

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

It creates three child processes

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

It consumes about 200 MB of memory and .1% of the CPU on idle

3. How much memory is available in the system?

I have 4 GB of memory available

4. Which process consumes the most CPU?

Gnome-shell uses the most CPU

5. Which process has the most memory?

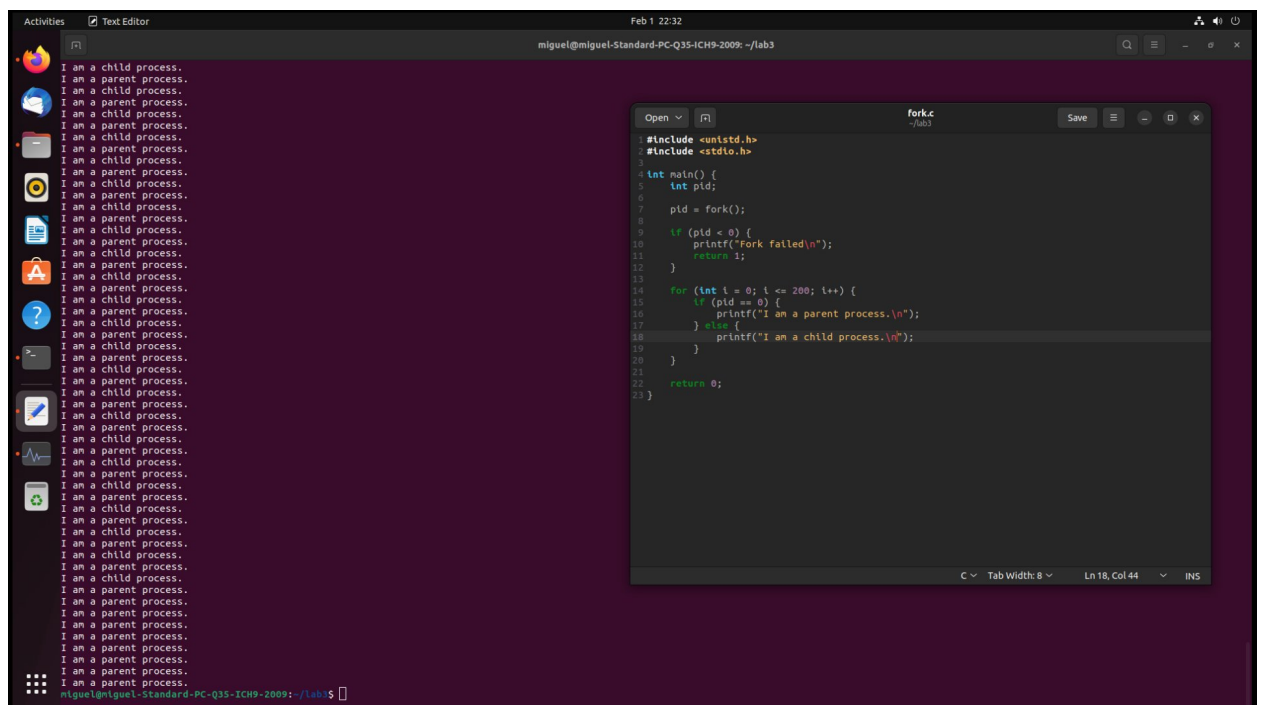
Gnome-shell uses the most memory

6. Could you please explain the following commands?

Apt-get, yum, wget, gzip, tar, rar

Apt-get and yum are package managers for Debian and Red Hat based distros respectively. Wget downloads files from a url. Gzip, tar, and rar are compression utilities that takes files and compresses them into a package.

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



The screenshot shows a Linux desktop environment. On the left, a terminal window displays the output of a program, showing a repeating pattern of "I am a parent process." and "I am a child process." lines. On the right, a code editor window titled "fork.c" shows the source code of the program. The code includes `<unistd.h>` and `<stdio.h>`, and implements a `main` function that uses `fork()` to create child processes. A loop in the parent process prints "I am a parent process." and a loop in the child process prints "I am a child process.".

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

int main() {
    int pid;

    pid = fork();

    if (pid < 0) {
        printf("Fork failed\n");
        return 1;
    }

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

    return 0;
}
```

8. Write a program that creates 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.

```
miguel@miguel-Standard-PC-Q35-ICH9-2009:~/lab3$ ./fork
I'm the child process! Psst, 1
I'm still the child process! Psst, 2
I'm still the child process! Psst, 3
I'm still the child process! Psst, 5
I'm still the child process! Psst, 8
I'm still the child process! Psst, 13
I'm still the child process! Psst, 21
I'm still the child process! Psst, 34
I'm still the child process! Psst, 55
I'm still the child process! Psst, 89
I'm still the child process! Psst, 144
I'm still the child process! Psst, 233
daz
fork
fork.c
..
foo
bar
.
miguel@miguel-Standard-PC-Q35-ICH9-2009:~/lab3$
```

9. Write a program that creates 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.

```
miguel@miguel-Standard-PC-Q35-ICH9-2009:~/lab3$ ./fork
I'm the parent process 8812! And this is my child, 8813!
I'm the child process 8813! And this is my parent, 8812!
miguel@miguel-Standard-PC-Q35-ICH9-2009:~/lab3$
```