CS302: Assignment4 Report

Name：陆荻芸 SID：12011537

1 Answers of Question 1

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

图 1: Schedule table of Q1

2 Answer of Question 2

2.1. Design Idea & Modified Codes

First define syscall number in `unistd.h`

```
/*only for labschedule*/
#define SYS_labschedule_set_priority 255
#define SYS_labschedule_set_good 254
```

图 2: Syscall number definition

Then implement the corresponding system call function. Define it in `syscall.h` and implement in `syscall.c`. In user program, good value is passed then `syscall` is called to pass the parameter to kernel. Then in kernel, argument is retrieved then call `labschedule_set_good` to set good value.

```
12    int sys_setgood(int good);
```

(a) Definition

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

(b) Implementation in user program

```
static int sys_setgood(uint64_t arg[]){
    int good = (int)arg[0];
    labschedule_set_good(good);
    return 0;
}
```

(c) Implementation in kernel program

图 3: sys_getgood

Register the syscall constant in syscall array.

```
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_good]      sys_setgood,
};
```

图 4: System call registration

Define `set_good` function in `ulib.h` and implement it in `ulib.c`. It is called in `ex3.c` which allows user program to invoke system call to set good value for a process.

```
34    int set_good(int good_value);
```

(a) Definition

```
int
set_good(int good_value){
    return sys_setgood(good_value);
}
```

(b) Implementation

图 5: set_good

In `proc.h` and `proc.c` to enable function `labschedule_set_good`. It is the core function to set a process' good value. After the change of good value, processes need to be reschduled, thus `do_yield` is called.

```c
void labschedule_set_good(uint32_t good);
```

(a) Definition

```c
void labschedule_set_good(uint32_t good){
    current->labschedule_good=good;
    cprintf("set good to %d\n",good);
    do_yield();
}
```

(b) Implementation

图 6: labschedule_set_good

The schedule scheme based on good value schedules the process with maximum good value to preempt current schedule. Thus we maintain a integer value and a pointer to indicate the current maximum good value and the corresponding process. Iterate through the queue each time to update the current maximum good value process. Assign the pick next scheme with `Good_pick_next` instead of RR.

```c
static struct proc_struct *
Good_pick_next(struct run_queue *rq)
{
    list_entry_t *le = list_next(&(rq->run_list));
    struct proc_struct *cur_max = le2proc(le,run_link);
    int max_good = cur_max->labschedule_good;
    while(le != &(rq->run_list)){
        struct proc_struct *p = le2proc(le, run_link);
        int cur_good=p->labschedule_good;
        if(cur_good>max_good){
            max_good=cur_good;
            cur_max=p;
        }
        //if good value equals then use fifo->following order of queue
        le = list_next(le);
    }
    return cur_max;

}
```

(a) Schedule scheme

```c
struct sched_class default_sched_class = {
    .name = "RR_scheduler",
    .init = RR_init,
    .enqueue = RR_enqueue,
    .dequeue = RR_dequeue,
    .pick_next = Good_pick_next,
    .proc_tick = RR_proc_tick,
};
```

(b) Assign scheme

图 7: Schedule

Finally, annotate the `clock_init()` in `init.c` to unable clock interrupt, and let `user_main` to run `ex3` instead of `rr` to check our implementation.

2.2. Running Result & Analysis

The result is shown in figure 8. First, after all processes were forked, program went to wait process 3. Then it will spin till the good value of process 3 was modified. After modification, the process is

rescheduled. Since the rest of the processes in the queue have the default good value 6, so the next scheduled process will be 4 now. Repeat this after all processes good value were reassigned. Now the cpu will continue to schedule the running order with their carious good value. Process 6 has the highest, it will be scheduled to run after it finished. After 6 was finished, 2 will be awaken. 2 will then recycle the dead 6 and schedule 5 to run. Process 5 then was dequeued and ran. Repeat this after all processes were finished according to the ranking of the good value.

```
OS is loading ...

memory management: default_pmm_manager
physcial memory map:
  memory: 0x08800000, [0x80200000, 0x885fffff].
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:420:
    initproc exit.

(base) ldy12011537@ludiyun-ROG:~/Desktop/OSlab/As4/ex3$
```

图 8: qemu result