FYD500 - Homework II

The first number of assignments for FYD500, an introduction to linux.

Exercise 1 - Monitoring processes

Run top in one shell and create another in which you run the ps. Two processes are running according to ps, namely:

```
 PID TTY          TIME CMD

 3921 pts/2    00:00:00 zsh
 4029 pts/2    00:00:00 ps
```

This isn't very comprehensive so instead one could use ps -a to list all processes associated with the current user. This will yield:

```
PID TTY          TIME CMD
3496 pts/0    00:00:02 vim
3877 pts/1    00:00:02 top
5463 pts/3    00:00:03 atril
5478 pts/3    00:00:00 WebKitNetworkPr
8112 pts/2    00:00:00 ps
```

So atm I'm using three different psuedo-terminals or pts'. That being said, I ran the command in pts/2.

Another alternative would be to run:

ps -aef

ps -aef | awk 'NR==1, NR==2'

This will print out all the processes and the first PID is /sbin/init.

Use the find command, what does *find /* do and what happens in the top window? The command will search for all files located in the root directory. Since all files and partitions are mounted in "/". The CPU-usage will basically spike.

Start the following programs:

- xclock - in the foreground and move it to the background
- xeyes - in the background
- bring xclock back from the background.

Ok! A bit tricky. So first off I will start xclock. It will freeze the terminal like one would expect. Hit ctrl+Z, the program will now pause and move to the background. By typing bg %1 the program will resume executing but in the background. The command fg %1 will move the program back to the foreground.

Start xeyes with an ampersand preceeding the first command. Ie. xeyes &

Note that you can always use "jobs" to check which %<n> corresponds to the correct application. It is also possible to kill application by using %<n> - pretty neat

What does *kill* do and *kill -9*? The kill command kills processes. This can be done on different levels, i.e. forcably or by asking nicely.

kill -9 <PID> - Will kill everything forceably

kill -15 <PID> - Kills the process nicely, that is, if it wants/can be killed.

Write and send a message to another tty. OK! Got tricky again, *ssh -Y backefel@remote11.chalmers.se*

mesg should return "is y"

type *tty* to get the current terminal. Ok! Now were ready to rock. write <$USR> pts/<n>

```
Message from backefel@remote11.chalmers.se on pts/37 at 20:00 ...
Test
Test
```

You should see your message twice.

dmesg prints the logs that the terminal printed during the start of the kernel. It's basically a way to do it in a controlled manner instead of the text blasting by.

To time a command, one can use *time <command>*. Here's the output from doing ls in ~ and /.

```
time ls
exercise_1  exercise_2  hw1.md  hw1.pdf  hw2.md  hw2.pdf  hw3.md  hw3.pdf
ls --color=tty  0.00s user 0.00s system 75% cpu 0.003 total


time ls
bin  boot  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  s
ls --color=tty  0.00s user 0.00s system 70% cpu 0.005 total
```