Home Assignment - 3 (XV6)

Name - Rajeev Kumar
ID - 12341700

- **Task 1: Update `proc.h` – Add Priority Field**

  File Name: `proc.h`
  Code Added: inside `struct proc` -

  ```
  int priority;
  ```

- **Task 2: Initialize Priority in `proc.c`**

  File Name: `proc.c`
  Code Added:

  ```
  p->priority = 50;
  ```

- **Task 2a & 2b: Implement Priority Scheduler in `proc.c`**

  File Name: `proc.c`
  Code Added:

  ```
  void scheduler(void) {
    struct proc *p;
    struct cpu *c = mycpu();
    c->proc = 0;
    for (;;) {
      sti();
      struct proc* highest_priority_p = 0;
      int highest_priority = 1000; // Start with a large value
      acquire(&ptable.lock);
  ```

```
    for (p = ptable.proc; p < &ptable.proc[NPROC]; p++) {
      if (p->state == RUNNABLE) {
        if (p->priority < highest_priority) {
          highest_priority = p->priority;
          highest_priority_p = p;
        }
      }
    }
    if (highest_priority_p != 0) {
      p = highest_priority_p;
      c->proc = p;switchuvm(p);
      p->state = RUNNING;
      swtch(&(c->scheduler), p->context);
      c->proc = 0;
    }
    release(&ptable.lock);
  }
}
```

- **Task 2c: Implement setpriority in `sysproc.c`**

  File Name: `sysproc.c`
  Code Added:

```
int sys_setpriority(void) {
    int priority;
    if (argint(0, &priority) < 0) return -1;
    myproc()->priority = priority;
    return 0;
}
```

- **Task 3: Update System Call Files**

  File Name: `syscall.h`
  Code Added:

  ```
  #define SYS_setpriority 30
  ```

  File Name: `syscall.c`
  Code Added:

  ```
  extern int sys_setpriority(void);
  ```

  And inside `syscalls[ ]` table:

  ```
  [SYS_setpriority] sys_setpriority,
  ```

- **Task 4: User-Space Interface**

  File Name: user.h

  Code Added:

  ```
  int setpriority(int);
  ```

  File Name: `usys.S`
  Code Added:

  ```
  SYSCALL(setpriority)
  ```

- **Task 5: Create User-Space Test Program prioritytest.c**

  File Name: prioritytest.c

  Code Added:

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

int main(int argc, char *argv[])
{
    int pid1, pid2, pid3, pid4, pid5;
    printf(1, "Starting priority scheduling test...\n");
    pid1 = fork();
    if (pid1 == 0) {
        setpriority(5);
        printf(1, "Child 1 (pid %d) with high priority (5) started.\n", getpid());
        for (int i = 0; i < 50000000; i++) {}
        printf(1, "Child 1 finished.\n");
        exit();
    }

    pid2 = fork();
    if (pid2 == 0) {
        setpriority(10);
        printf(1, "Child 2 (pid %d) with priority (10) started.\n", getpid());
        for (int i = 0; i < 50000000; i++) {}
        printf(1, "Child 2 finished.\n");
        exit();
    }

    pid3 = fork();
    if (pid3 == 0) {
        setpriority(15);
        printf(1, "Child 3 (pid %d) with medium priority (15) started.\n",
getpid());
        for (int i = 0; i < 50000000; i++) {}printf(1, "Child 3 finished.\n");
        printf(1, "Child 3 finished.\n");
        exit();
    }

    pid4 = fork();
    if (pid4 == 0) {
        setpriority(20);
        printf(1, "Child 4 (pid %d) with priority (20) started.\n", getpid());
        for (int i = 0; i < 50000000; i++) {}
        printf(1, "Child 4 finished.\n");
        exit();
    }

    pid5 = fork();
    if (pid5 == 0) {
        setpriority(25);
        printf(1, "Child 5 (pid %d) with low priority (25) started.\n", getpid());
        for (int i = 0; i < 50000000; i++) {}
        printf(1, "Child 5 finished.\n");
        exit();
    }

    wait();
    wait();
    wait();
    wait();
    wait();
    printf(1, "Priority scheduling test complete.\n");
    exit();
}
```

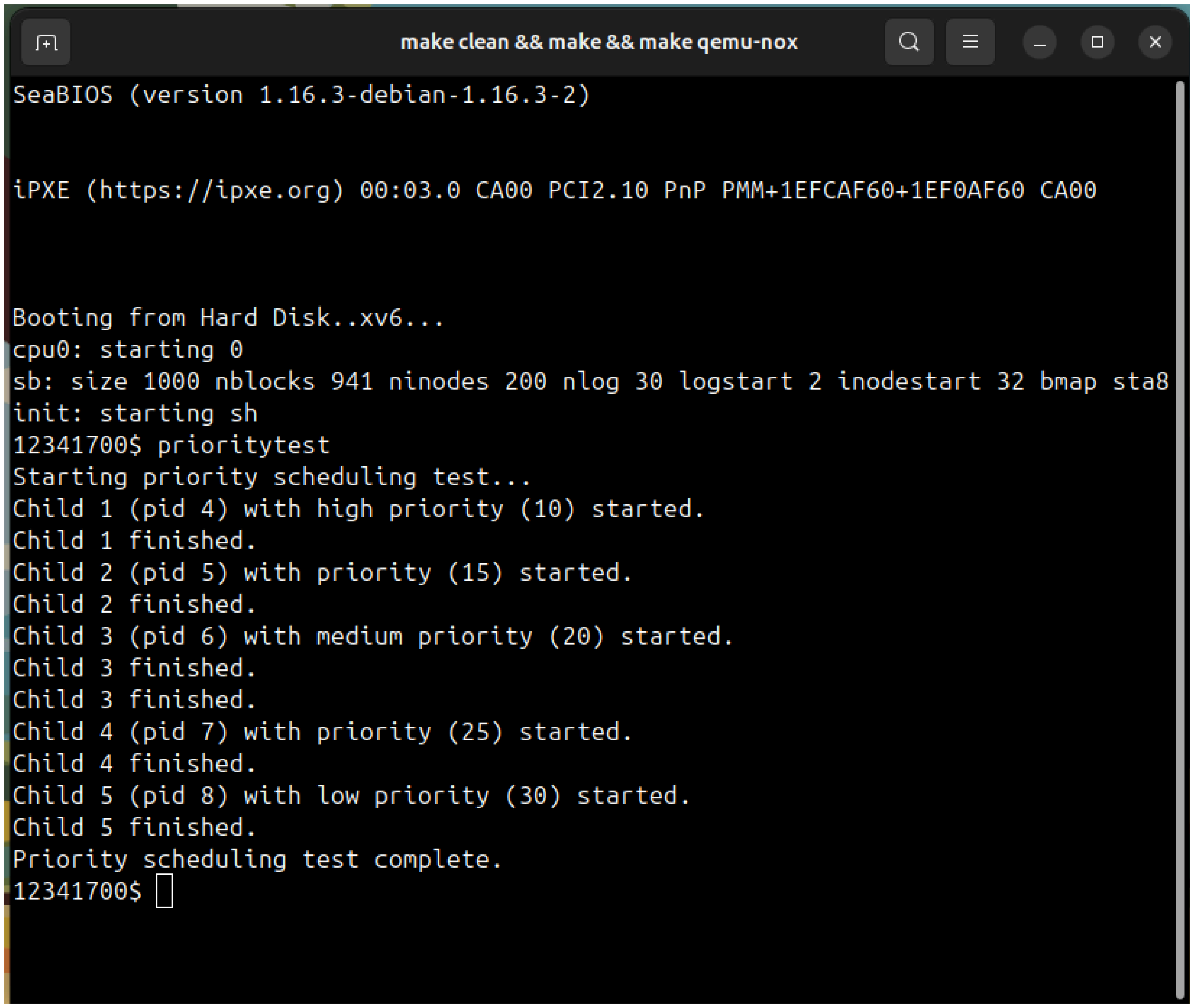
- **Task6: Add to UPROGS in Makefile**

  File Name: Makefile

  Code Added:

  ```
                  _prioritytest\
  ```

OUTPUT :



```
SeaBIOS (version 1.16.3-debian-1.16.3-2)


iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+1EFCAF60+1EF0AF60 CA00



Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap sta8
init: starting sh
12341700$ prioritytest
Starting priority scheduling test...
Child 1 (pid 4) with high priority (10) started.
Child 1 finished.
Child 2 (pid 5) with priority (15) started.
Child 2 finished.
Child 3 (pid 6) with medium priority (20) started.
Child 3 finished.
Child 3 finished.
Child 4 (pid 7) with priority (25) started.
Child 4 finished.
Child 5 (pid 8) with low priority (30) started.
Child 5 finished.
Priority scheduling test complete.
12341700$
```