Operating Systems Assignment 1

Clive Miller

Log

- 10 — 7 — 2025: Read Full assignment requirements
- 10 — 8 — 2025: Wrote 3 programs and did initial runs
- 10 — 9 — 2025: Wrote writeup and Latex doc

Part One, See Program One

Wrote Simple Program that completed the task.

Output:

```
clivemiller@clivepc:~/code/c-code/assignments/operating_sys_ass_1$ cc programPartOne.c
clivemiller@clivepc:~/code/c-code/assignments/operating_sys_ass_1$ ./a.out
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 0
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 1
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 2
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 3
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 4
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 5
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 6
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 7
My Program Name: ./a.out | PID: 6920 | Loop Iteration: 8
```

Part 2

Wrote the two small programs, compiled them into named executables.

- `programPartTwoFile.c` - `./pptf` to start, `./pptf &` to start and send to background
- `programPartTwoMath.c` - `./pptm` to start, `./pptm &` to start and send to background

kill any process with `pkill {name}`

Testing pptm

Then I sent pptm into the background and ran top:

```
PID  USER      PR  NI    VIRT   RES     SHR S  %CPU   %MEM  TIME+   COMMAND
7375 clivemi+  20   0    2552   1056    964 S  19.3   0.0   0:11.41 pp2m
```

I noticed that its using a sizeable amount of CPU (at least the CPU amount assigned to my WSL container) But it is also not using little if not no memory. Its id is 7375

Testing pptf

Next I sent pptf into the background and ran top:

```
PID   USER      PR   NI   VIRT   RES     SHR S  %CPU   %MEM  TIME+   COMMAND
11011 clivemi+  20   0    2684   1008    916 R  29.3   0.0   0:08.48 pp2f
```

It is using a lot more CPU. its ID is 11011.

Observations

Obviously PID is the process ID, the USER is the logged in user, the COMMAND is the process name, the time is time running (at least since the top command updated) The MEM is memory used, CPU is cpu used. PR is probably priority. VIRT is likely virtual memory. S is likely state, since it swaps between S (sleeping), R (running), etc.

That is all I could guess, looking it up I found:

- RES: Resident memory — actual physical RAM currently used by the process.
- SHR: Shared memory — part of RES that may be shared with other processes (e.g., shared libraries).
- NI: Niceness value — affects scheduling priority (-20 = highest priority, 19 = lowest).

Running both processes

Finally I ran them both:

```
PID   USER      PR   NI   VIRT     RES     SHR    S  %CPU   %MEM  TIME+   COMMAND
2307  clivemi+  20   0    1262884  67208   47412  S  32.3   0.8   3:20.08 node
15680 clivemi+  20   0    2684     1008    916    D  27.7   0.0   0:06.17 pp2f
2280  clivemi+  20   0    1177400  103372  47696  S  25.0   1.3   1:30.62 node
15738 clivemi+  20   0    2552     1044    948    S  21.3   0.0   0:03.29 pp2m
```

You can see some processes and our two here. Then I ran pkill and ran top again

```
PID  USER      PR   NI VIRT    RES     SHR    S   %CPU  %MEM   TIME+   COMMAND
2555 clivemi+  20   0  54.0g   1.0g    66528  S   3.0   13.0   2:55.58 node
2247 clivemi+  20   0  11.3g   145020  52100  S   0.3   1.8    1:40.74 node
2280 clivemi+  20   0  1197624 122864  47696  S   0.3   1.5    2:30.76 node
2896 clivemi+  20   0  110432  45440   21224  S   0.3   0.6    0:08.25 cpptools
```

Programs are in fact killed

Part 3

Got some exe information from objdump, along with all the section headers

- pp2f: file format elf64-x86-64
- pp2m: file format elf64-x86-64

by running `stat -c '%n %s bytes' exeName` I can get the byte count

```
clivemiller@clivepc:~/code/c-code/assignments/operating_sys_ass_1$ stat -c '%n %s bytes' pp2f
pp2f 16056 bytes
clivemiller@clivepc:~/code/c-code/assignments/operating_sys_ass_1$ stat -c '%n %s bytes' pp2m
pp2m 16016 bytes
```

I then ran these to try and get memory use but it is using so little it doesnt show:

```
clivemiller@clivepc:~/code/c-code/assignments/operating_sys_ass_1$ pidof pp2m
4715
clivemiller@clivepc:~/code/c-code/assignments/operating_sys_ass_1$ ps -p 4715 -o %mem
%MEM
 0.0
```

Source Code

programPartOne.c:

```c
#include <stdio.h>
#include <sys/types.h> // for pid_t
#include <unistd.h> // for sleep and getpid()

int main(int argc, char *argv[]) {
    pid_t pid; // Declare a variable of type pid_t to store the PID
    pid = getpid(); // Call getpid() to get the current process's PID
    int count = 0; // counter

    while (1) {
        // fprintf (printf to file) is printing to stderr the argv[0] (exe name), the pid (getpid()
    ), and the count (which is then iterated)
        fprintf(stderr, "My Program Name: %s | PID: %d | Loop Iteration: %d\n", argv[0], getpid(),
    count++);
        sleep(1); // sleep
    }

    return 0;
}
```

programPartTwoMath.c

```c
#include <stdio.h>

int main() {
    while (1) {
        int num = (100 + 20 / 23)*12; // do simple math
        fprintf(stderr, "Calc ans is: %d \n", num); // Print ans
    }

    return 0;
}
```

programPartTwoFile.c

```c
#include <stdio.h>

int main() {
     while (1) {
        FILE *file; // declare file pointer
        file = fopen("./test.txt", "w"); // open file in write mode
        fprintf(file, "a"); // overwrite with a a single char "a"
        fclose(file); // close
    }

    return 0;
}
```