Assignment3/Scripts$ bash
 load_reduce.sh

Killing the processs /bin/bash
load_reduce.sh: line 33:  9875 Terminated              ./busy_wait.sh

With a PID of 9875.

It was using 103% of your CPU resources.
```

Here is the load_reduce.sh script in action, I was monitoring the execution and termination of the busy_wait.sh script using Htop and sorted by CPU_USAGE so could have a more real-time view of the processes. Everything worked successfully, although I few times I notices the CPU usage was > 100%, I suspected it to be something with the creation and termination of the busy_wait.sh script, as I didn't run into this problem with the signal_trap.sh script.



Lastly the signal_trap.sh script, with this script (and some of the previous ones) I wanted to meet the requirements that were assigned, but also to do a little bit of experimentation. I was testing out different methods with signals and interrupts, but in the end, I didn't want to take this script to far off course. I was monitoring this scripts behaviour with Htop like that last script.

This is the script after running for 22 seconds and the interrupt signal stopping the script.

```
========Process run time: 22 seconds============
The PID for most CPU intensive is: 10745
Current CPU utilisation is: 98.7%
^C
PID for busy_wait was: 10745
At termination the CPU utilisation was: 98.1%
endian@endian-laptop:~/University/Semester1/OperatingSystemsET4725/Assignments/Assignment3/Scripts$
```

Statement of completion

With this assignment, I learned a lot about the different signals that can be used with processes. Using the trap command in conjunction with functions (or just using functions in general) can make for really powerful and useful scripts. It was useful for me to experiment with the different conditionals like that if-else-fi and while loops that are presented in bash.

The overall time for completion of this assignment was about 13-14 hours, with reading, coding, documentation and experimentation.

Source code

file_stat.sh

```
#! /bin/bash


# Title:        file_stat
# Description:   This is a script used to display the current diskspace used and available blocks from the home directory.
#                When the availbable blocks go below a certian treshold a warning message is displayed.
#                I also put in a double warning for usage % at 90%.
# Author:       Graham Claffey
# ID:           18296661



# Display available blocks and percentage fo used space
# If available blocks < 5000000
# echo warning to the terminal "Disk Space is running low, n% is used"
# else echo the used that the disk space is sufficient

# creating some global variables
# AVAILABLE BLOCKS
availableBlocks=$(df ~ | tail -1 | awk '{print $4}')
# DISK USAGE
usagePercent=$(df ~ | tail -1 | awk '{print $5}' | sed 's/%//')
# REMAINING SPACE
remainingPercent=$((100-$usagePercent))
```

```
# if else block to check the if the availbable blocks is below 500k
if(($availableBlocks < 5000000));then
        echo -e "\n\e[41mWARNING:\e[0m Disk space is running low, \e[1m\e4m$usagePercent%\
e[0m is used"
else
        echo -e "\n\e[42mMESSAGE:\e[0m There is a suffiecnt amount of diskspace, at \e[1m\
e[4m$remainingPercent%\e[0m left"
fi


# if else block to check if the disk usage is above 90%
if(($usagePercent > 90));then
        echo -e "\n\e[41mWARNING:\e[0m currently only \e[1m\e[4m$reainingPercent%\e[0m of
diskspace is remaining!!"
else
        echo -e "\n\e[42mMESSAGE:\e[0m You are currently at \e[1m\e[4m$usagePercent%\e[0m
with \e[1m\e[4m$availableBlocks\e[0m 1K-blocks remaining"
fi
```

load_reduce.sh

```
#! /bin/bash

# Title:        load_reduce
# Description:  A program that is used to display and terminate the highest running process
#               in terms of CPU usage. In our example this process will be the busy_wait.sh
#               script that just increments a variable forever with no delay.
#               So the busy wait script will always end up being the highest process for us
#               due to the fact that it will hog 1 of the cores on our CPU
# Author:       Graham Claffey
# ID:           18296661

# Run the busy_wait script in the backgroud with all execution privs
# Display the PID number for the busiest process, in terms of CPU%
# Assign the PID number to a variable
# Kill the busiest process and display a message saying the naem of which process was killed
```

```
# busy_wait execution
chmod a+x busy_wait.sh
./busy_wait.sh &
sleep 1

# Display PID of busiest process
highestProcessPID=$(ps aux | awk '{print $3, $2}' | sort -h | tail -1 | awk '{print $2}')

# Display how much CPU% the busiest process uses
highestProcessUsage=$(ps aux | awk '{print $3}' | sort -h | tail -1)

# Display highest process name
highestProcessName=$(ps -aux | awk '{print $3, $11}' | sort -h | tail -1 | awk '{print $2}')

# kill and display the details of the process with the highest CPU usage
echo -e "`kill -15 $highestProcessPID` \nKilling the processs $highestProcessName"
echo -e "\nWith a PID of $highestProcessPID."
echo -e "\nIt was using $highestProcessUsage% of your CPU resources."
```

signal_trap.sh

```
#! /bin/bash

# Title: signal_trap
# Description:  A program that displays the most intensive CPU process which in our case will
#               be busy_wait.sh. auto launch the busy_waith.sh script and wait for an
#               interrupt signal, when we get the signal output some details of the process
#               to the screen and then use SIGKILL(9) to kill the process.
# Author:       Graham Claffey
# ID:           18296661
```

```
# Run the busy_wait script
# trap_function() modification
# - Display the CPU utilisation for busy_wait
# - kill the busy_wait program and exit when we recieve a signal interrupt

# trap_functoin is waiting for an interrupt signal(Ctrl+c)
trap 'trap_function' SIGINT

# trap_function to trigger when we press Ctrl+c
trap_function()
{
        # Create local variables inside this function
        local highestProcessPID=$(ps -aux | awk '{print $3, $2}' | sort -h | tail -1 | awk '{print $2}')
        local highestProcessUsage=$(ps -p $highestProcessPID --format %cpu | tail -1)

        # echo the detials of the process to the screen and KILL the process (SIGKILL) also -9
        echo -e "\n\e[1m\e[4mPID for busy_wait was:\e[0m $highestProcessPID"
        echo -e "\e[1m\e[4mAt termination the CPU utilisation was:\e[0m $highestProcessUsage%"
        kill -SIGKILL "$highestProcessPID"
        wait
        exit
}

i=0 # our counter variable used to count the seconds

./busy_wait.sh & # run the busy_wait.sh script in the backgroud (& << background signifier)
# This loop will run and print the current real time information of the highest CPU % process
# the information is printed every second, the loop will end when it recieves a SIGINT which tiggers the SIGKILL
while true
do
        clear
        let i++
        procPID=$(ps -aux | awk '{print $3, $2}' | sort -h | tail -1 | awk '{print $2}')
        procUsage=$(ps -p $procPID --format %cpu | tail -1)
```

```
        echo -e "\n\e[1m========Process run time: $i seconds==========\e[0m"
        echo -e "\e[1m\e[4mThe PID for most CPU intensive is:\e[0m $procPID"
        echo -e "\e[1m\e[4mCurrent CPU utilisation is:\e[0m $procUsage%"
        sleep 1
done
wait
exit
```