Week11

EX0. CPU Scheduling

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

Time	HRRN	FIFO/FCFS	RR	SJF	Priority
1	A	A	A	A	A
2	A	A	B	A	A
3	A	A	A	A	A
4	A	A	D	A	A
5	B	B	C	B	B
6	D	D	A	D	C
7	D	D	D	D	C
8	C	C	C	C	C
9	C	C	A	C	D
10	C	C	C	C	D
Avg. Turn-around Time	(4+4+4+8) /4 = 5	(4+4+4+8) /4 = 5	(9+1+4+8) /4 =5.5	(4+4+4+8) /4 = 5	(4+4+4+7)/4 = 3.75

EX1

The design id is quit simple, just follow the way to do the syscall and finally change the attribute priority.

```
static int
user_main(void *arg) {
#ifdef TEST
    KERNEL_EXECVE2(TEST, TESTSTART, TESTSIZE);
#else
    KERNEL_EXECVE(ex1);
#endif
    panic("user_main execve failed.\n");
}
```

Change to run ex1

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

```
static int
sys_setpriority(uint64_t arg[]) {
    int priority = (int)arg[0];
    set_priority(priority);
    return 0;
}

static int sys_gettime(uint64_t arg[]){
    return (int)ticks*10;
}

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,
};
```

Add syscall

```
int sys_setpriority(int priority){

    return syscall(SYS_labschedule_set_priority,priority);
}
```

change priority

```
set_priority(int priority)
{   cprintf("set priority to %d\n",priority);
    current->labschedule_priority = priority;
}
```

The change of .h file is trivial and I think it is not necessary to show.

```
OpenSBI v0.6
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|____/_____|
        | |
        |_|

Platform Name          : QEMU Virt Machine
Platform HART Features : RV64ACDFIMSU
Platform Max HARTs     : 8
Current Hart           : 0
Firmware Base          : 0x80000000
Firmware Size          : 120 KB
Runtime SBI Version    : 0.2

MIDELEG : 0x0000000000000222
MEDELEG : 0x000000000000b109
PMP0    : 0x0000000080000000-0x000000008001ffff (A)
PMP1    : 0x0000000000000000-0xffffffffffffffff (A,R,W,X)
OS is loading ...

memory management: default_pmm_manager
physcial memory map:
  memory: 0x08800000, [0x80200000, 0x885fffff].
sched class: RR_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:413:
    initproc exit.

wangxiequn11910405@ubuntu-linux-20-04-desktop:~/Desktop/Parallels Shared Folders
/Home/CLionProjects/week11$
```

EX2

```
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 10010
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
```

```c
    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);
    }
```

EX3

```
kernel_execve: pid = 2, name = "ex3".
Breakpoint
main: fork ok,now need to wait pids.
The next proc is pid:7
set good to 2
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
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.
```

```c
set_good(int good)
{   cprintf("set good to %d\n",good);
    current->labschedule_good = good;

}
```

```c
void
set_good(int good)
{   cprintf("set good to %d\n",good);
    current->labschedule_good = good;

}
```

```c
static struct proc_struct *
alloc_proc(void) {
    struct proc_struct *proc = kmalloc(sizeof(struct proc_struct));
    if (proc != NULL) {
        proc->state = PROC_UNINIT;
        proc->pid = -1;
        proc->runs = 0;
        proc->kstack = 0;
        proc->need_resched = 0;
        proc->parent = NULL;
        proc->mm = NULL;
        memset(&(proc->context), 0, sizeof(struct context));
        proc->tf = NULL;
        proc->cr3 = boot_cr3;
        proc->flags = 0;
        memset(proc->name, 0, PROC_NAME_LEN);
        proc->wait_state = 0;
        proc->cptr = proc->optr = proc->yptr = NULL;
        proc->time_slice = 0;
        proc->labschedule_priority = 1;
        proc->labschedule_good = 6;
    }
```

```c
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
};
```

```
#define SYS_labschedule_set_priority 255
#define SYS_labschedule_set_good 256
/* SYS_fork flags */
```

The good is like a priority to set. For init, set to 6

We close the clock and change pick function.

```
static struct proc_struct *
RR_pick_next(struct run_queue *rq) {
    list_entry_t *le = list_next(&(rq->run_list));
    list_entry_t *cu = list_prev(le);
    struct proc_struct * tp = le2proc(le, run_link);
    struct proc_struct * cp = le2proc(cu, run_link);
    struct proc_struct * max_p = cp;

    while (tp!=cp){
        if(max_p->labschedule_good<tp->labschedule_good){
            max_p = tp;

        }
        le = list_next(le);
        tp = le2proc(le, run_link);
    }


    return max_p;
}
```

For every pick, choose the biggest value of good.