## Process Management:

1. Create a script named cpu_hog.sh. Inside put the following code:

```
#!/bin/bash

COUNTER=0

while [ true ]; do
  echo “It’s happening! $COUNTER”
  let COUNTER=COUNTER+1
done
```

2. Use chmod to make the file cpu_hog.sh executable.

3. Run the script cpu_hog.sh. Put a three-line sample of output here:

4. Open another terminal and run the command top. Look for the process cpu_hog.sh and notice how much of the cpu it is using. Write down the process id here.

5. Exit top by typing q. It is now time to kill the process cpu_hog.sh. Type: `kill -9 pid#`
Make sure you replace pid# with the actual number of the process which you noted by looking at top in step 4.

6. Run top again and make sure that cpu_hog.sh has stopped running. If it is still there, whatever you do, don’t panic, you have just simply killed some other probably very important process that is critical to system function. Everything should be fine. And if not, reboot. Repeat step 5 until you are successful.

7. Return to the terminal where cpu_hog was started. Create a new script called waiting.sh. Fill it with the following lines:

```
#!/bin/bash

n=begin

while [ “$n” != “quit” ]; do
read n
echo $n
done
```

8. Save the file and edit its permissions to be executable. Then run the file. Notice it’s behavior. You can exit the program by typing quit and pressing enter.

9. Run the program waiting.sh again, but place an & at the end: ./waiting.sh &

10. Notice that you can continue to use the terminal. waiting.sh is running in the background. To see it’s process id type:
ps -fax | grep waiting.sh
11. Make note of the process id. To kill the backgorund process, type:
`kill -9 pid#`

Be sure to replace pid# with the actual number of the process

## Adding to your PATH environment variable permanently:

1. Navigate to your users specific home folder /home/username and create a directory named bin

2. Inside your home folder is a hidden system file called .bashrc. All system files start with a dot. This file runs everytime you log into your shell with this user. Open .bashrc inside your favorite text editor. We're going to add a line that edits the path. Inside .bashrc, type the following (make sure you are out of all if statements and loops):

```
export PATH=$PATH:/home/username/bin
```

Save the file and exit.

3. print out the PATH variable to stdout to see if your new directory has been added to the path. If not, open a new terminal and repeat this step, verifying that the directory /home/username/bin has been added to the path.

4. Inside /home/username/bin create a file called testpath

5. Change the permissions to executable. Go inside the file and add the lines:

```
#!/bin/bash
echo "path test is successful"
```

6. Save the file. Go to your root directory (or any directory besides your home folder) and type:

```
testpath
```

7 If you see the words "path test is successful" print to the screen, congratulations you have altered your path!

8. Now move your programs cpu_hog.sh and waiting.sh into the /home/username/bin folder that is now part of the PATH. You should now be able to use both of these applications regardless of your present working directory.

**Other Utilities:**

1. To see how long your computer has been running type:

uptime

2. To see how full your secondary storage devices are, such as your harddrive, type:

df -h

3. To see how much RAM your system is using type:

free

4. To get a report of the size of your home directory type:

du -sh /home/username

**The find command**

a. Find all files in your home folder that end in .csv
find /home/username -name *.csv

b. find all files in /var/log that are larger than 10 megabytes
find /var/log -size +10M

c. find all directories in /etc that start with the letter I
find /etc -type d -name “i*”

d. edit the permissions of all files in your home dir (and sub dirs) so that they are group executable
find . -type f -exec chmod g+x {} \;

**Running a Cron Job:**
Sometimes we want script to run automatically on a regular basis. This lab shows how to achieve this.

1. crontab is a program that is used to schedule scripts to run at regular intervals.

2. To edit the cronjob listings, type:
crontab -e

3. If it is your first time you will be asked what editor you would like to use. Select one. From here you will enter into the editor. If it is your first time there will be a lot of comments explaining where you are and what you must do. Delete all of them.

4. Save and exit the file

5. In your home directory, create a directory named bin if it is not already there. Create a file inside this directory named testcron.sh. Give the file executable privileges and then type the following code:

```
SHELL=/bin/bash

/bin/echo “crontest has run at $(date)” >> /home/username/bin/crontest.log
```

6. Save and quit and then reenter the crontab using the -e flag to edit the scheduled jobs listings

7. Inside crontab type:

```
* * * * * /bin/bash /home/username/bin/crontest.sh
```

8. Save and quit. Make sure that you have pressed enter after the end of your line or your cronjob may not run. Also be sure to replace username with the name of your home folder (which should be your username). This job was set to run once every minute.

9. The *’s represent the time frequency that the job should run. Each star can be replaced by a number to limit how frequently the job will run. With all stars the job will run every minute. The order of the stars goes from smallest temporal unit to largest: minute, hour, day of month, month, day of week. Setting any of the stars to a number will mean you want to run the job on that number of the unit. For instance if the first star is set to 0, the job will run once every hour at the turn of the hour. If you edit the second *, representing the hour, as well, to 17, the job will run only once each day when it turns 5:00 PM.

10. Verify that the cronjob is in the file by typing:
crontab -l

11. Check the log file your script redirects to to verify that the job is running. You should see a line in the log file for every time the job has run.

## Getting Started with tput

1. We're going to make an interactive prompt based off the waiting script. Make a new copy of the waiting.sh script and name it menu.sh

2. Inside add the following code, set to execute, and run:

```
#!/bin/bash

n=1

while [ “$n” != “quit” ]; do
clear
echo ""
echo ""
echo ""

case "$n" in
  1) echo "Hola amigos!"
;;
  2) echo "Ciao!"
;;
  3) echo "Thank you so much!"
;;
  4) echo "GET LOST!!!!!"
;;
  5) n=quit
;;
  *) echo "I'm sorry we did not understand your request"
;;
esac

echo ""
echo ""
echo ""
echo "1. Say Hello"
echo "2. Say Goodbye"
echo "3. Say Thank You"
echo "4. Say Get Lost!"
echo "5. Quit"

read n
done

clear
echo "Thanks for using the menu.sh program!"
echo ""
echo ""
echo ""
```

#END OF PROGRAM

3. Notice that we have not used tput once. Go to the next page to see the same program using tput

3. Create a new file named test-tput.sh. Add the following code, set to execute and run:

```
#!/bin/bash

n=1

while [ “$n” != “quit” ]; do

msg1="Hola amigos!"
msg2="Ciao!"
msg3="Thank you so much!"
msg4="GET LOST!!!!!"

width=$(tput cols)
height=$(tput lines)
length=${#msg3}

clear

tput cup $((height / 2)) $(((width / 2) - (length / 2)))

case "$n" in
  1)
  echo $msg1
;;
  2)
  echo $msg2
;;
  3)
  echo $msg3
;;
  4)
  echo $msg4
;;
  5)
  echo "Quiting the menu program..."
;;
  *) echo "I'm sorry we did not understand your request"
;;
esac


echo "1. Say Hello"
echo "2. Say Goodbye"
echo "3. Say Thank You"
echo "4. Say Get Lost!"
echo "5. Quit"

read n

if [ "$n" == "5" ]; then
  n=quit
fi

done

clear
echo "Thanks for using the menu.sh program!"
echo ""
echo ""
echo ""
```

## Executing Sub-scripts:

Create the following files:

eat.sh
sleep.sh
walk.sh
dance.sh

In each file type:

```
echo “eat.sh has been executed on $(date)” >> actions.log
```

Be sure to replace eat.sh with the appropriate filename.

Now create a new file named actions.sh with the following code:

```
#!/bin/bash

export TERM=xterm-256color
tput setaf 0
tput setab 66

n=6

while [ “$n” != “quit” ]; do

msg1="eat.sh has been executed"
msg2="sleep.sh has been executed"
msg3="walk.sh has been executed"
msg4="dance.sh has been executed"
msg6="Greetings cherished user, choose a script to run:"

redraw() {
 local width height length

 width=$(tput cols)
 height=$(tput lines)
 length=${#msg6}


tput clear

tput cup $((height / 2)) $(((width / 2) - (length / 2)))

case "$n" in
  1)
  ./eat.sh
  echo $msg1
;;
  2)
  ./sleep.sh
  echo $msg2
;;
  3)
  ./walk.sh
  echo $msg3
;;
  4)
  ./dance.sh
  echo $msg4
;;
  5)
  echo "Quitting the actions program..."
;;
  6)
  echo $msg6
;;
  *) echo "I'm sorry we did not understand your request"
;;
esac

echo ""
echo ""
echo "1. execute eat.sh"
echo "2. execute sleep.sh"
echo "3. execute walk.sh"
echo "4. execute dance.sh"
echo "5. Quit"
echo ""
echo ""
read n

if [ "$n" == "5" ]; then
  n=quit
fi
}

trap redraw WINCH
redraw
done

tput reset
tput clear
echo "Thanks for using the actions.sh program!"
echo ""
echo ""
echo ""
```

When the program is ready and you have had a chance to use it for awhile, checkout the file actions.log. Notice that there is a logline for everytime you have run one of the subscripts from the menu system.