210011 Dodajon Xunistdinov Lab 8

ps au

```
satoshi_khd@satoshiKhd:~/Desktop/SystemProgramming/lab9$ ps -au
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
satoshi+    1312  0.0  0.0 165268  1736 tty2     Ssl+ Mar28   0:00 /usr/libexec/
satoshi+    1315  0.0  0.0 225920    88 tty2     Sl+  Mar28   0:00 /usr/libexec/
satoshi+  151620  0.0  0.1  14424  5656 pts/0    Ss   12:10   0:00 bash
satoshi+  156124  0.0  0.0  15552  3456 pts/0    R+   13:51   0:00 ps -au
```

ps -ef

```
root      156771       2  0 14:05 ?        00:00:00 [kworker/3:0H-events_highpri
root      156792       2  0 14:06 ?        00:00:00 [kworker/0:2H-events_highpri
root      156799       2  0 14:06 ?        00:00:00 [kworker/1:0H-events_highpri
root      156825       2  0 14:06 ?        00:00:00 [kworker/2:1H-events_highpri
root      156841       2  0 14:07 ?        00:00:00 [kworker/2:2-events]
satoshi+  156849  151620  0 14:07 pts/0    00:00:00 ps -ef
```

ps -ax

```
 156089 ?        D<     0:00 [kworker/u9:1+i915_flip]
 156113 ?        I      0:00 [kworker/u8:1-flush-179:0]
 156191 ?        I      0:00 [kworker/u8:2]
 156192 ?        I      0:00 [kworker/1:3-i915-unordered]
 156199 ?        SNsl   0:00 /usr/libexec/tracker-extract-3
 156206 pts/0    R+     0:00 ps -ax
```

ps -l

```
satoshi_khd@satoshiKhd:~/Desktop/SystemProgramming/lab9$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  151620  151594  0  80   0 -  3606 do_wai pts/0    00:00:00 bash
4 R  1000  156985  151620  0  80   0 -  3888 -      pts/0    00:00:00 ps
```

ps -f

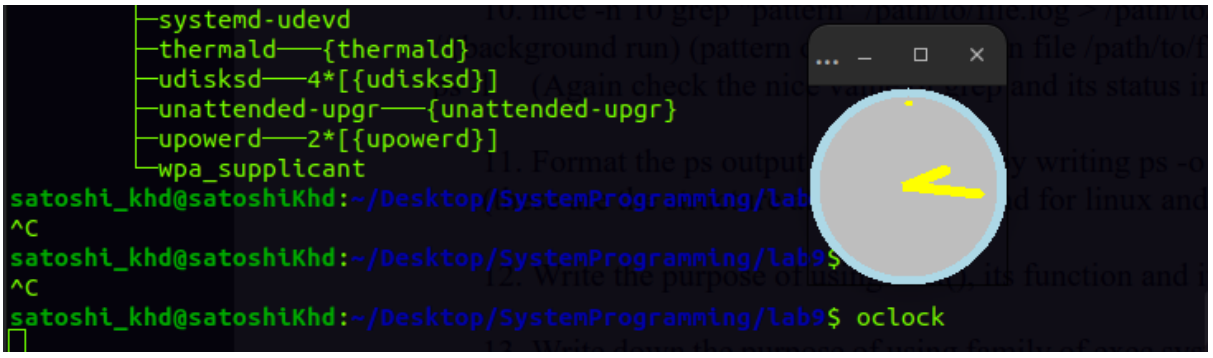
```
satoshi_khd@satoshiKhd:~/Desktop/SystemProgramming/lab9$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
satoshi+  151620  151594  0 12:10 pts/0    00:00:00 bash
satoshi+  157039  151620  0 14:12 pts/0    00:00:00 ps -f
```

ps -x

```
 156236 ?        Sl     0:00 /usr/lib/libreoffice/program/oosplash --impress fil
 156251 ?        Sl     0:24 /usr/lib/libreoffice/program/soffice.bin --impress
 156587 ?        Sl     0:00 /snap/firefox/4033/usr/lib/firefox/firefox -content
 157080 pts/0    R+     0:00 ps -x
```

pstree

oclock



9)

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

int main()
{
	system("grep \"pattern\" ./sometext.txt > ./output.txt\n");
	system("ps -l | grep grep\n");
	printf("Done.\n");
	return 0;
}
```

```
satoshi_khd@satoshiKhd:~/Desktop/SystemProgramming/lab9$ ./example
0 S  1000    9694    9692  0  80   0 -  3023 pipe_r pts/1    00:00:00 grep
Done.
```

10)

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


int main() {
        char *command = "nice -n 10 grep \"pattern\" ./sometext.txt > ./output.txt\>
        char *check__command = "ps -l \n";

        int result = system(command);
        result == -1 ? perror("-1") : printf("successfully run \n%s", command);

        printf("checking the nice value of the grep\n");
        system(check__command);


        return 0;
}
```

```
satoshi_khd@satoshiKhd:~/Desktop/SystemProgramming/lab9$ ./task10
successfully run
nice -n 10 grep "pattern" ./sometext.txt > ./output.txt
checking the nice value of the grep
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000   43719   43701  0  80   0 -  3573 do_wai pts/0    00:00:00 bash
0 S  1000   43845   43719  0  80   0 -   694 do_wai pts/0    00:00:00 task10
0 S  1000   43848   43845  0  80   0 -   723 do_wai pts/0    00:00:00 sh
4 R  1000   43849   43848  0  80   0 -  3888 -      pts/0    00:00:00 ps
```

11) Format the ps output as per choice by writing ps -o pid, vsz,cpu,tt
(these are the structure attributes use cmd for linux and command for mac)

```
satoshi_khd@satoshiKhd:~/Desktop/SystemProgramming/lab9$ ps -o pid,vsz,tt,cpu
    PID    VSZ TT       CPU
  43719  14292 pts/0      -
  43964  15524 pts/0      -
```

12) Write the purpose of using fork(), its function and its syntax

purpose -> simply, copy or instance of the parent process that's been executing.

function -> in simple words, fork function create copy of the parent process, then both processes will do the same, but child process can run another code as well, child process will inherit almost all the features that had parent except of PID it'll be different(guess, 'cuz it's different process).

syntax ->

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

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

	switch(pid):
		case -1: perror("error while forking");
		case 0: printf("Child process: \n%s", getpid());
		case 1: printf("Parent process: \n%s", getpid());

	return 0;
};
```

13) Write down the purpose of using family of exec system calls, specifically execl, execlp, execle {list of arguments} vs execv, execvp, execve {array of arguments}. What is the difference among these system calls?

exec -> replaces current process with a new process

execl, execlp, execle => take a variable number of arguments (list of arguments).

int execl, execlp, execle(const char *path, const char *arg0, ….,)

execv, execvp, execve => take an array of the arguments (array of arguments)/.

int execv, execve(const char *path, char *const argv[]);
int execvp(const char file, char *const argv[]);

both types are pretty much similar, there are differences only how they take parameters and what they take as parameters

execlp -> searches for the executable file in the given path

execvp -> does the same thing as execlp, but takes only file as an argument instead of path.