Assignment 4

12011702 张镇涛

Q1

Process	Estimated CPU Cost	Arrives	Priority
A	4	1	1
B	1	2	2
C	3	5	3
D	2	4	4

The result is shown in followed table:

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+4+6+4)/4=4.5	(4+4+6+4)/4=4.5	(6+2+6+5)/4=4.75	(4+4+6+4)/4=4.5	(10+1+4+2)/4=4.25

Q2

design idea:

创建新的用户和内核的syscall, 实现用set_good()时，最终修改到进程的`labschedule_good`.

修改原有调度算法`RR_pick_next()`，遍历整个队列，寻找拥有最大`labschedule_good`的进程，并返回。

Running sequence of processes: 6->5->3->7->4

Code Modification:

先注释掉clock interrupt

```
    //clock_init();                  // init clock interrupt
    ints_enable();                   // enable irq interrupt
```

在kern/process/proc.c修改user_main，使其执行ex3

Open ~/oslab/Asg4/ex3/kern/process/proc.c

```
767 static int
768 user_main(void *arg) {
769 #ifdef TEST
770     KERNEL_EXECVE2(TEST, TESTSTART, TESTSIZE);
771 #else
772     KERNEL_EXECVE(ex3);
773 #endif
774     panic("user_main execve failed.\n");
775 }
776
```

观察ex3.c发现其调用set_good()方法，因此我们需要实现这个函数。

首先在libs\unistd.h中增加#define

Open ~/oslab/Asg4/ex3/libs/unistd.h

```
1 #ifndef __LIBS_UNISTD_H__
2 #define __LIBS_UNISTD_H__
3
4 #define T_SYSCALL           0x80
5
6 /* syscall number */
7 #define SYS_exit            1
8 #define SYS_fork            2
9 #define SYS_wait            3
10 #define SYS_exec           4
11 #define SYS_clone          5
12 #define SYS_yield          10
13 #define SYS_sleep          11
14 #define SYS_kill           12
15 #define SYS_gettime        17
16 #define SYS_getpid         18
17 #define SYS_brk            19
18 #define SYS_mmap           20
19 #define SYS_munmap         21
20 #define SYS_shmem          22
21 #define SYS_putc           30
22 #define SYS_pgdir          31
23 /*only for labschedule*/
24 #define SYS_labschedule_set_priority 255
25 #define SYS_labschedule_set_good 254
```

然后在kern/syscall/syscall.c实现sys_setgood()函数

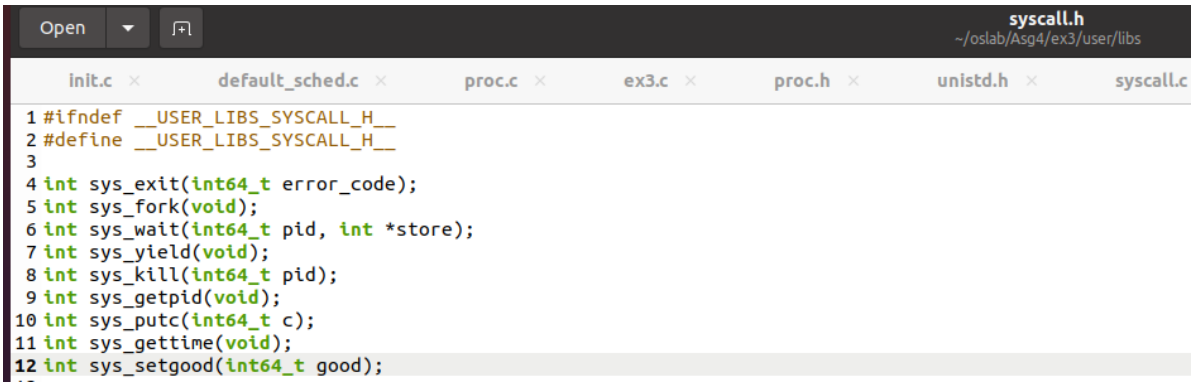
Open ~/oslab/Asg4/ex3/kern/syscall/syscall.c

```
66 static int sys_setgood(uint64_t arg[]){
67     current -> labschedule_good = arg[0];
68     schedule();
69     return arg[0];
70 }
71
72 static int (*syscalls[])(uint64_t arg[]) = {
73     [SYS_exit]              sys_exit,
74     [SYS_fork]              sys_fork,
75     [SYS_wait]              sys_wait,
76     [SYS_exec]              sys_exec,
77     [SYS_yield]             sys_yield,
78     [SYS_kill]              sys_kill,
79     [SYS_getpid]            sys_getpid,
80     [SYS_putc]              sys_putc,
81     [SYS_gettime]           sys_gettime,
82     [SYS_labschedule_set_good] sys_setgood,
83 };
84
```

在user下的syscall.c（user/libs/syscall.c）实现对于内核对应syscall调用：

```
int sys_setgood(int64_t good){
    return syscall(SYS_labschedule_set_good, good);
}
```

然后同样在相应的头文件syscall.h中增加函数声明：

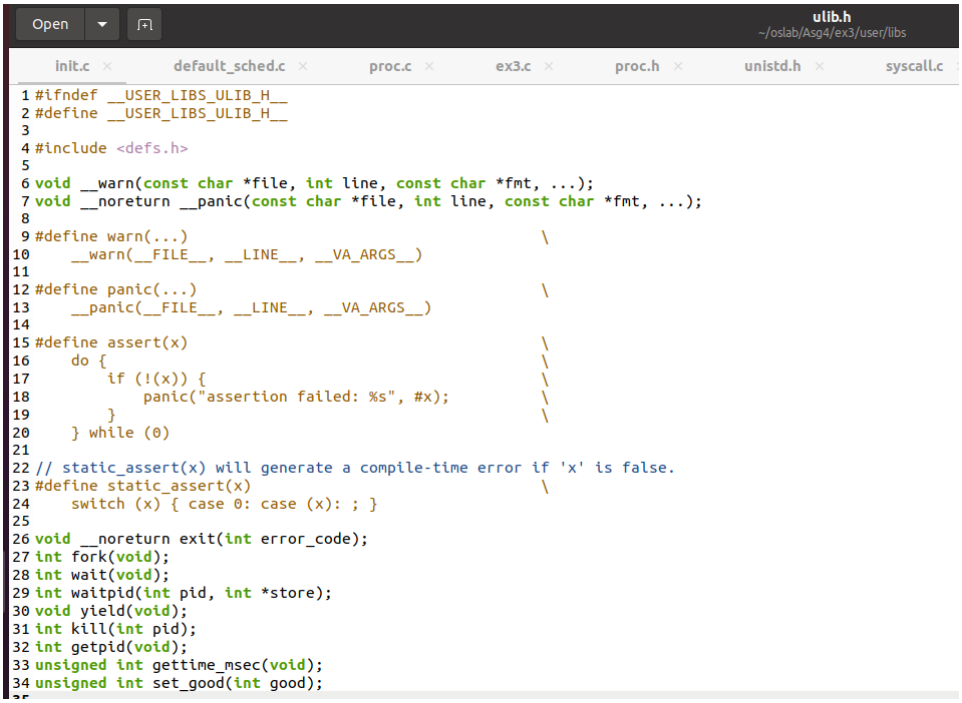


最后进行包装：

在user/libs/ulib.c增加set_good()方法

```
unsigned int set_good(int good){
    cprintf("set good to %d\n", good);
    return (unsigned int) sys_setgood(good);
}
```

同样在对应头文件增加声明：



至此已经实现set_good()。接下来需要实现调度算法：

Open

default_sched.c
~/oslab/Asg4/ex3/kern/schedule

init.c | default_sched.c | proc.c | ex3.c | proc.h | unistd.h | syscall.c

```
31
32 static struct proc_struct *
33 RR_pick_next(struct run_queue *rq) {
34     list_entry_t *le = list_next(&(rq->run_list));
35
36     list_entry_t * res = le;
37     int current_max = 0;
38     while (le != &(rq->run_list)) {
39         int tmp = le2proc(le, run_link) -> labschedule_good;
40         if (tmp > current_max){
41             current_max = tmp;
42             res = le;
43         }
44         le = list_next(le);
45     }
46     if (res != &(rq->run_list)) {
47         return le2proc(res, run_link);
48     }
49     return NULL;
50 }
51
```

代码运行结果：

```
sched class: RR_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:413:
    initproc exit.

os12011702@vmos-tony:~/oslab/Asg4/ex3$
```