Q & A related to Assignment 1

1. The pstat command requires us to list the information of utime and stime of a process. I tried to find this information in /proc/pid/stat file, but found utime and stime are always zero. Why this happened?

Answer: The unit of utime/stime is in CLOCK TICKS.

In Linux system, sysconf(_SC_CLK_TCK) returns 100 by default (although the default value might change since modern computers may dynamically change the value depending on current work load). So OS keeps time measurements in units of 100 cycles per second, or each system tick every 10 ms. In other words, if the value of utime is 4, then the value translate to 4/100 = 0.4 seconds, which is 40 ms. 0.4 seconds (or 40 ms) is the value that your code should print out.

For the above reason, you should cast the return value as float instead of int. In the above example, int 4 / int 100 will return 0, which is wrong.

Then why are utime and stime zero nearly always? This phenomenon is because your background process is nearly always sleeping. If you use usleep() rather than using sleep() to let linux schedule your process more frequently (like usleep(100), which let linux to schedule your process every 100 micro seconds), then you should be able to see that utime and stime are not zero anymore.

2. The pstat command requires us to list the information of RSS of a process. But both /proc/pid/stat and /proc/pid/status have the information of RSS, which one should I use?

Answer: You should use the one in /proc/pid/stat, the one listed in /proc/pid/status is VMRSS, which means number of pages the process has in virtual memory. Since virtual memory = the part in physical memory + the part on disk, thus VMRSS should be greater than RSS.

3. Where can I find the information (state, utime etc.) that pstat command required?

Answer: All the information pstat required can be found in /proc/pid/stat and /proc/pid/status two pseudo files. Their specifications have been posted in connex→resource→Assignment 1

4. Should PMan care about the background processes that were not executed by PMan bg?

Answer: No, you do not need to handle those background processes that were not executed by PMan bg.

5. Should PMan indicate to the user when a background process was terminated by a kill command outside of PMan?

Answer: Yes. A background process can be killed either inside PMan by bgkill or outside PMan e.g., by using command line % kill pid from another terminal. Either case, PMan should tell the user that the background process has been terminated. Please be reminded again that your code does not need to care about those background processes that were not executed by PMan bg.

To make your life easy, we do not require that your PMan immediately reports to the user when a background terminates. Such events can be reported at a later time when a user types bglist in PMan. Of course, you can also implement PMan in a way that it immediate reports the termination of a background process, but this is not mandatory.

6. How to implement the requirement stated in Question 5?

Answer: use system call waitpid with -1 as the first parameter, e.g.,

```
pid = waitpid(-1, &p_status, WNOHANG))
```

where -1 means wait for any child process, WNOHANG means "return immediately if no child has exited."

Note that waitpid returns the process ID of the terminated process whose status was reported. If unsuccessful, a -1 is returned.

You can further check the value in p_status to tell if a child process was killed or has exited. E.g.,

```
if (WIFSIGNALED(p_status)) {
     printf("Process %d was killed\n", pid);
     remove_process_from_list(pid);
}
if (WIFEXITED(p_status)) {
     printf("Process %d exits\n", pid);
     remove_process_from_list(pid);
```