ITF22519, Fall 2021

Østfold University College

Processes & System Calls

A Look from Implementation Points of Views

Østfold University College

Expected conduct for lab assignment

Students can discuss an assignment with other students and ask TAs or teachers for assistance. However, each student has to complete his/her assignment individually. Copying of another' assignment or copying code from the Internet is strongly prohibited. We assume that all programming and exercises throughout this course is your own. If we are not sure that the work you submitted demonstrates your clear understanding, we may request that you give an oral presentation.

Discussion is permitted but copying is not.

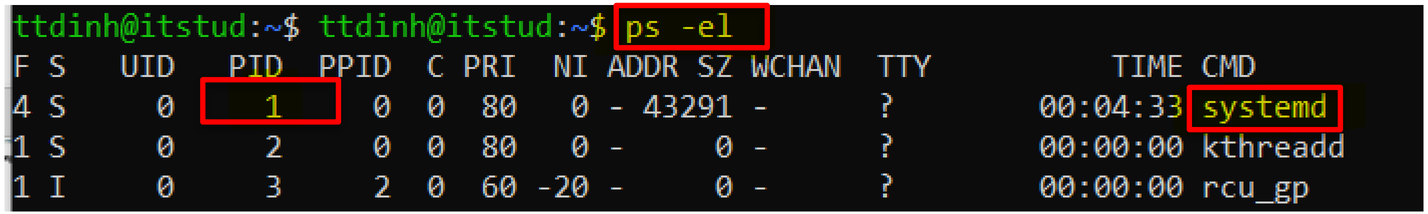
Plagiarism

- If plagiarism is detected, students will get 0 points for the lab session with plagiarism
- HiOF policy: https://www.hiof.no/studier/eksamen/fusk-og-plagiat/

Process

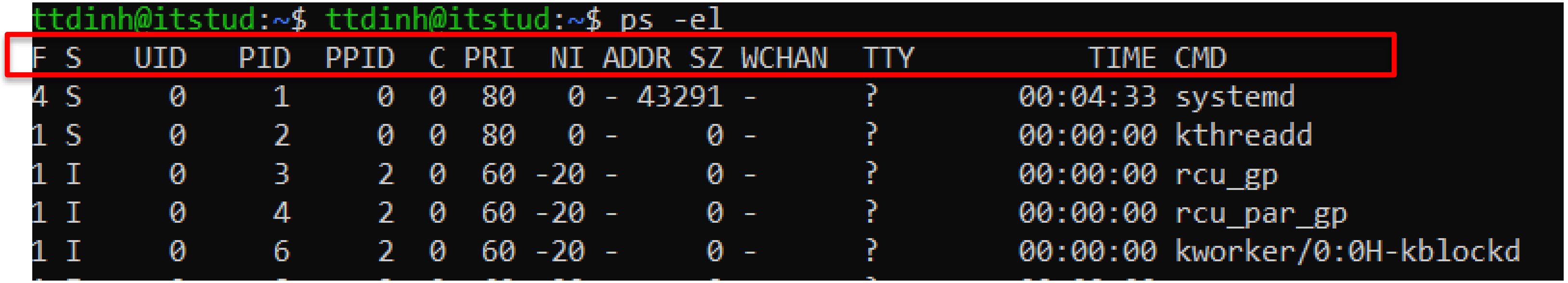
The first process

- /sbin/init or systemd depending on Linux distribution.
  - Super parent process
  - Starts other proccesses
  - PID of 1



```
ttdinh@itstud:~$ ttdinh@itstud:~$ ps -el
F S   UID     PID   PPID  C PRI   NI ADDR SZ WCHAN   TTY          TIME CMD
4 S     0       1      0  0  80    0 - 43291 -       ?        00:04:33 systemd
1 S     0       2      0  0  80    0 -     0 -       ?        00:00:00 kthreadd
1 I     0       3      2  0  60  -20 -     0 -       ?        00:00:00 rcu_gp
```

Process table

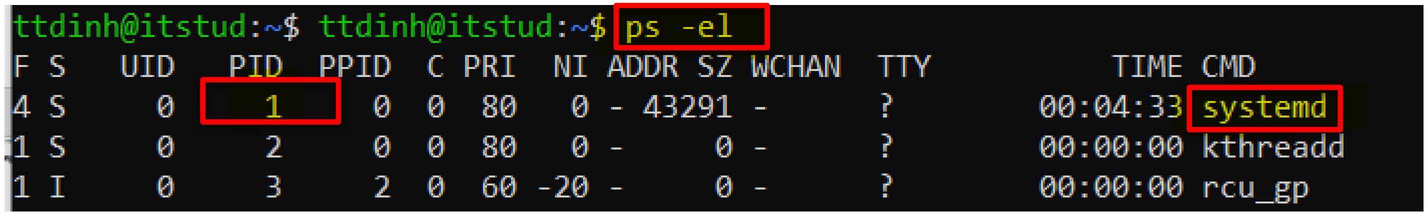


```
ttdinh@itstud:~$ ttdinh@itstud:~$ ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 43291 -      ?        00:04:33 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0     3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
1 I     0     4     2  0  60 -20 -     0 -      ?        00:00:00 rcu_par_gp
1 I     0     6     2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H-kblockd
```

Østfold University College

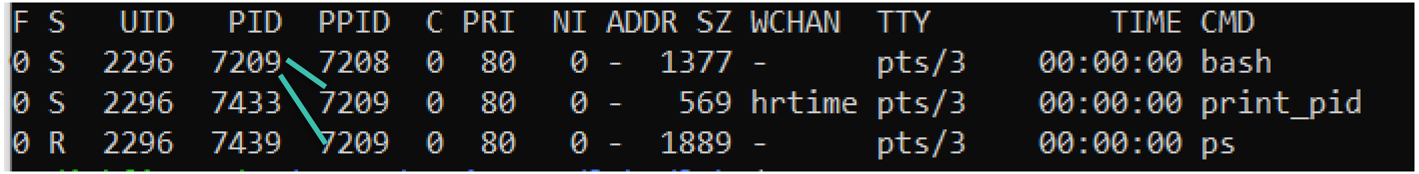
The first process

- **/sbin/init** or **systemd** depending on Linux distribution.
  - Super parent process
  - Reponsible for forking other processes
  - PID of 1



```
ttdinh@itstud:~$ ttdinh@itstud:~$ ps -el
F S   UID    PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0      1     0  0  80   0 - 43291 -      ?        00:04:33 systemd
1 S     0      2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 I     0      3     2  0  60 -20 -     0 -      ?        00:00:00 rcu_gp
```

Process tree



```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  2296  7209  7208  0  80   0 -  1377 -      pts/3    00:00:00 bash
0 S  2296  7433  7209  0  80   0 -   569 hrtime pts/3    00:00:00 print_pid
0 R  2296  7439  7209  0  80   0 -  1889 -      pts/3    00:00:00 ps
```

Foreground and background processes

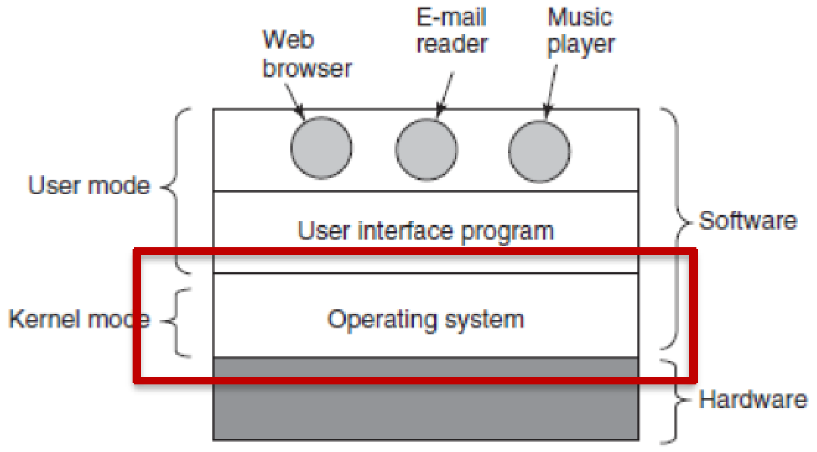
- Foreground process
  - Every program run in foreground by default
  - Requires users to start or interact
  - No other processes can be run while foreground process is running
- Background process
  - A process runs in background without keyboard input and waits until keyboard input is required
  - Several processes can run in parallel
  - Adding **ampersand &** at the end of command name

System calls

Østfold University College

Where the OS fits in?

- Kernel mode
    - Access to all hardware
    - Execute all instructions
    - Operating Systems
- User mode
    - Limited access to instructions
    - Rest of software runs in user mode
    - Limited capacity
    - User interface program: Shell, GUI



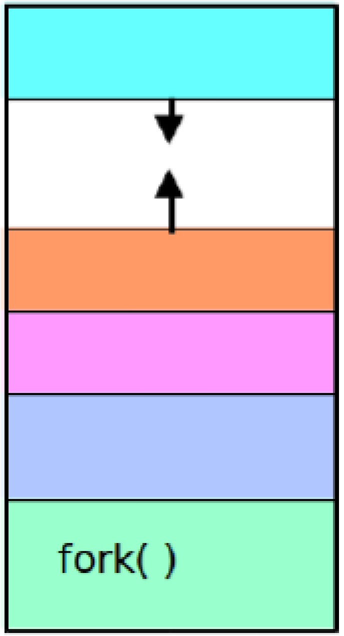
Where the OS fits in

fork(): Creating a new process

```
#include <sys/types.h>
#include <unistd.h>
// ...
pid_t child;
child = fork();
// ...
```

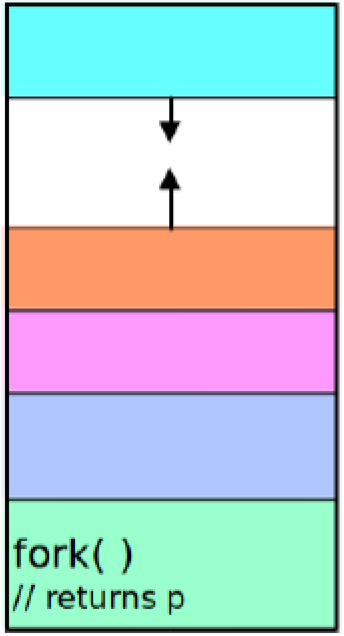
- Only way to create a new process
- Create an exact duplicate of the original process
- After creation, parent and child processes go different ways

Before fork()

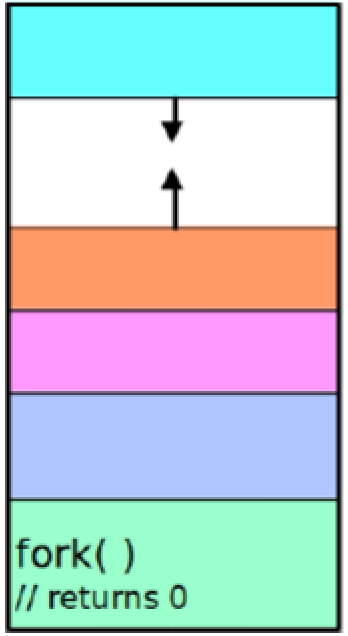


parent process

After fork()



parent process



child process
(process id = p)

wait(): Waiting on a child process

- Blocks parent process until the child process exits or a signal is received

- After child process terminates, parent continues its execution after wait system call instruction

wait(): Waiting on a child process

```
#include <sys/types.h>
#include <sys/wait.h>
...

int status = 0;
....

pid_t child = fork();
if ( 0 == child){
      // do something here
} else if ( 0 < child ) {
      wait(&status);
      printf("child process is done, status is: %d\n", status);
      // do something else here
     return 0;
} else {
     perror("fork");
     exit(-1);
}
}
```

exec(): Making a process running another program

- Replace current process by a new program
- Load the program into current process place and run the program

kill(): Sending a signal to another process

- Used to send all kinds of signal to a process.

```
#include <sys/types.h>
#include <signal.h>
int kill(pid_t pid, int sig);
```

Q&A