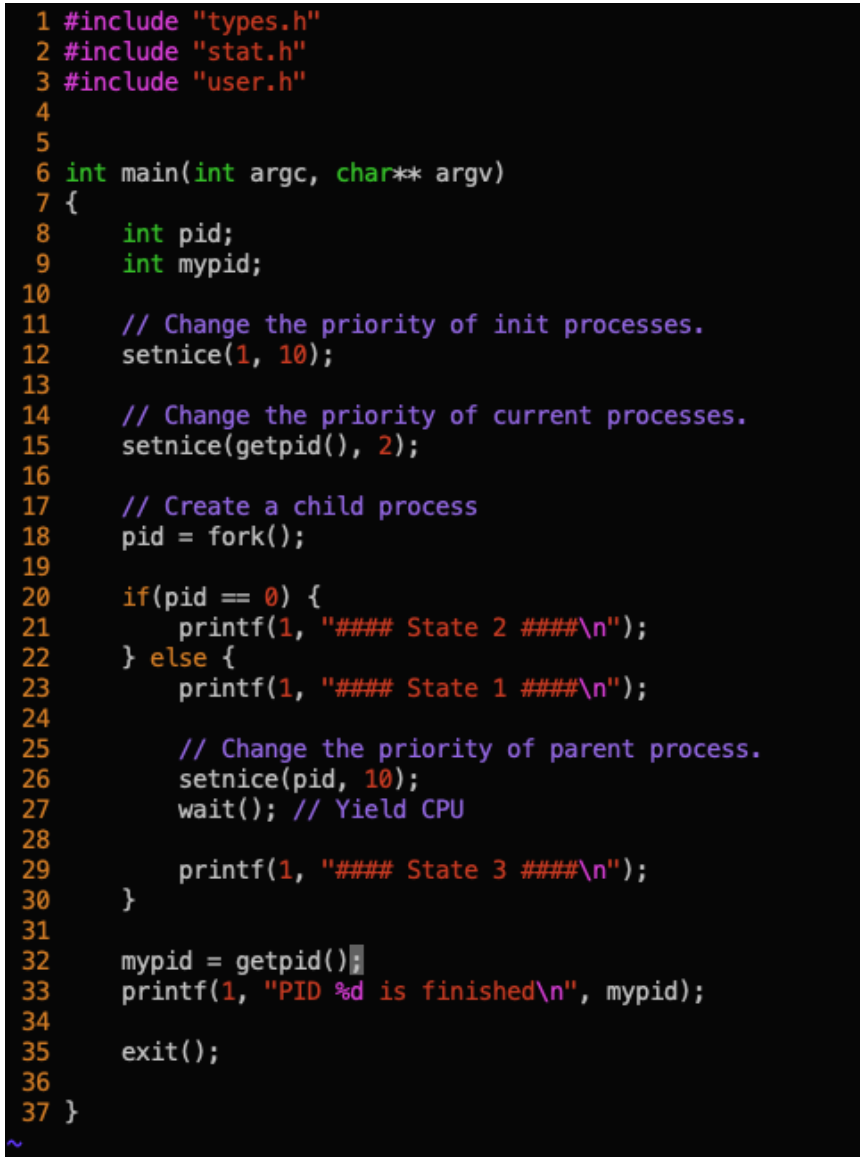
Project 2. Priority Scheduling

- Implement priority scheduler in xv6
  - The lower nice value, the higher priority
  - The highest priority process should be chosen for next running
    - Tiebreak: Arbitrary

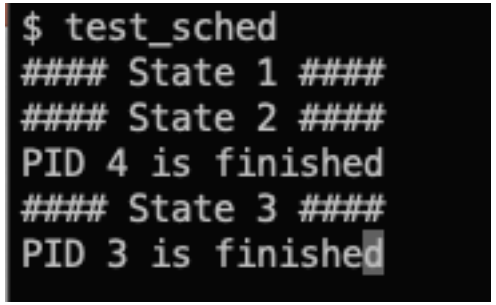
- Scheduler runs only when a change occurs in process priorities
  - DO NOT call the scheduler on the timer interrupt
  - When a process calls fork(), the nice value of child process is set to 5.

test_sched.c

- Add test_sched.c
- ./test_sched



```c
#include "types.h"
#include "stat.h"
#include "user.h"


int main(int argc, char** argv)
{
    int pid;
    int mypid;

    // Change the priority of init processes.
    setnice(1, 10);

    // Change the priority of current processes.
    setnice(getpid(), 2);

    // Create a child process
    pid = fork();

    if(pid == 0) {
        printf(1, "#### State 2 ####\n");
    } else {
        printf(1, "#### State 1 ####\n");

        // Change the priority of parent process.
        setnice(pid, 10);
        wait(); // Yield CPU

        printf(1, "#### State 3 ####\n");
    }

    mypid = getpid();
    printf(1, "PID %d is finished\n", mypid);

    exit();

}
```



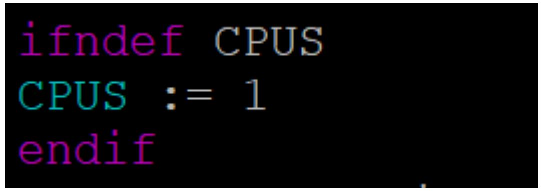
```
$ test_sched
#### State 1 ####
#### State 2 ####
PID 4 is finished
#### State 3 ####
PID 3 is finished
```

Hand-in Procedures (1/2)

- Download template
  - https://github.com/eunjicious/xv6-ssu.git (pull or clone)
  - `tar xvzf xv6_ssu_syscall.tar.gz`

- Rename directory
  - `mv xv6_ssu_syscall xv6_ssu_sched`

- Add test_sched.c to your codes and modify Makefile properly
- Build with CPUS=1 flag
  - Makefile



```
ifndef CPUS
CPUS := 1
endif
```

Hand-in Procedures (2/2)

- Compress your code (ID: 20201234)
  - `$tar cvzf xv6_ssu_sched_20201234.tar.gz xv6_ssu_sched`
  - Please command `$make clean` before compressing

- Submit your `tar.gz` file through myclass.ssu.ac.kr

- NO DELAY is allowed !!
- PLEASE DO NOT COPY !!