Gilbert Condon

Lab3

CS470

1) Three child processes are created after the execution of the above code.
2) After opening the browser I see that it consumes %3.3 of the CPU and %3.4 of the Memory

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1770 gilbert   20   0 6155060 530864 143736 S  35.0   4.9  11:07.20 gnome-shell
 2877 gilbert   20   0  201592  71800  56500 S   4.3   0.7   0:14.82 Xwayland
66440 gilbert   20   0   86.3g 163360  98332 S   3.7   1.5   0:09.82 yelp
68796 gilbert   20   0   11.2g 362312 149800 S   3.3   3.4   0:14.37 firefox
66458 gilbert   20   0  102.2g 151072 117008 S   3.0   1.4   0:06.77 WebKitWebProces
66533 gilbert   20   0  563292  53636  40992 S   3.0   0.5   0:09.13 gnome-terminal-
```

```
gilbert@gilbert-VirtualBox:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:          10535        1447        3370          69        5717        8725
Swap:          1409           0        1409
```

3) 8725mb

4) gnome shell is consuming the most

5) the process that has the most memory is also gnome shell

6) apt-get is a command line tool for managing packes in Debian based linux distros such as ubuntu. Allows user to install remove update and upgrade packages. It is used to work with ubuuntus APT repo

Yum is a command line tool for managing gpackages in red hat based linux distros such as fedora. Allows user to install remove.

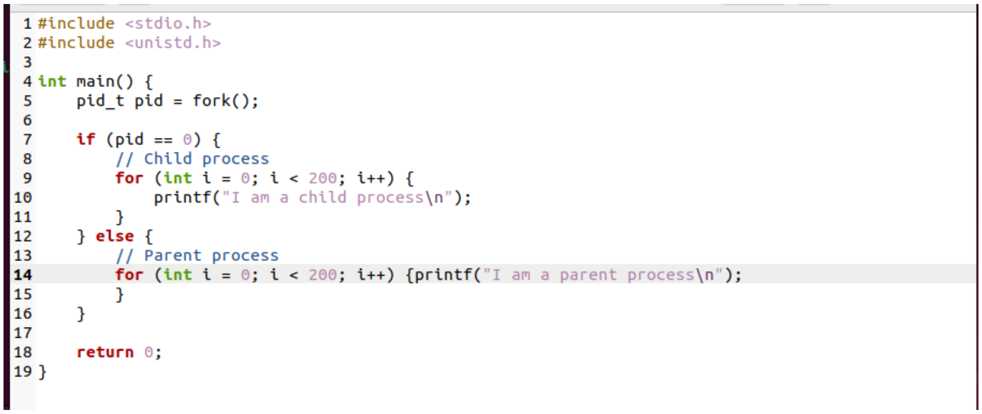
Wget is the non-interactive network downloader used to download files from the server even when the user has not logged on to the system.

Gzip is a command which is used as a compressing tool and does so by truncating the file size.

Tar is a command which stands for tape archive and is used to create archive and extract archive files. It is the most widely used archiving utility in Linux systems.

Rar is a proprietary file format for data compression and archiving. The command used to extract these files is unrar.

7)



```c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid = fork();

    if (pid == 0) {
        // Child process
        for (int i = 0; i < 200; i++) {
            printf("I am a child process\n");
        }
    } else {
        // Parent process
        for (int i = 0; i < 200; i++) {printf("I am a parent process\n");
        }
    }

    return 0;
}
```

```
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
I am a child process
I am a parent process
```

8)

practice3.c

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>

int main(){

        pid_t pid;
        char *buf;
        pid=fork();

        if (pid == 0) {
        //child process

                printf("I am the child process\n");
                }
        else{

                wait(NULL);
                printf("child process complete \n");
                printf("I am the parent process\n");
                buf = (char *)malloc(100*sizeof(char));
                getcwd(buf, 100);
                printf("\n %s \n", buf);

        }

        return 0;
}
```

```
gilbert@gilbert-VirtualBox:~$ ./practice3
I am the child process
child process complete
I am the parent process

 /home/gilbert
```

9)

practicev1.c
~/

```c
#include <stdio.h>
#include <unistd.h>
int main() {

    pid_t pid;
    pid = fork();

    if (pid == 0) {

        // Child process

        printf("I am the child process, my PID is %d, my parent's PID is %d\n", getpid(), getppid());

    } else {

        // Parent process

        printf("I am the parent process, my PID is %d, my child's PID is %d\n", getpid(), pid);

    }
    return 0;

}
```

```
gilbert@gilbert-VirtualBox:~$ ./practicev1
I am the parent process, my PID is 5730, my child's PID is 5731
I am the child process, my PID is 5731, my parent's PID is 5730
gilbert@gilbert-VirtualBox:~$
```