CS302 OS Week11 Assignment - Report

Name: 刘仁杰
SID: 11911808

EX0. CPU Scheduling

Time	HRRN	FIFO/FCFS	RR	SJF	Priority
1	A	A	A	A	A
2	A	A	A	A	B
3	A	A	B	A	A
4	A	A	A	A	D
5	B	B	D	B	D
6	D	D	A	D	C
7	D	D	C	D	C
8	C	C	D	C	C
9	C	C	C	C	A
10	C	C	C	C	A
Avg. Turn-around Time	4.5	4.5	4.75	4.5	4.25

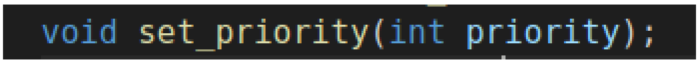
EX1. Implement a syscall that can set the priority of current process

Design idea:

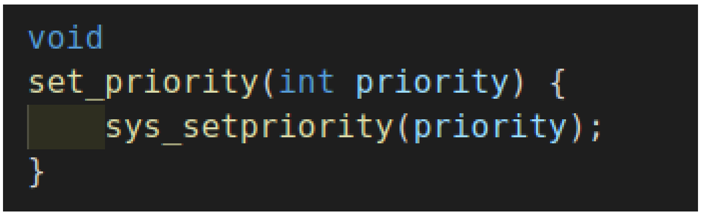
- First, since `ex1.c` includes `ulib.h`, we can create a function `set_priority()` in `ulib.c` and define it in `ulib.h`. Then the user program can call this function to set priority.
- In the implementation of `set_priority()`, we invoke call to `sys_setpriority()` which is implemented in `user/libs/syscall.c`.
- In `sys_setpriority()`, `syscall()` is invoked with parameter `SYS_labschedule_set_priority` and `priority`.
- `ecall` then trigger trap into the kernel `syscall()`, where we register `SYS_labschedule_set_priority` with function `sys_setpriority()`.
- In kernel's `sys_setpriority()`, current process's priority is set to the passed priority.

Modified codes:

ulib.h and ulib.c:

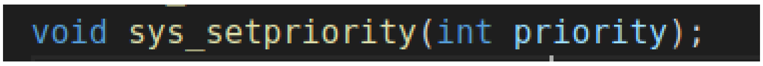


```
void set_priority(int priority);
```

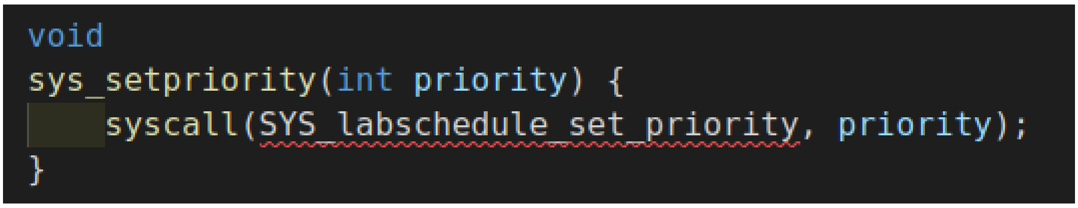


```c
void
set_priority(int priority) {
    sys_setpriority(priority);
}
```

user/libs/syscall.h and user/libs/syscall.c:

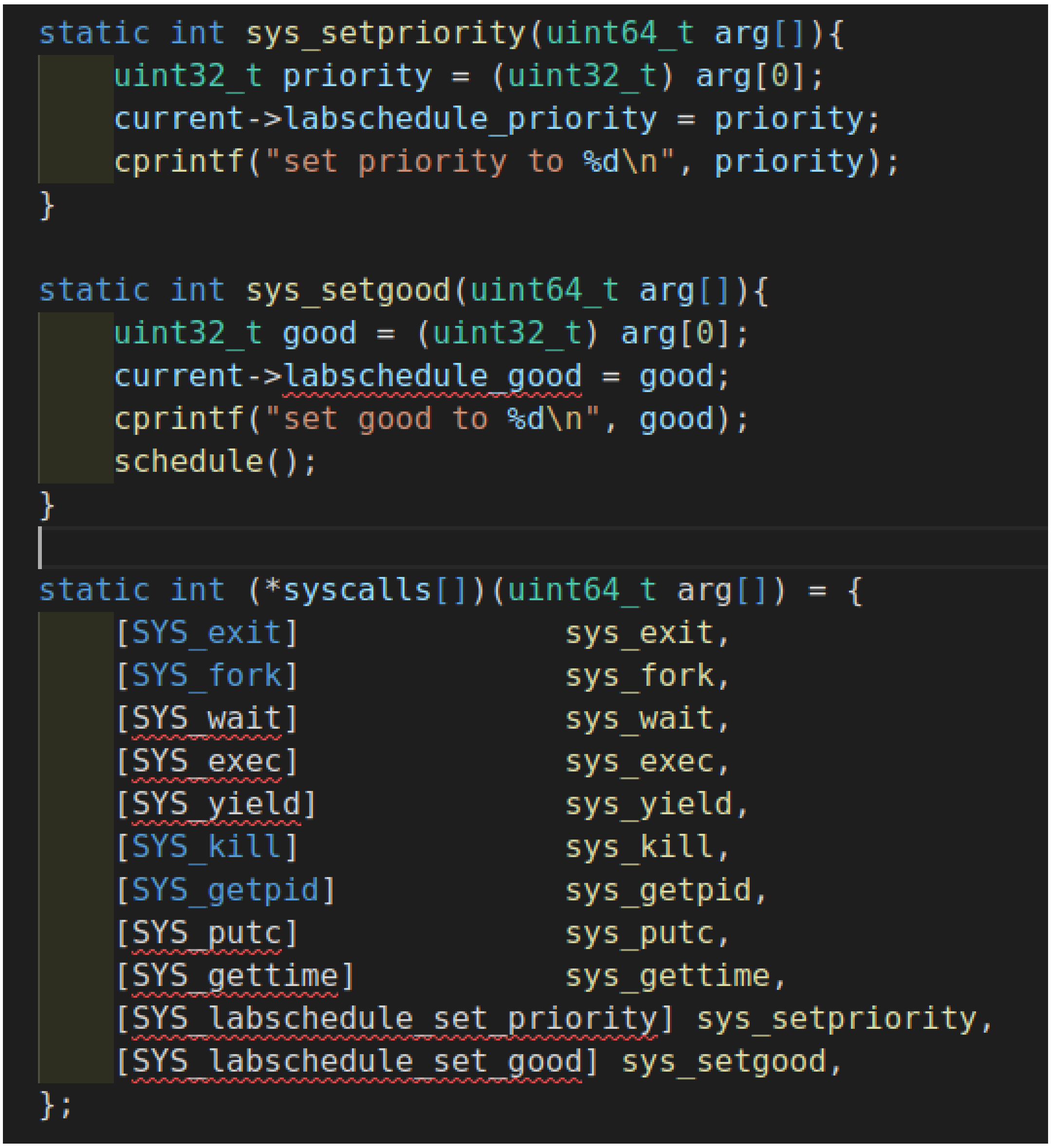


```c
void sys_setpriority(int priority);
```



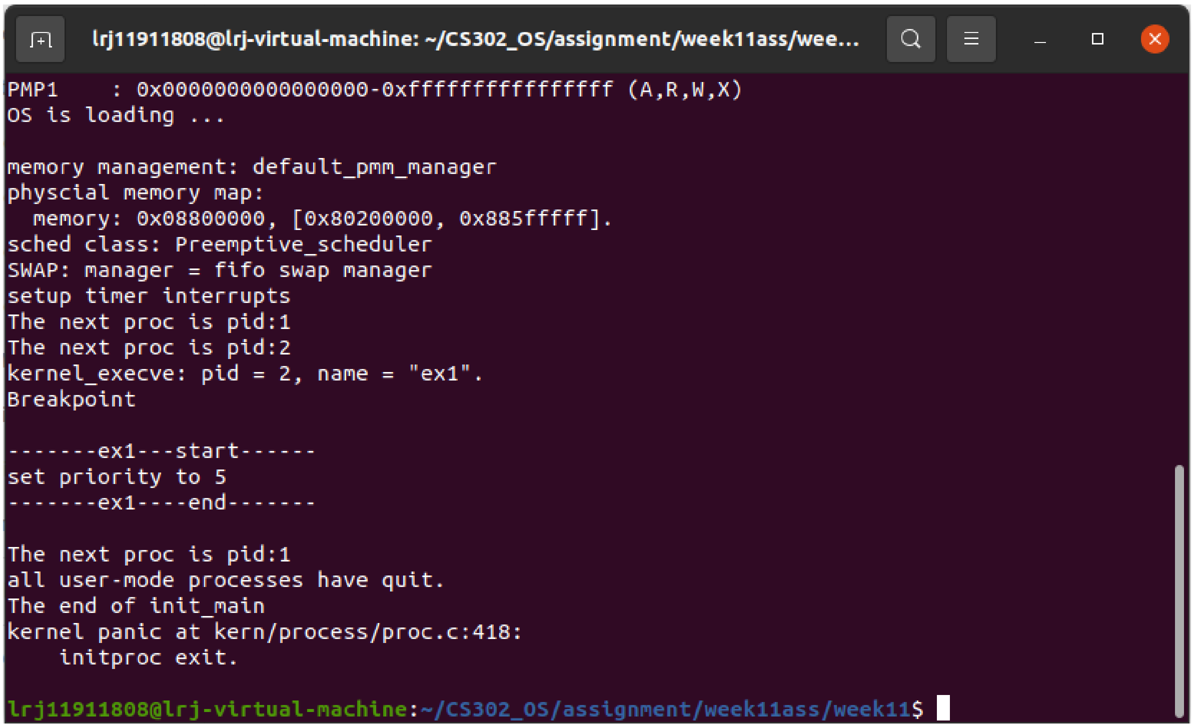
```c
void
sys_setpriority(int priority) {
    syscall(SYS_labschedule_set_priority, priority);
}
```

kernel/syscall/syscall.c:



```c
static int sys_setpriority(uint64_t arg[]){
    uint32_t priority = (uint32_t) arg[0];
    current->labschedule_priority = priority;
    cprintf("set priority to %d\n", priority);
}

static int sys_setgood(uint64_t arg[]){
    uint32_t good = (uint32_t) arg[0];
    current->labschedule_good = good;
    cprintf("set good to %d\n", good);
    schedule();
}

static int (*syscalls[])(uint64_t arg[]) = {
    [SYS_exit]              sys_exit,
    [SYS_fork]              sys_fork,
    [SYS_wait]              sys_wait,
    [SYS_exec]              sys_exec,
    [SYS_yield]             sys_yield,
    [SYS_kill]              sys_kill,
    [SYS_getpid]            sys_getpid,
    [SYS_putc]              sys_putc,
    [SYS_gettime]           sys_gettime,
    [SYS_labschedule_set_priority] sys_setpriority,
    [SYS_labschedule_set_good] sys_setgood,
};
```

Result:


lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week11ass/wee...

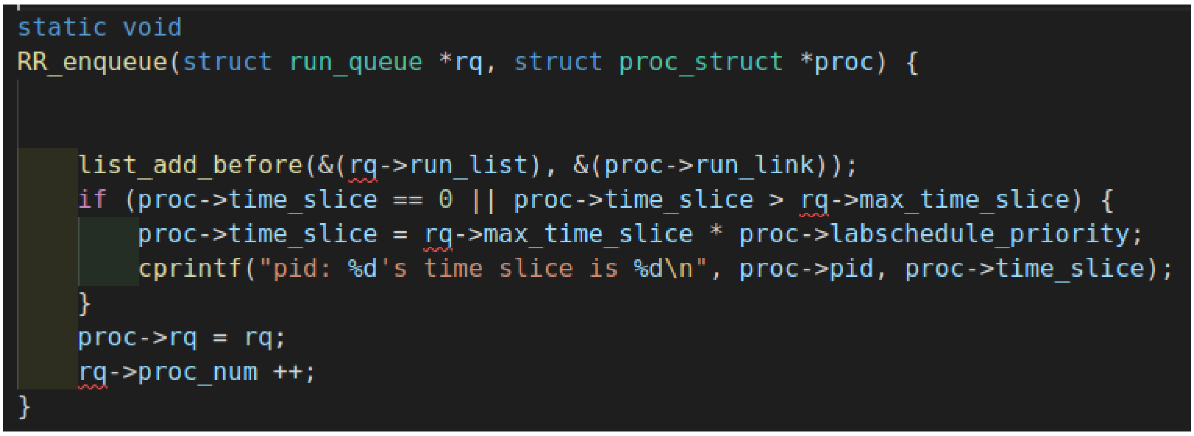
```
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
OS is loading ...

memory management: default_pmm_manager
physcial memory map:
  memory: 0x08800000, [0x80200000, 0x885fffff].
sched class: Preemptive_scheduler
SWAP: manager = fifo swap manager
setup timer interrupts
The next proc is pid:1
The next proc is pid:2
kernel_execve: pid = 2, name = "ex1".
Breakpoint

-------ex1---start------
set priority to 5
-------ex1----end-------

The next proc is pid:1
all user-mode processes have quit.
The end of init_main
kernel panic at kern/process/proc.c:418:
    initproc exit.

lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week11ass/week11$ 
```

EX2. the RR scheduling Algorithm based on Priority

Design idea:

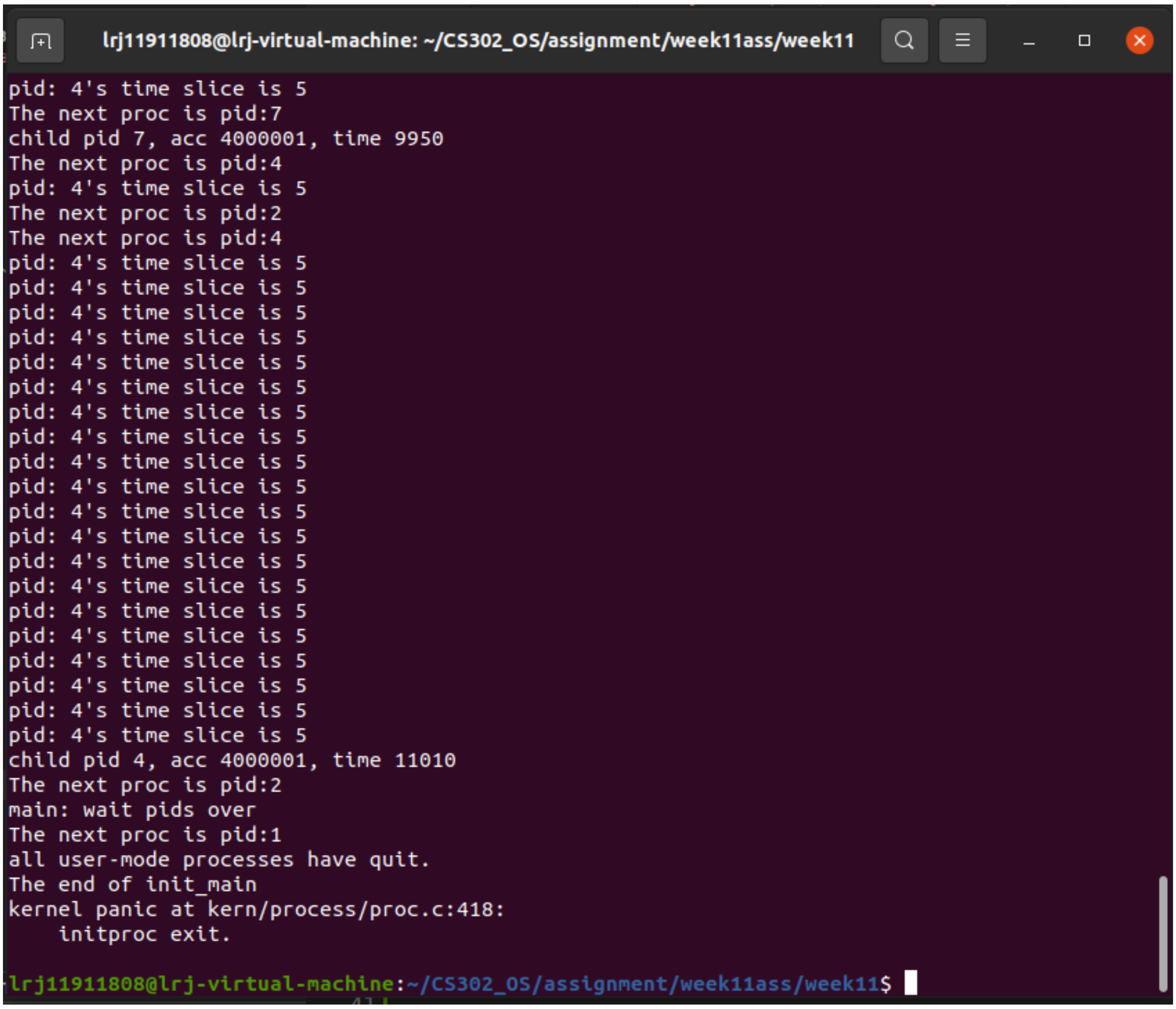
- For proiority based RR, we just need to modify the time slice to be `max_time_slice * priority` when a process is enqueued into `run_queue`.

Modified codes:



```
static void
RR_enqueue(struct run_queue *rq, struct proc_struct *proc) {


    list_add_before(&(rq->run_list), &(proc->run_link));
    if (proc->time_slice == 0 || proc->time_slice > rq->max_time_slice) {
        proc->time_slice = rq->max_time_slice * proc->labschedule_priority;
        cprintf("pid: %d's time slice is %d\n", proc->pid, proc->time_slice);
    }
    proc->rq = rq;
    rq->proc_num ++;
}
```

Results:


lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/week11ass/week11

```
pid: 4's time slice is 5
The next proc is pid:7
child pid 7, acc 4000001, time 9950
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:2
The next proc is pid:4
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
child pid 4, acc 4000001, time 11010
The next proc is pid:2
main: wait pids over
The next proc is pid:1
all user-mode processes have quit.
The end of init_main
kernel panic at kern/process/proc.c:418:
    initproc exit.

lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week11ass/week11$ 
```

```
memory management: default_pmm_manager
physcial memory map:
  memory: 0x08800000, [0x80200000, 0x885fffff].
sched class: RR_scheduler
pid: 1's time slice is 5
SWAP: manager = fifo swap manager
setup timer interrupts
The next proc is pid:1
pid: 2's time slice is 5
The next proc is pid:2
kernel_execve: pid = 2, name = "ex2".
Breakpoint
pid: 3's time slice is 5
pid: 4's time slice is 5
pid: 5's time slice is 5
pid: 6's time slice is 5
pid: 7's time slice is 5
main: fork ok,now need to wait pids.
The next proc is pid:3
set priority to 3
pid: 3's time slice is 15
The next proc is pid:4
set priority to 1
pid: 4's time slice is 5
The next proc is pid:5
set priority to 4
pid: 5's time slice is 20
The next proc is pid:6
```

```
set priority to 5
pid: 6's time slice is 25
The next proc is pid:7
set priority to 2
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
```

```
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
pid: 6's time slice is 25
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:6
child pid 6, acc 4000001, time 6880
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
pid: 5's time slice is 20
The next proc is pid:2
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:5
child pid 5, acc 4000001, time 7760
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
```

```
The next proc is pid:2
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
pid: 3's time slice is 15
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:3
child pid 3, acc 4000001, time 8840
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:2
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
pid: 7's time slice is 10
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:7
child pid 7, acc 4000001, time 9950
The next proc is pid:4
pid: 4's time slice is 5
The next proc is pid:2
The next proc is pid:4
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
```

```
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
pid: 4's time slice is 5
child pid 4, acc 4000001, time 11010
The next proc is pid:2
main: wait pids over
The next proc is pid:1
all user-mode processes have quit.
The end of init_main
kernel panic at kern/process/proc.c:418:
    initproc exit.
```

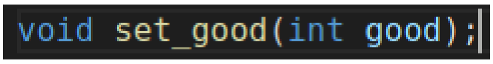
EX3. Preemptive process scheduling [40pts]

Design idea:

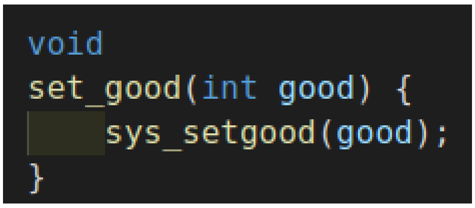
- Nearly the same as EX1, first we need to implment `set_good` syscall along the invoking trace.
- After implementing `set_good`, we need to overwrite a new preemptive scheduling functions in `default_schedule.c` and register it as the member function of `default_sched_class`.

Modified codes:

ulib.h and ulib.c:

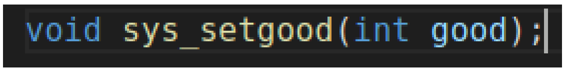


```
void set_good(int good);
```

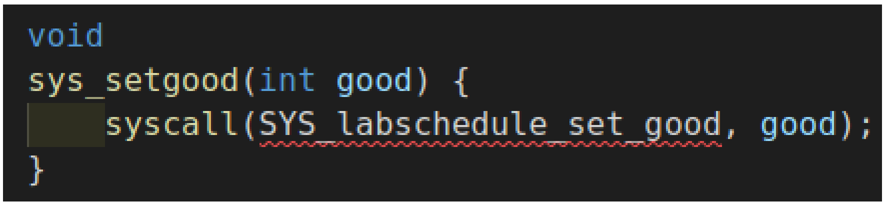


```
void
set_good(int good) {
    sys_setgood(good);
}
```

user/libs/syscall.h and syscall.c:

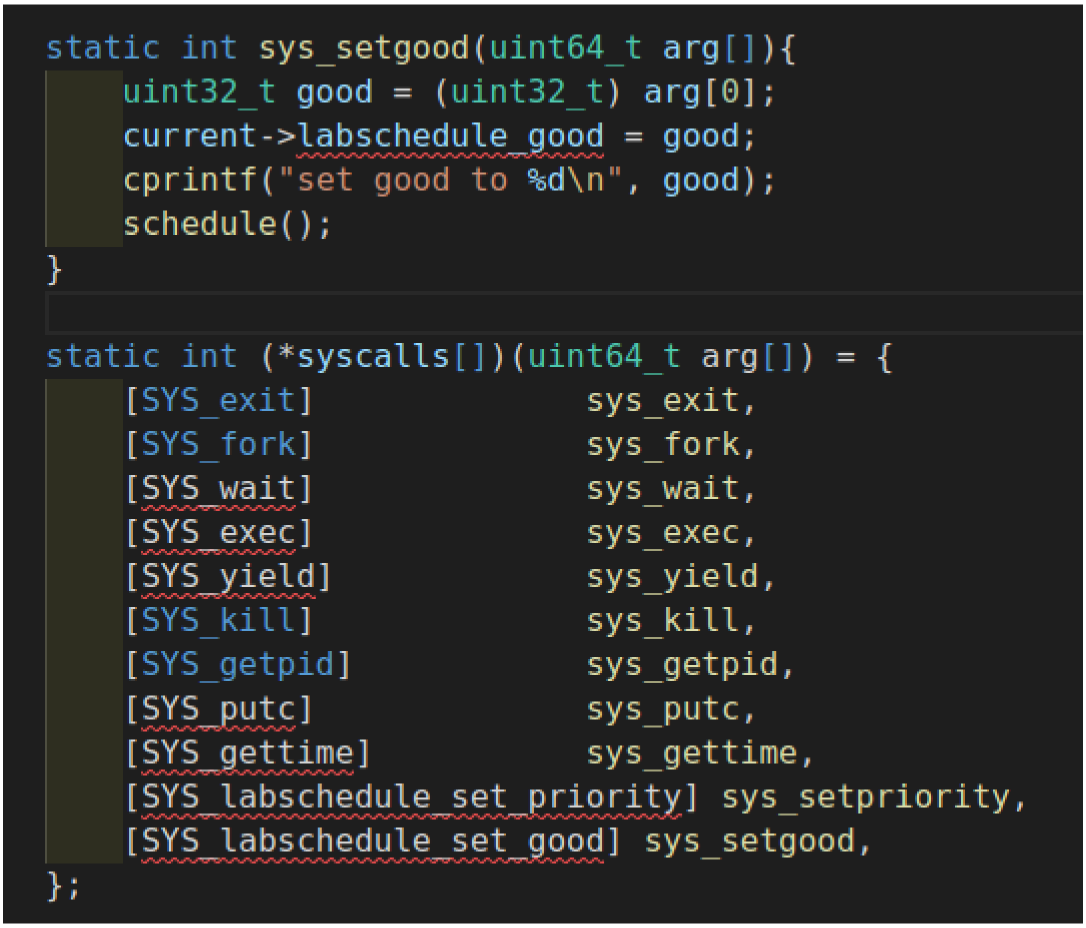


```
void sys_setgood(int good);
```



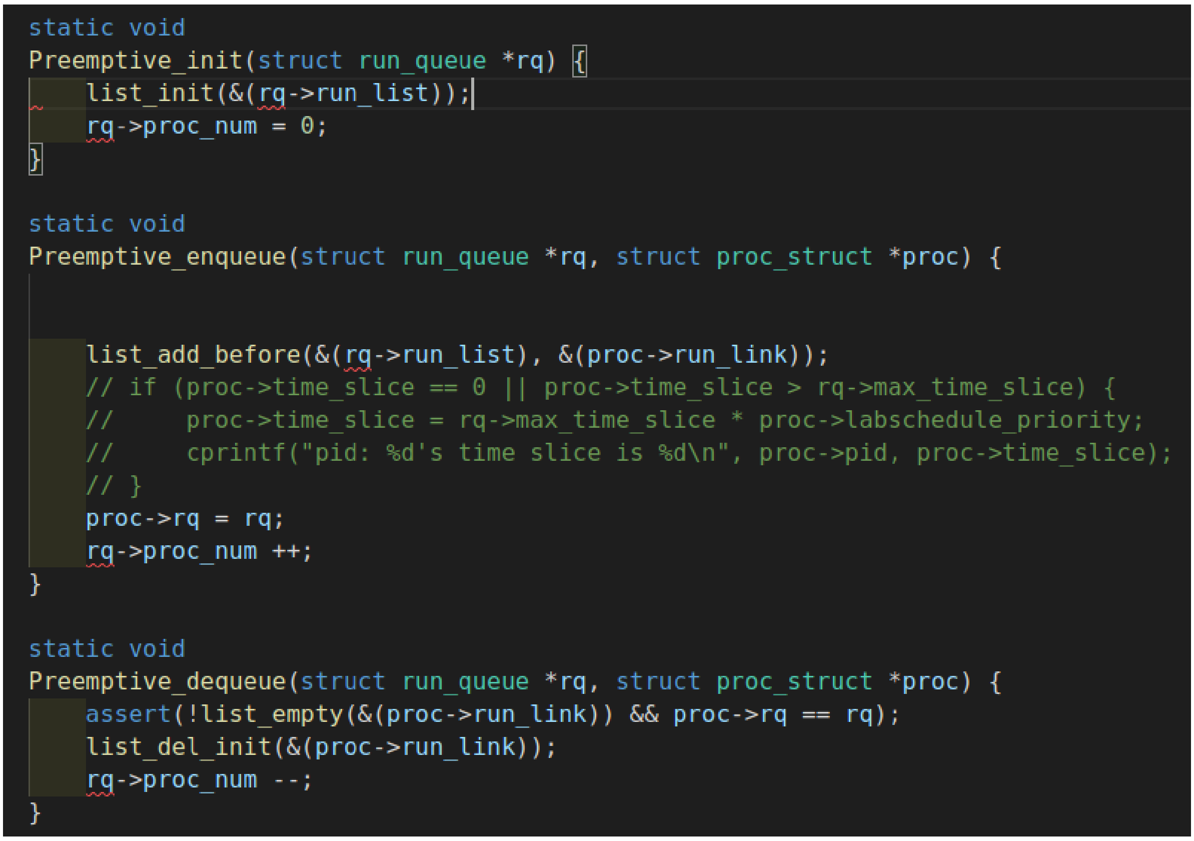
```
void
sys_setgood(int good) {
    syscall(SYS_labschedule_set_good, good);
}
```

kernel/syscall/syscall.c:

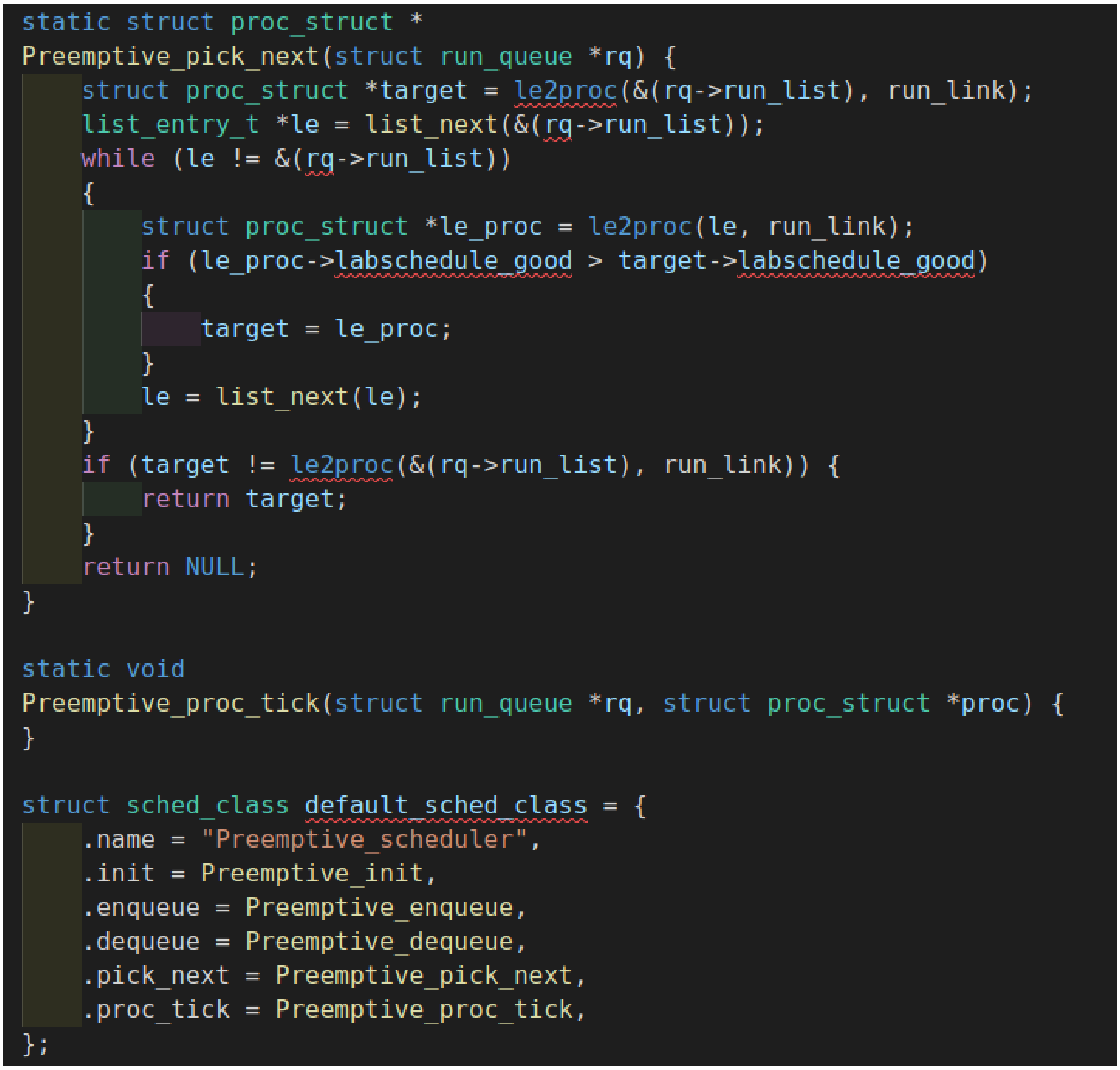


```c
static int sys_setgood(uint64_t arg[]){
    uint32_t good = (uint32_t) arg[0];
    current->labschedule_good = good;
    cprintf("set good to %d\n", good);
    schedule();
}

static int (*syscalls[])(uint64_t arg[]) = {
    [SYS_exit]              sys_exit,
    [SYS_fork]              sys_fork,
    [SYS_wait]              sys_wait,
    [SYS_exec]              sys_exec,
    [SYS_yield]             sys_yield,
    [SYS_kill]              sys_kill,
    [SYS_getpid]            sys_getpid,
    [SYS_putc]              sys_putc,
    [SYS_gettime]           sys_gettime,
    [SYS_labschedule_set_priority] sys_setpriority,
    [SYS_labschedule_set_good] sys_setgood,
};
```

default_sched.c:

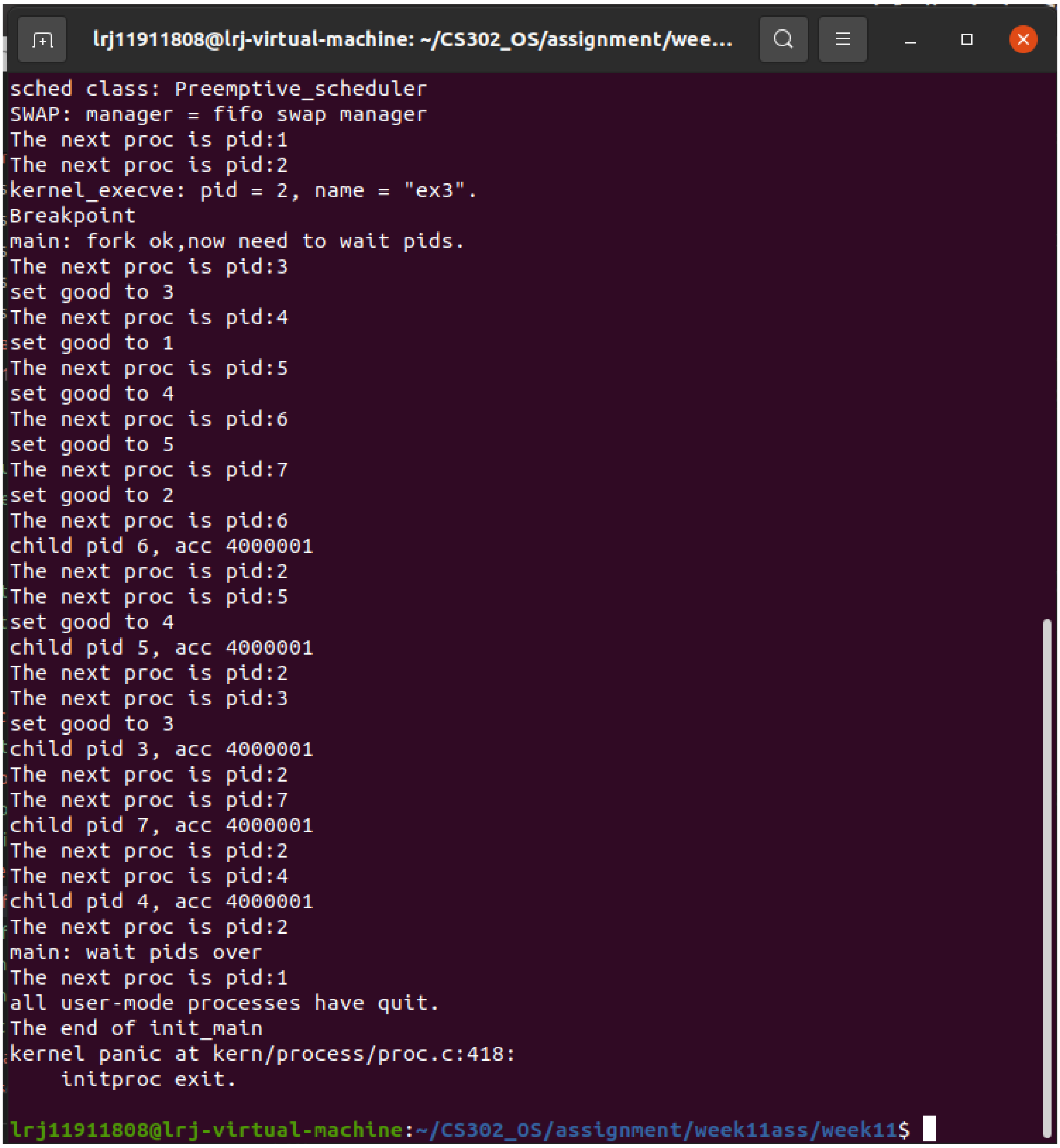


```c
static void
Preemptive_init(struct run_queue *rq) {
    list_init(&(rq->run_list));
    rq->proc_num = 0;
}

static void
Preemptive_enqueue(struct run_queue *rq, struct proc_struct *proc) {


    list_add_before(&(rq->run_list), &(proc->run_link));
    // if (proc->time_slice == 0 || proc->time_slice > rq->max_time_slice) {
    //     proc->time_slice = rq->max_time_slice * proc->labschedule_priority;
    //     cprintf("pid: %d's time slice is %d\n", proc->pid, proc->time_slice);
    // }
    proc->rq = rq;
    rq->proc_num ++;
}

static void
Preemptive_dequeue(struct run_queue *rq, struct proc_struct *proc) {
    assert(!list_empty(&(proc->run_link)) && proc->rq == rq);
    list_del_init(&(proc->run_link));
    rq->proc_num --;
}
```



```
static struct proc_struct *
Preemptive_pick_next(struct run_queue *rq) {
    struct proc_struct *target = le2proc(&(rq->run_list), run_link);
    list_entry_t *le = list_next(&(rq->run_list));
    while (le != &(rq->run_list))
    {
        struct proc_struct *le_proc = le2proc(le, run_link);
        if (le_proc->labschedule_good > target->labschedule_good)
        {
            target = le_proc;
        }
        le = list_next(le);
    }
    if (target != le2proc(&(rq->run_list), run_link)) {
        return target;
    }
    return NULL;
}

static void
Preemptive_proc_tick(struct run_queue *rq, struct proc_struct *proc) {
}

struct sched_class default_sched_class = {
    .name = "Preemptive_scheduler",
    .init = Preemptive_init,
    .enqueue = Preemptive_enqueue,
    .dequeue = Preemptive_dequeue,
    .pick_next = Preemptive_pick_next,
    .proc_tick = Preemptive_proc_tick,
};
```

Results:



lrj11911808@lrj-virtual-machine: ~/CS302_OS/assignment/wee...

```
sched class: Preemptive_scheduler
SWAP: manager = fifo swap manager
The next proc is pid:1
The next proc is pid:2
kernel_execve: pid = 2, name = "ex3".
Breakpoint
main: fork ok,now need to wait pids.
The next proc is pid:3
set good to 3
The next proc is pid:4
set good to 1
The next proc is pid:5
set good to 4
The next proc is pid:6
set good to 5
The next proc is pid:7
set good to 2
The next proc is pid:6
child pid 6, acc 4000001
The next proc is pid:2
The next proc is pid:5
set good to 4
child pid 5, acc 4000001
The next proc is pid:2
The next proc is pid:3
set good to 3
child pid 3, acc 4000001
The next proc is pid:2
The next proc is pid:7
child pid 7, acc 4000001
The next proc is pid:2
The next proc is pid:4
child pid 4, acc 4000001
The next proc is pid:2
main: wait pids over
The next proc is pid:1
all user-mode processes have quit.
The end of init_main
kernel panic at kern/process/proc.c:418:
    initproc exit.

lrj11911808@lrj-virtual-machine:~/CS302_OS/assignment/week11ass/week11$
```