Name: Jimmy Battistoni

Date: 02/01/2023

Course: CS470 Operating System

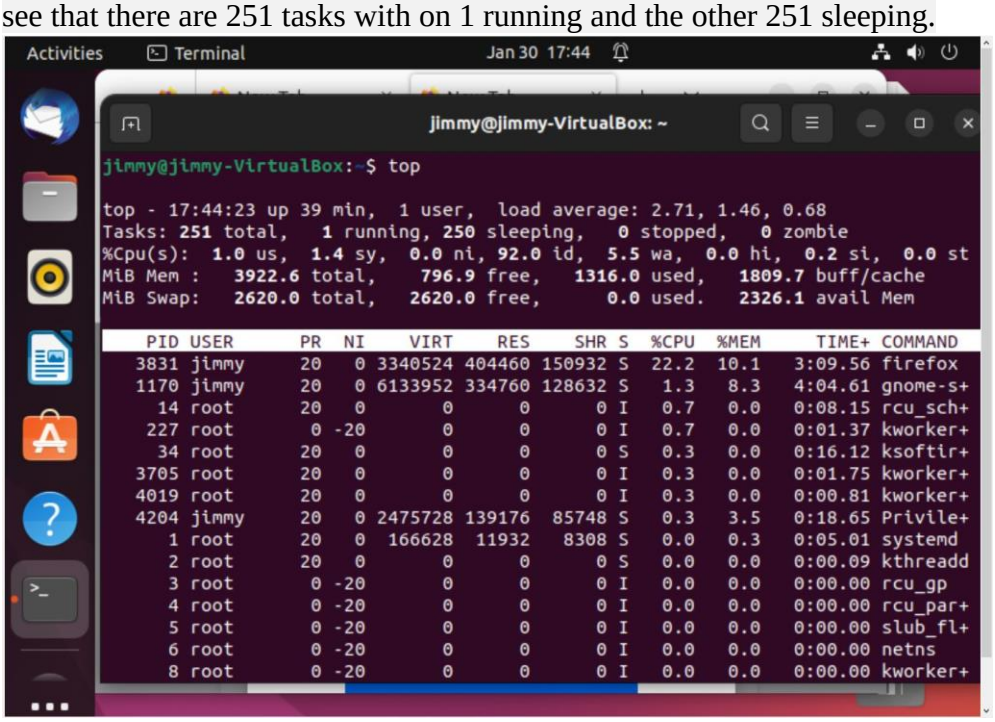
Assignment: Lab 3

1. How many child processes are created upon execution of this program?

The amount of process that created upon execution of this program is 3 child process. We know this from using 2^n – 1 and the code has two fork(); meaning that n is 2. So now we have 2^2 – 1 which equals = 3 child processes are created.

2. When you start a browser, you will notice the browser process appear in the top display. What does it consume?

When opening Firefox browser and using the top command to see what the process is consuming. Looking at the screenshot below Firefox browser consume 1.0% CPU, along with 1316 Mib used memory, followed by 1809.7 MiB in buff/cache. Looking at the Tasks line we cam see that there are 251 tasks with on 1 running and the other 251 sleeping.



3. How much memory is available in the system?

The available memory in the system is 3922.6 MiB in total. Followed by 796.9 MiB in memory that is free as seen below in the screenshot below.

```
MiB Mem :   3922.6 total,    796.9 free,   1316.0 used,   1809.7 buff/cache
```

4. Which process consumes the most CPU?

The process that consumes the most CPU is PID 1170 with the user jimmy and which uses 0.7% CPU. As seen in the screenshot below.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1170 jimmy     20   0 6139092 341548 130200 S   0.7   8.5   4:36.62 gnome-shell
  670 systemd+  20   0   14824   6148   5356 S   0.3   0.2   0:04.13 systemd-oomd
    1 root      20   0  166628  11932   8308 S   0.0   0.3   0:05.05 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.09 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

5. Which process has the most memory?

The process that has the most memory is PID 1170 with the user jimmy and uses 8.5% of memory with the virtual size of 6139092. As seen in the screenshot below.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1170 jimmy     20   0 6139092 341548 130200 S   0.7   8.5   4:36.62 gnome-shell
  670 systemd+  20   0   14824   6148   5356 S   0.3   0.2   0:04.13 systemd-oomd
    1 root      20   0  166628  11932   8308 S   0.0   0.3   0:05.05 systemd
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.09 kthreadd
    3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

6. Could you please explain the following commands?

apt-get – a Linux command line tool that is used to manage packages by installing, updating, and removing different software packages on the system. It also can be used to upgrade the system and make sure the system is up to date with the latest.

yum – a Linux command line tool that allows that users or system admin search, install, update, and remove different software packages that are on the system.

wget – a Linux command line tool that allows the system to download files from the internet, best with a URL and does not require the user to be logged in to download files.

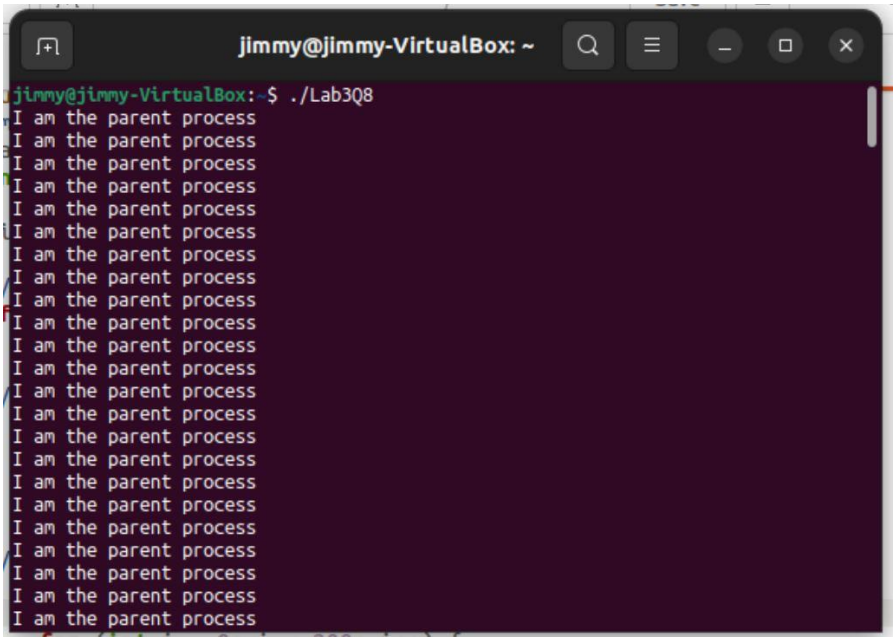
gzip – a Linux command line tool that allows for the decompressing and compressing of files. This is done by reducing the size of the files utilizing the GZIP algorithm to compress the files taking less space on the system, in return smaller files also allow for faster transfer.

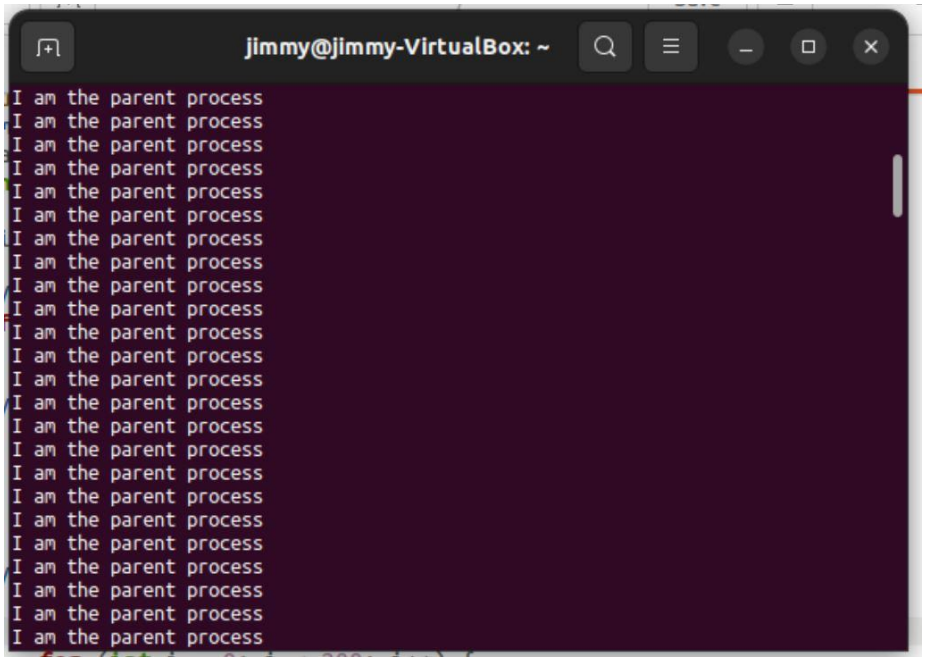
tar – a Linux command line tool that allows extraction and creation of archives that are used to compress many files to a single archive, that can have the files extracted from the single archive.

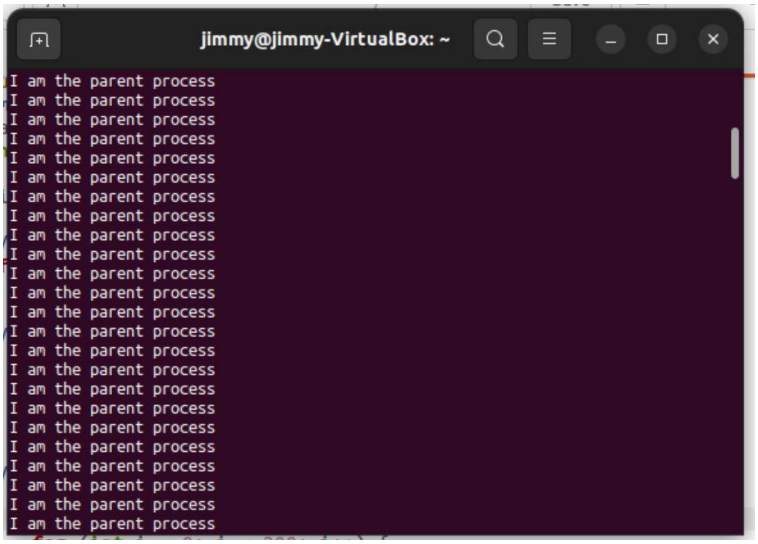
rar – is a Linux command line tool that allows for the creation of a rar file that compress the files that can than be used to extract the files from the compressed rar file.

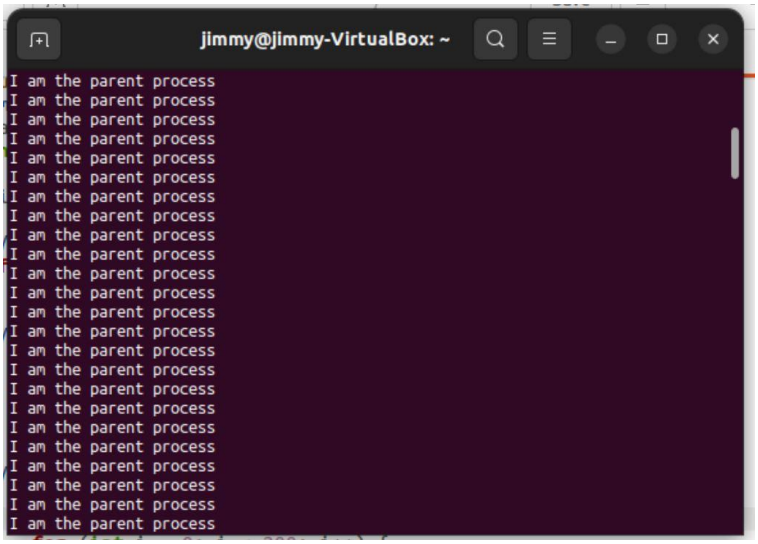
7. Write a program that will generate a child process. In a loop, the child process writes "I am a child process" 200 times and the parent process repeatedly prints "I am a parent process" in a loop.

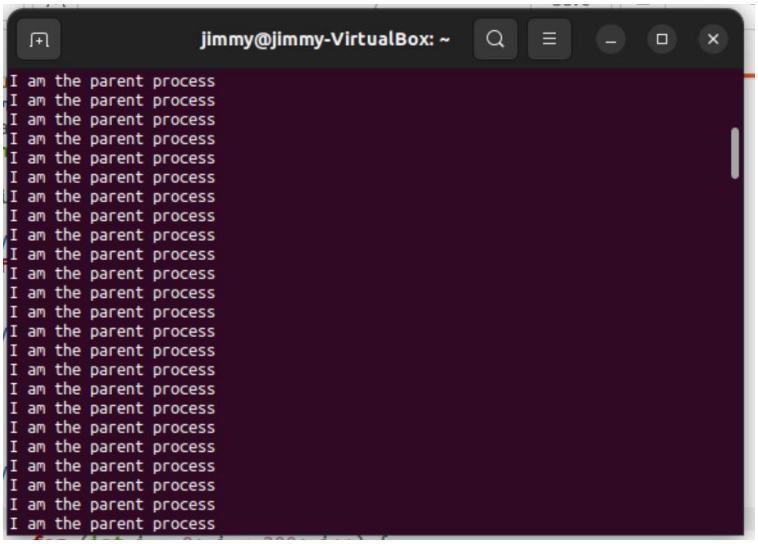
Screenshots of the program and the code are below:

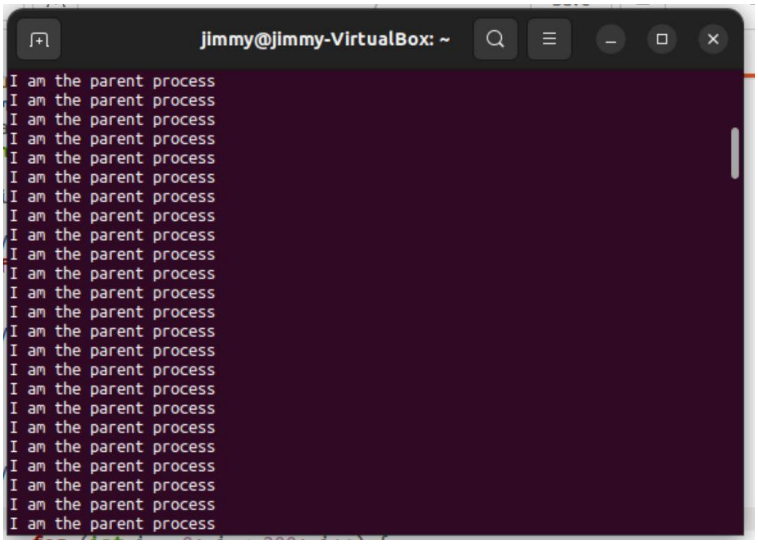


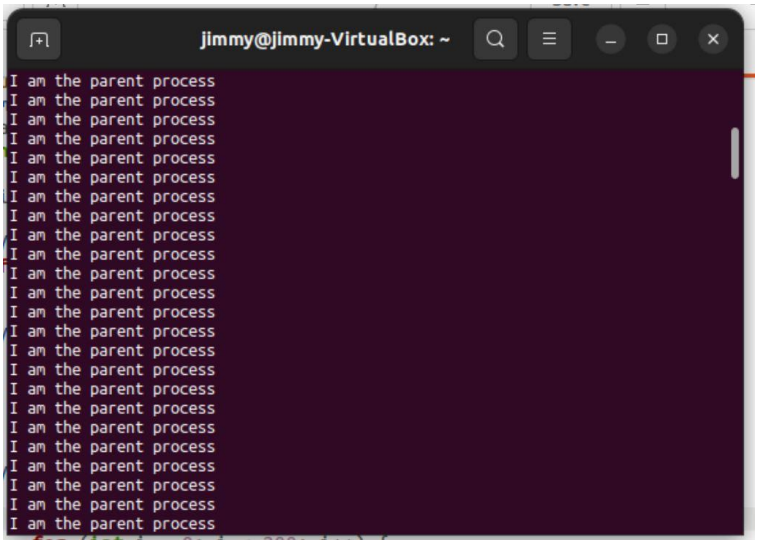


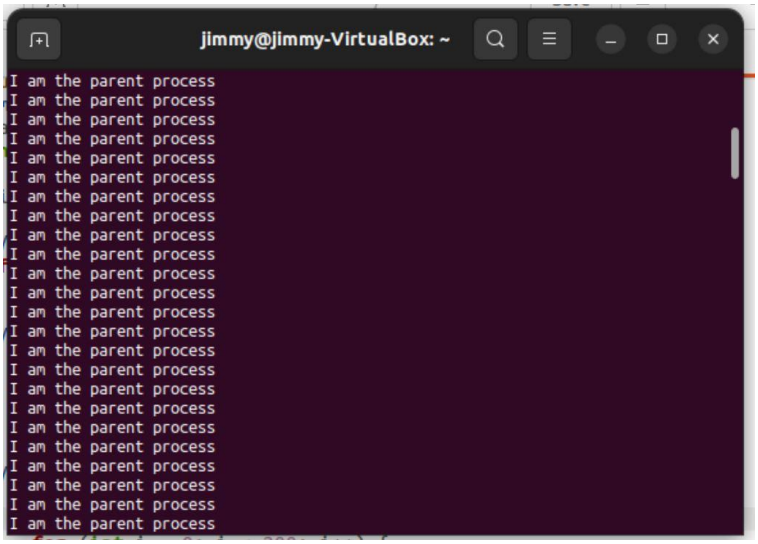

jimmy@jimmy-VirtualBox: ~
I am the parent process

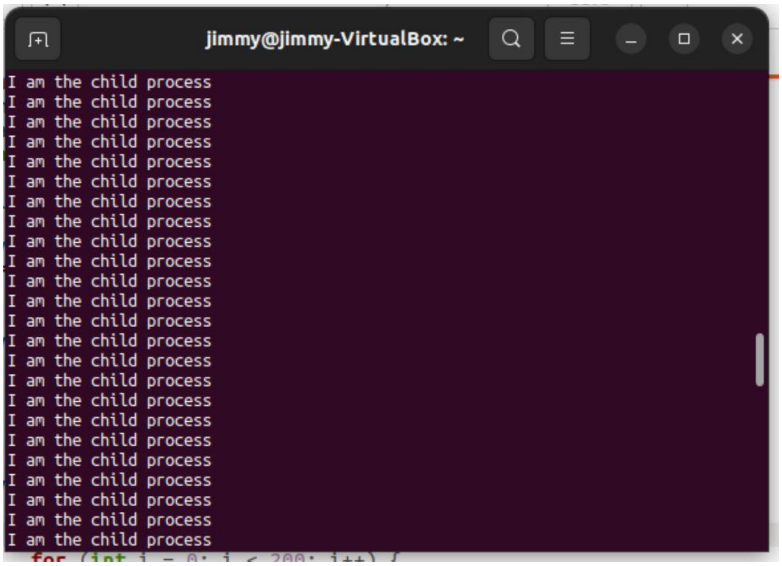

jimmy@jimmy-VirtualBox: ~
I am the parent process

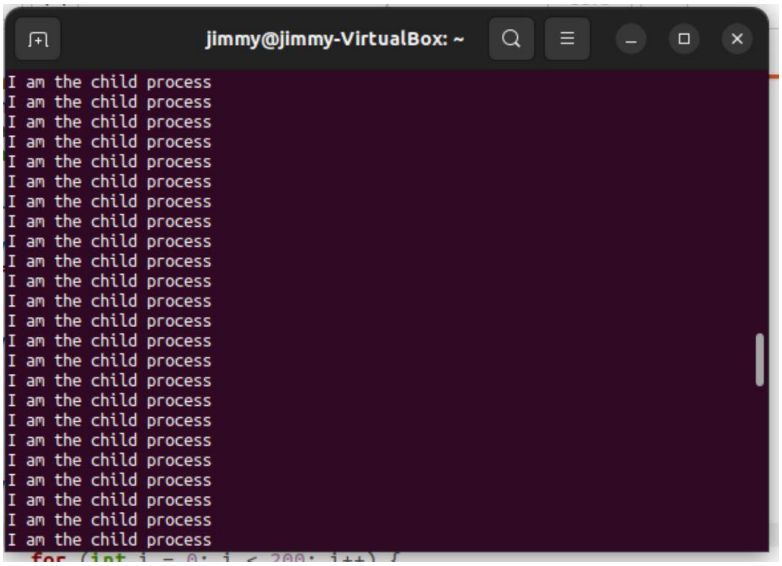

jimmy@jimmy-VirtualBox: ~
I am the parent process

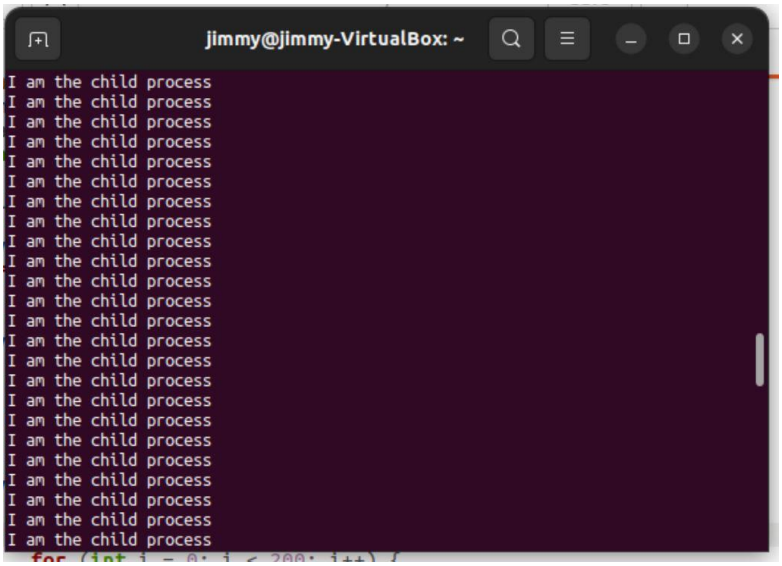

jimmy@jimmy-VirtualBox: ~
I am the parent process

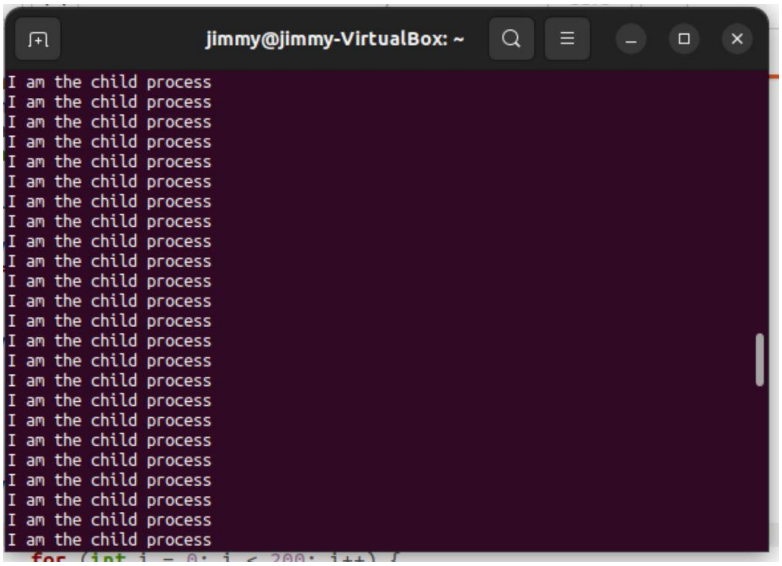

jimmy@jimmy-VirtualBox: ~
I am the parent process

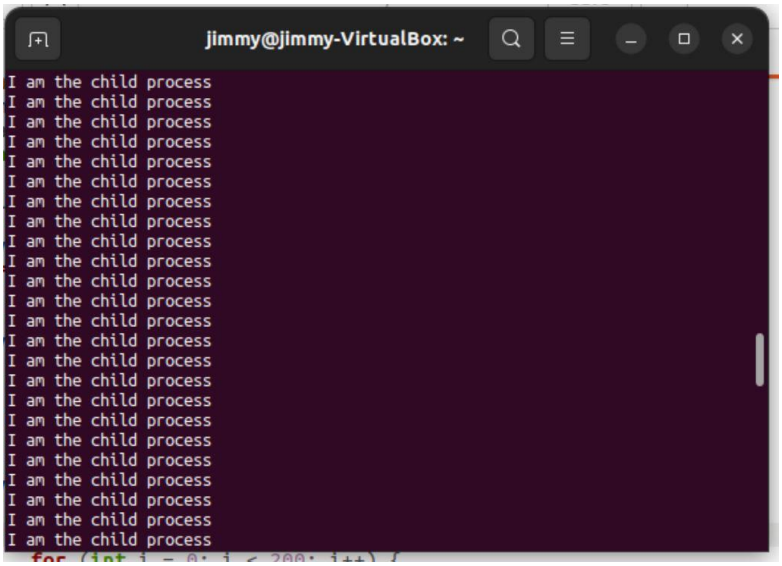

jimmy@jimmy-VirtualBox: ~
I am the parent process

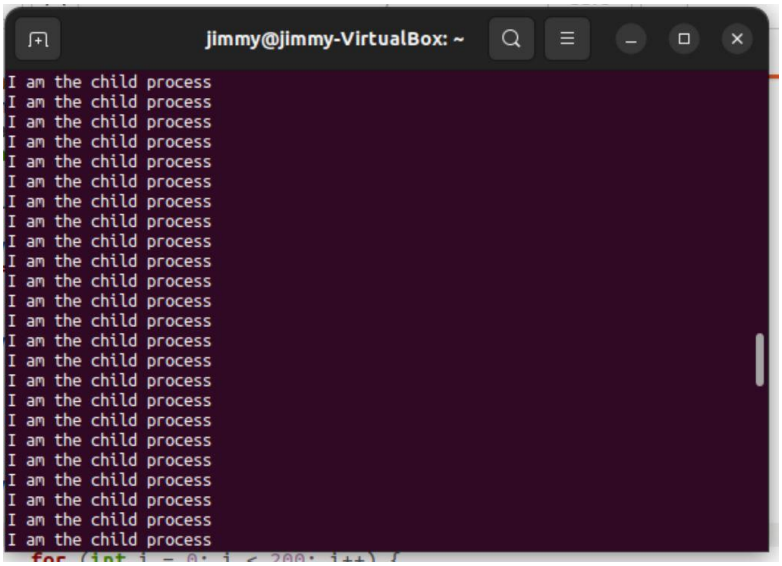

jimmy@jimmy-VirtualBox: ~
I am the child process

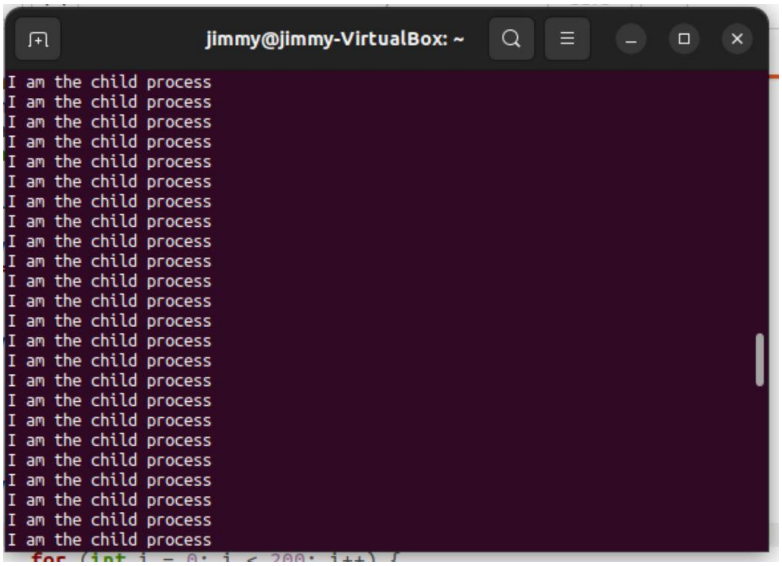

jimmy@jimmy-VirtualBox: ~
I am the child process

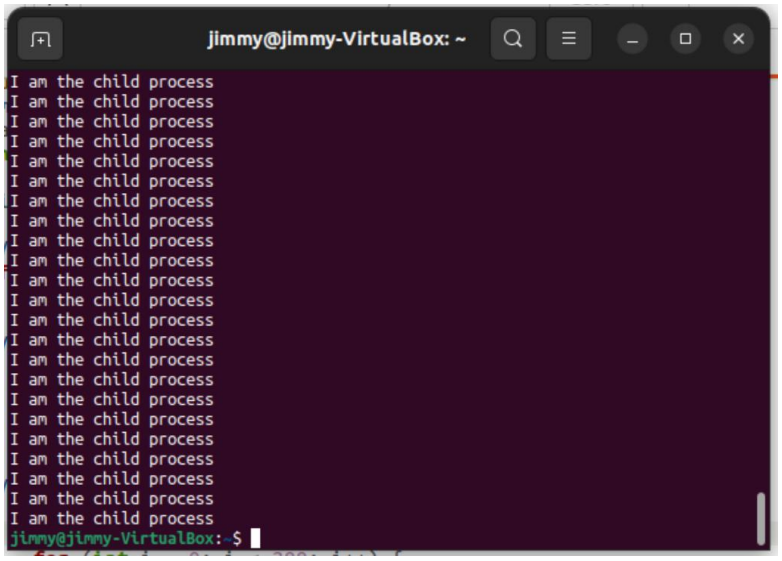

jimmy@jimmy-VirtualBox: ~
I am the child process

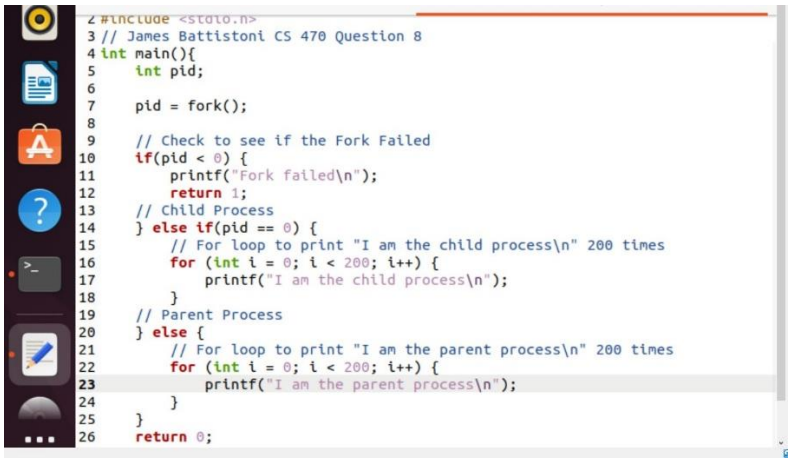

jimmy@jimmy-VirtualBox: ~
I am the child process


jimmy@jimmy-VirtualBox: ~
I am the child process


jimmy@jimmy-VirtualBox: ~
I am the child process





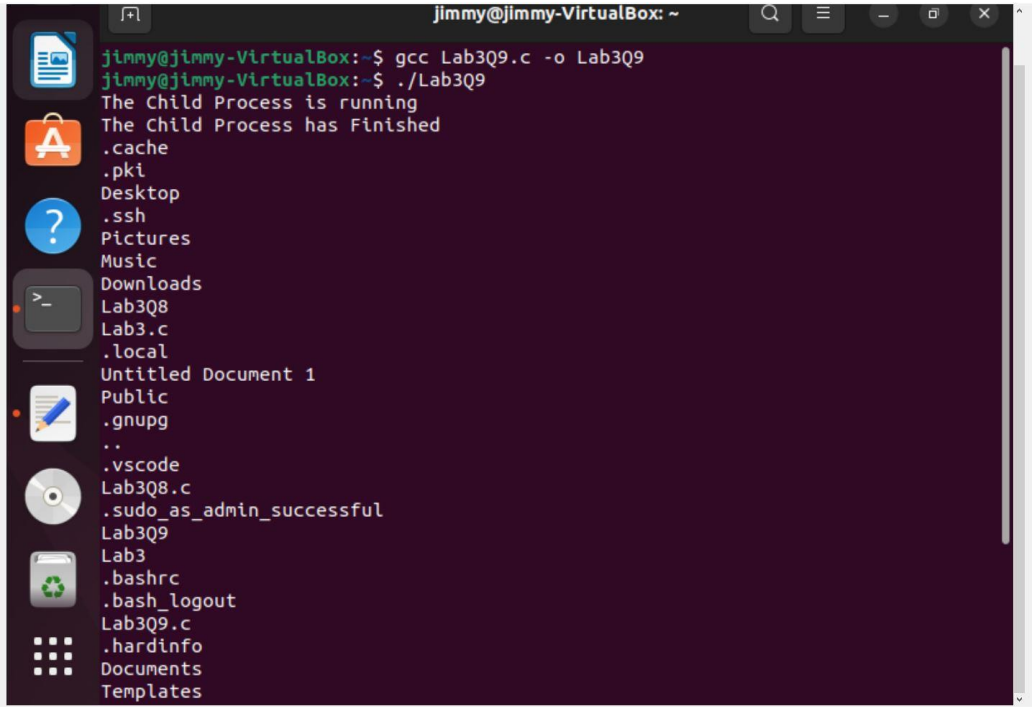


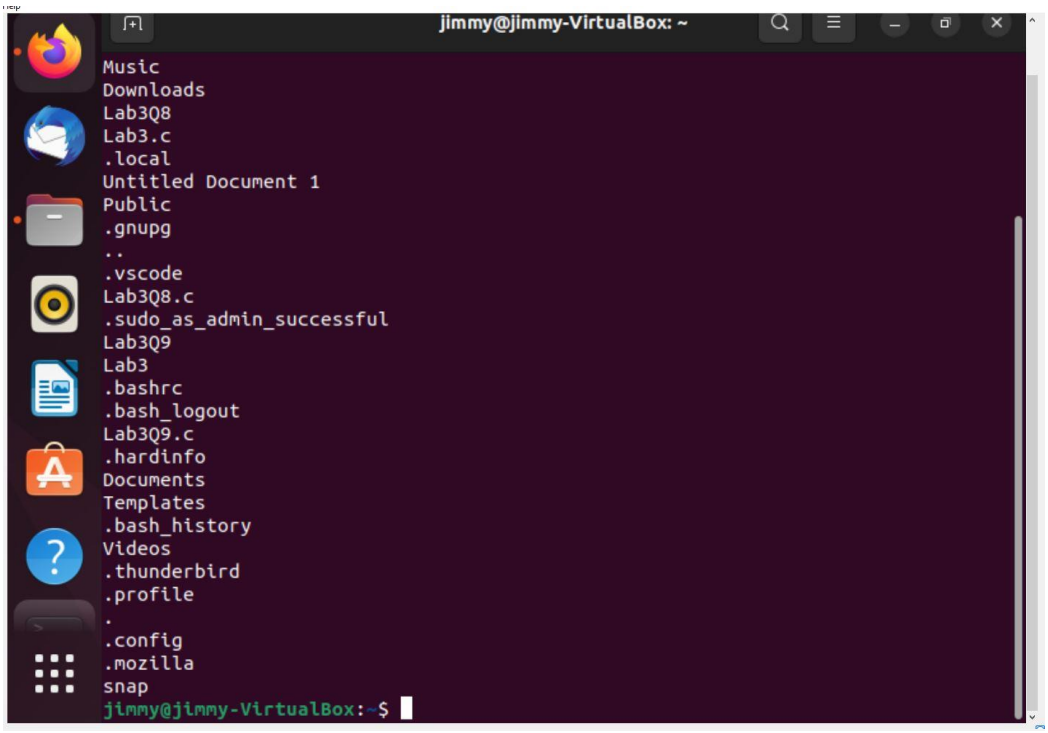
```
#include <stdio.h>
// James Battistoni CS 470 Question 8
int main(){
    int pid;

    pid = fork();

    // Check to see if the Fork Failed
    if(pid < 0) {
        printf("Fork failed\n");
        return 1;
    // Child Process
    } else if(pid == 0) {
        // For loop to print "I am the child process\n" 200 times
        for (int i = 0; i < 200; i++) {
            printf("I am the child process\n");
        }
    // Parent Process
    } else {
        // For loop to print "I am the parent process\n" 200 times
        for (int i = 0; i < 200; i++) {
            printf("I am the parent process\n");
        }
    }
    return 0;
```

8. Write a program that create a child process with the fork () system call. The

parent process waits for the child process to finish before printing the contents of the current directory.

Screenshots of program running below, followed by code of program:





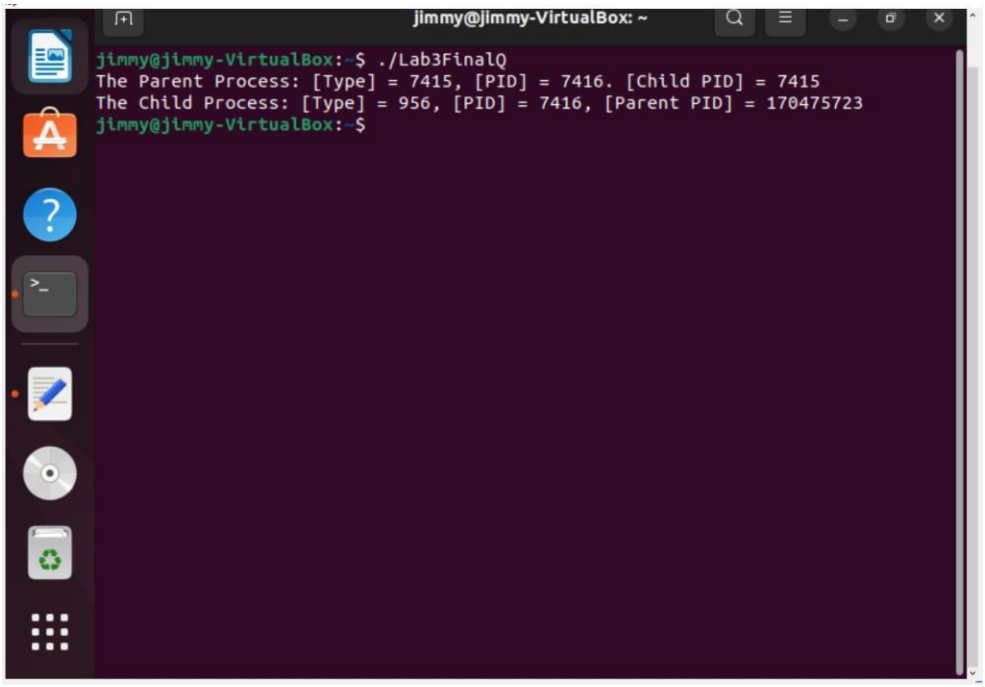
```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <dirent.h>

// James Battistoni CS 470 Question 9
int main(){
    pid_t pid;

    pid = fork();

    // Check to see if the Fork Failed
    if(pid < 0) {
        printf("Fork failed\n");
        return -1;
    // Child Process
    } else if(pid == 0) {
            printf("The Child Process is running\n");
            sleep(10);          // wait
    // Parent Process
    } else {
        // integer for the current status
        int currStat;
        waitpid(pid, &currStat, 0);
        // Print out that the child process has finished
```

C Tab Width: 4 Ln 16, Col 19 INS

```c
        int currStat;
        waitpid(pid, &currStat, 0);
        // Print out that the child process has finished
        printf("The Child Process has Finished\n");

        DIR *dir;
        struct dirent *ent;
        // Open the Directory
        if((dir = opendir(".")) != NULL) {
            while(( ent = readdir(dir)) != NULL) {
                // Print out content
                printf("%s\n", ent->d_name);
            }
            closedir(dir);      // Close the Directory
        } else {
            perror("");
            return EXIT_FAILURE;
        }
    }
    return 0;
}
```

C Tab Width: 4 Ln 16, Col 19 INS

9. Write a program that create a child process with the fork () system call and print its PID. Following a fork () system call, both parent and child processes print their process type and PID. Additionally, the parent process prints the PID of its child, and the child process prints the PID of its parent.




```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
// James Battistoni Lab3 Question 9

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

    if (child_PID == 0) {
        printf("The Child Process: [Type] = %d, [PID] = %d, [Parent PID] = %d\n", getppid(), getpid());
    } else {
        printf("The Parent Process: [Type] = %d, [PID] = %d. [Child PID] = %d\n", getpid(), child_PID);
    }
    return 0;
}
```

https://github.com/JimmyBattis/CS-OS-470-Labs