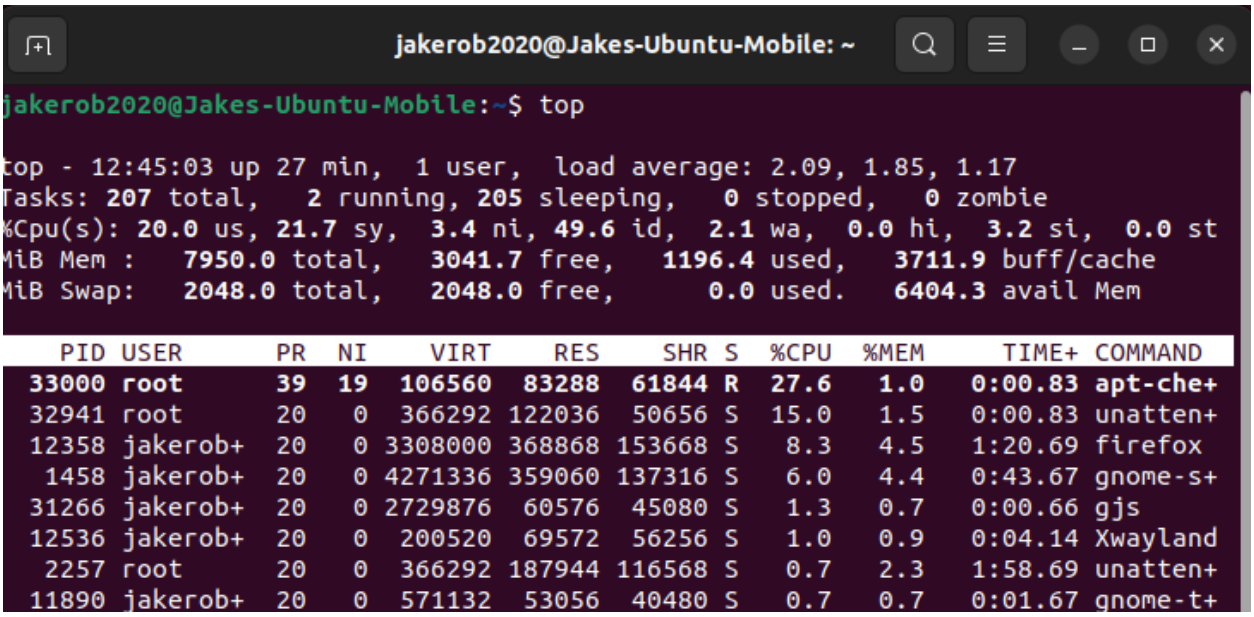
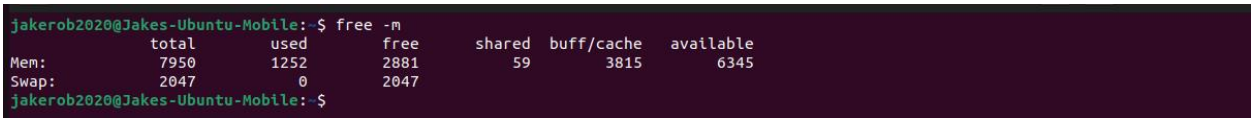
Robinson – CS 470 Lab 3

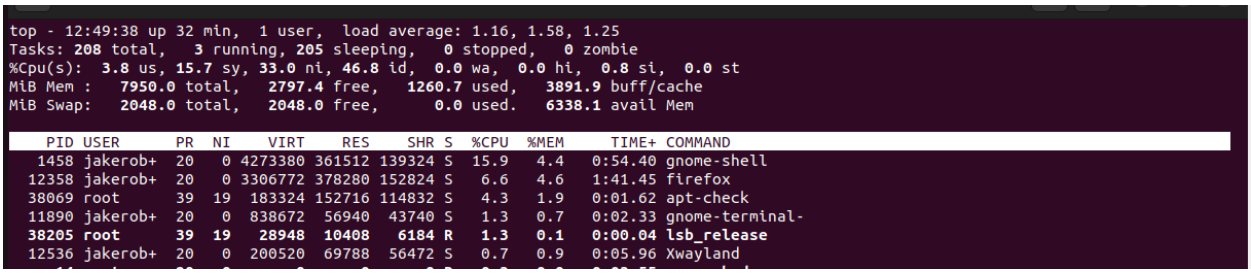
1. Three child processes are executed.

2. Res is what is consumed: 368868

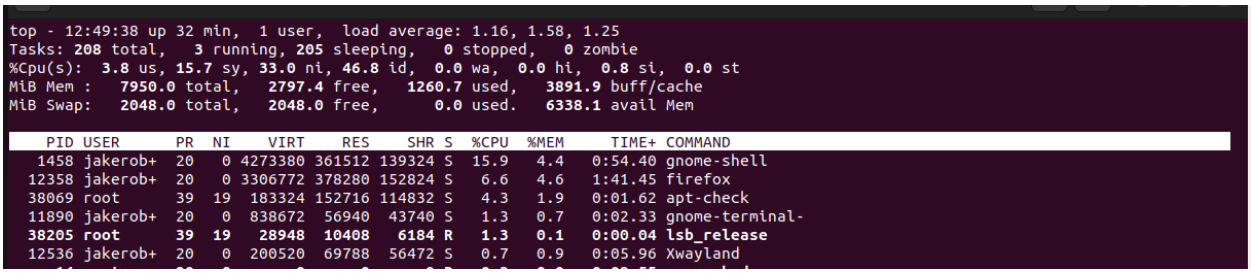


3. 

7950 Total but only 2881 available

4. 

The gnome-shell is the process using the most memory and cpu in my second run but in my first top execution apt-che+ was using the most at %CPU: 27.6

5. 

Firefox has the most memory.

6. apt-get: a command-line tool for managing packages in Debian-based Linux distributions such as Ubuntu, Debian, and Linux Mint. It allows users to install, remove, update, and upgrade packages. It is used to work with Ubuntu's Advanced Packaging Tool (APT) repository.

   yum: is a command-line tool for managing packages in Red Hat-based Linux distributions such as Fedora, Centos and RHEL. It allows users to install, remove, update, and upgrade packages. It uses the Yellowdog Updater, Modified (yum) to automatically handle dependency resolution and software management.

   wget: is a command-line utility that allows users to download files from the internet. It supports HTTP, HTTPS, and FT protocols and is commonly used to download files from the command line. It can also be used to download files in the background, or to mirror an entire website.

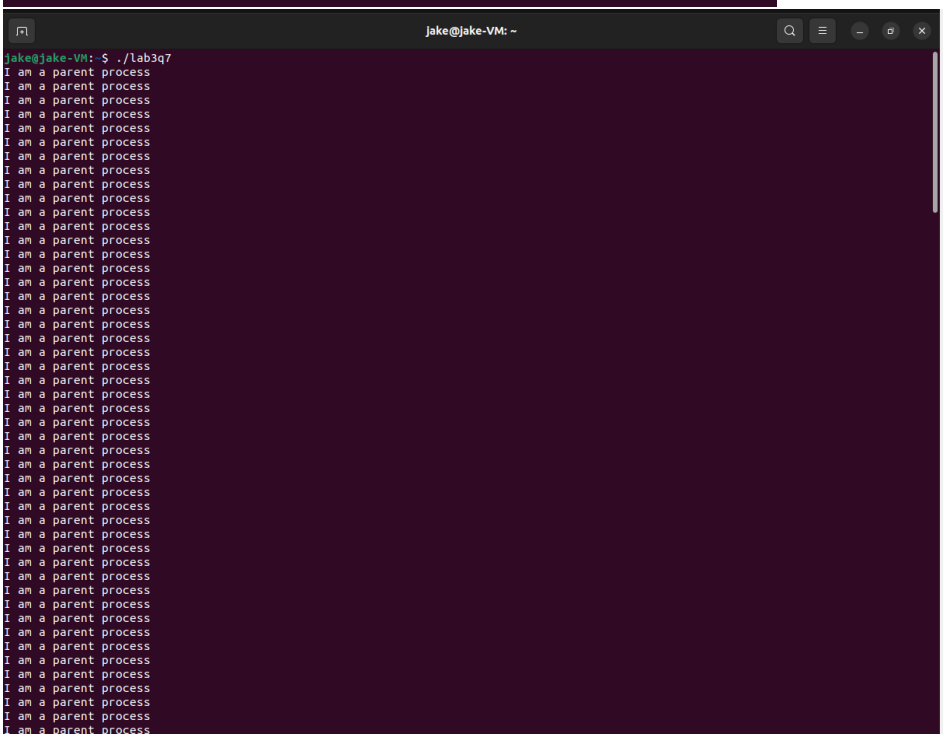
   gzip: is a command-line tool used to compress and decompress files. It uses the Lempel-Ziv-Welch (LZW) algorithm to compress files and adds a g file extension. It is commonly used to reduce the size of large files for faster transfer over the internet.

   tar: tar is a command-line tool used to create, extract, and manipulate archives. It can be used to create a single archive file from multiple files, extract files from an archive, and more. It is often used in combination with gzip to create a compressed archive file.

   rar: rar is a commercial software that is used to create, extract, and manage RAR archives. It is a proprietary file format and can only be opened using the WinRAR software. It has several features such as compression, encryption, and file recovery.

7.

```
  GNU nano 6.2                                                    lab
#include <unistd.h>
#include <stdio.h>

int main() {

    pid_t pid;

    pid = fork();

    //If pid ==0 then it is a child process, else it is a parent
    if (pid == 0) {

        for(int i = 0; i < 200; i++) {
            //Child Process
            printf("I am a child process\n");

        }

    } else {

        for(int j = 0; j < 200; j++) {
            //Parent Process
            printf("I am a parent process\n");

        }
    }

    return 0;
}
```



8.

```
  GNU nano 6.2
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdio.h>

int main() {

    pid_t pid = fork();
    char *buf;

    if (pid == 0) {

        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;

}
```

```
jake@jake-VM:~$ gcc lab3q8.c
jake@jake-VM:~$ .\a.out
.a.out: command not found
jake@jake-VM:~$ ./a.out
I am the child process
Child process Complete
I am the parent process

 /home/jake
jake@jake-VM:~$
```

9.

```
  GNU nano 6.2                                  lab3q9.c
#include <stdio.h>
#include <unistd.h>
int main() {

    pid_t 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;

}
```

```
jake@jake-VM: ~
jake@jake-VM:~$ gedit lab3q9.c
jake@jake-VM:~$ gcc lab3q9.c -o lab3q9
jake@jake-VM:~$ ./lab3q9
I am the parent process, my PID is 2797, my child's PID is 2798
I am the child process, my PID is 2798, my parent's PID is 2797
jake@jake-VM:~$
```