**SUSTech CS302 OS Lab4 Report**

Title: **SUSTech CS302 OS Lab4 Report**

Student: Name: \_赵诣\_, ID: \_11612917\_

Time: 2019 Year 3 Month 23 Day

Experimental Environment: linux ubuntu 18.04

Deadline: **11:59, 2018-03-27**

Summit by: Blackboard

Task:

Task 1. Create a directory named with your studentID and begin the following task.

Task 2. Find out all bugs in source codes

\_\_Task 3. Modify the source code according to the requirements of the experimental

\_\_instructions, compile and run each program

Task 4. Observe the results and finish the questions

Experiments:

1. Fundamental:

* What is a process? What is a program? And what is the difference?

Process: an execution of certain program with its own address space.

Program: a group of instructions to carry out a specified task.

The difference is that the program is essentially instructions stored in files, but process is actually the execution of a program (with address space).

* What is job? And what is the difference between process and job?

Job: a series of program submitted to operating system for some goals.

The difference is that a job is a set of program, but process is actually the

execution of a program (with address space), a job may include a set of processes.

* What are the differences between job scheduling and process scheduling?

The difference is that job scheduling selects a mix of processes, controlling the degree of multi-programming, while process scheduling selects one process from the ready queue.

* What are the similarities and differences among structures, classes, and unions in C language：

Similarities: they are all composite data type that define a physically grouped list of variables to be placed under one name in the block of memory.

Differences:

1. structures and classes has all the members in memory at the same time, but unions only allow one member to be put in memory at one time.

2. unions and structures has their members designed as public by default, but classes has their members designed as private by default.

2. Knowledge:

* How many states are in a job? And what are they?

There are three states, including READY, RUNNING, DONE.

* What programs are used in this experiment for job control?

And their function?

enq: Send a enque request to the scheduler and submit the job for running. select->next=NULL;

deq: Send a deque request to the scheduler.

stat: Output information about the currently running job and all the jobs in the waiting queue.

* What is used for inter-process communication in this experiment?

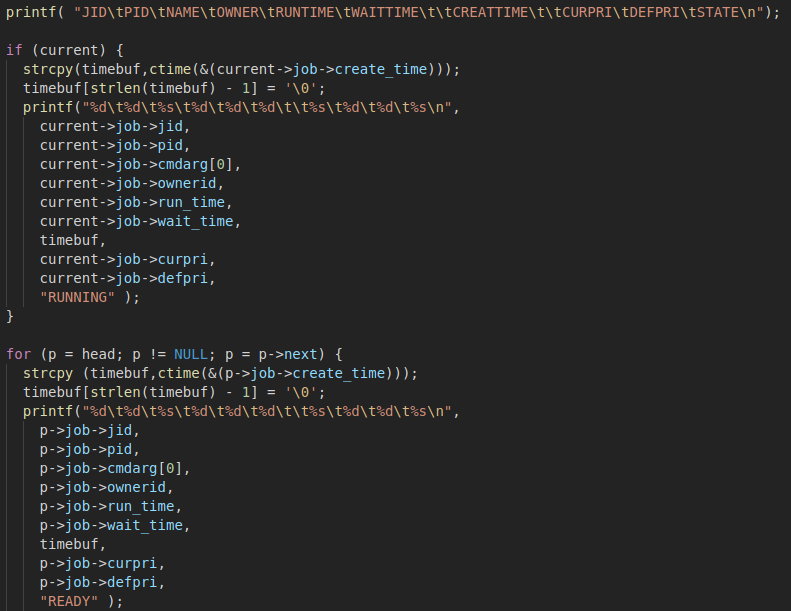
And its function? Named pipe(FIFO)

Give out a way that deq, enq, stat are able to communicate with the scheduler, the former ones write the request in the FIFO file and scheduler read the request in the FIFO file.

3. Questions:

* What should be noted when printing out the job name:

The job name is placed in cmdarg[0], use %s to print it.



* Submit a job that takes more than 100 milliseconds to run:

#include <stdio.h>

#include <unistd.h>

int main(){

for(int i=0;i<10;i++){

printf(“iteration %d\n, pid: %d\n, i , getpid());

}

return 0;

}

* List the bugs you found and your fix (code and screenshot)

\_1. the head pointer of the ready queue is wrongly assigned. \_

\_2. there are rings if not assign select’s next pointer to NULL after select is chosen to be next. (both in do\_deq() and job\_select())

// 选中head

    if (selectprev == select) {

      head = head -> next;

    // 选中中间

    }else{

      selectprev->next = select->next;

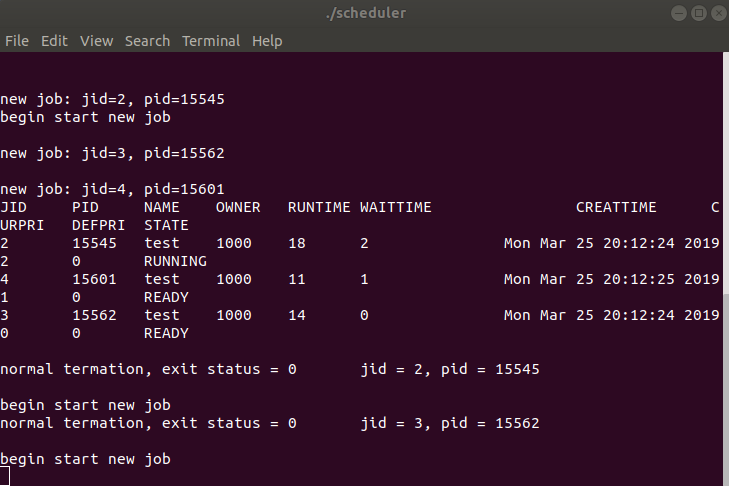
    }

    if(select){

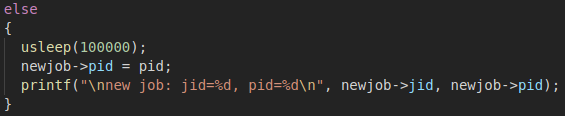
      select->next = NULL;

    }

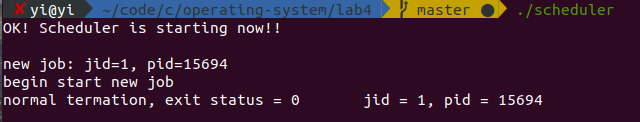
After bug fix: <program run as desired.>



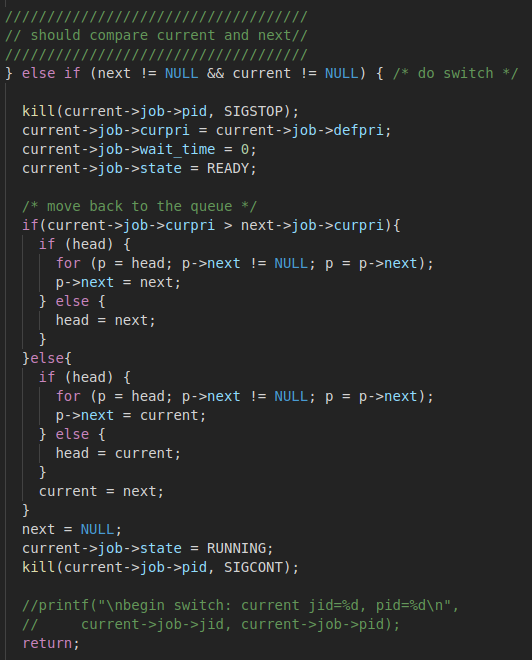
3. the first program last as T (traced/stop) state forever if no new job added.



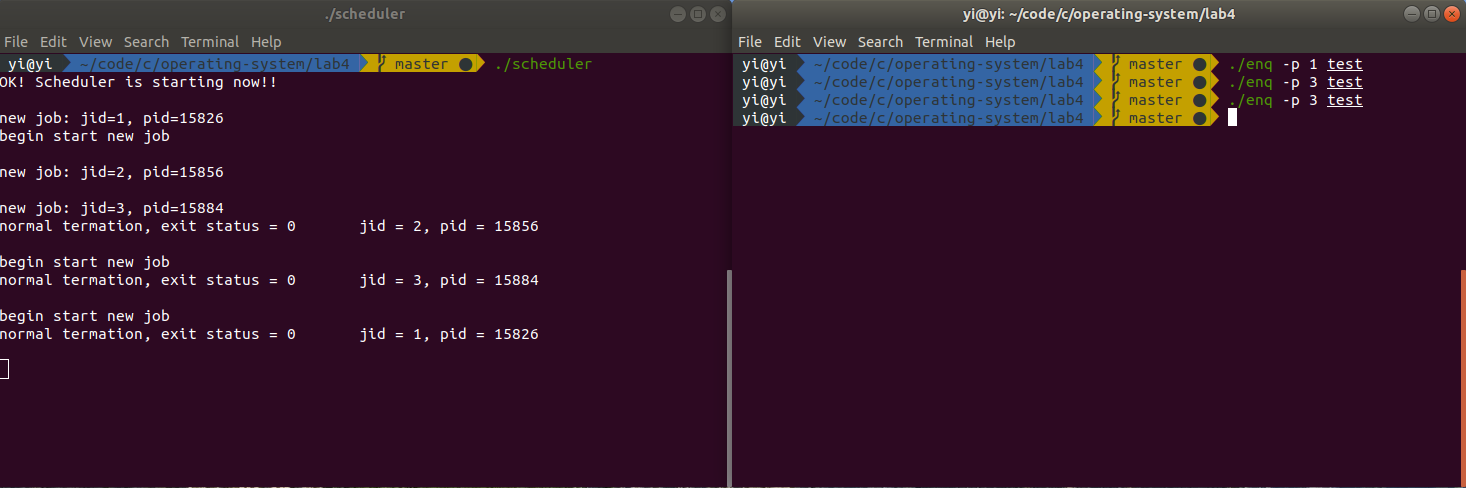
After bug fix: <run and terminate.>



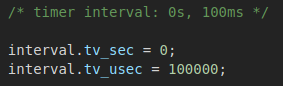
4. the priority selecting method ignores the current job



After bug fix: <current job can also be selected>



5. the timeslice is not 100 milliseconds as the interval.tv\_usec set microseconds as the unit.

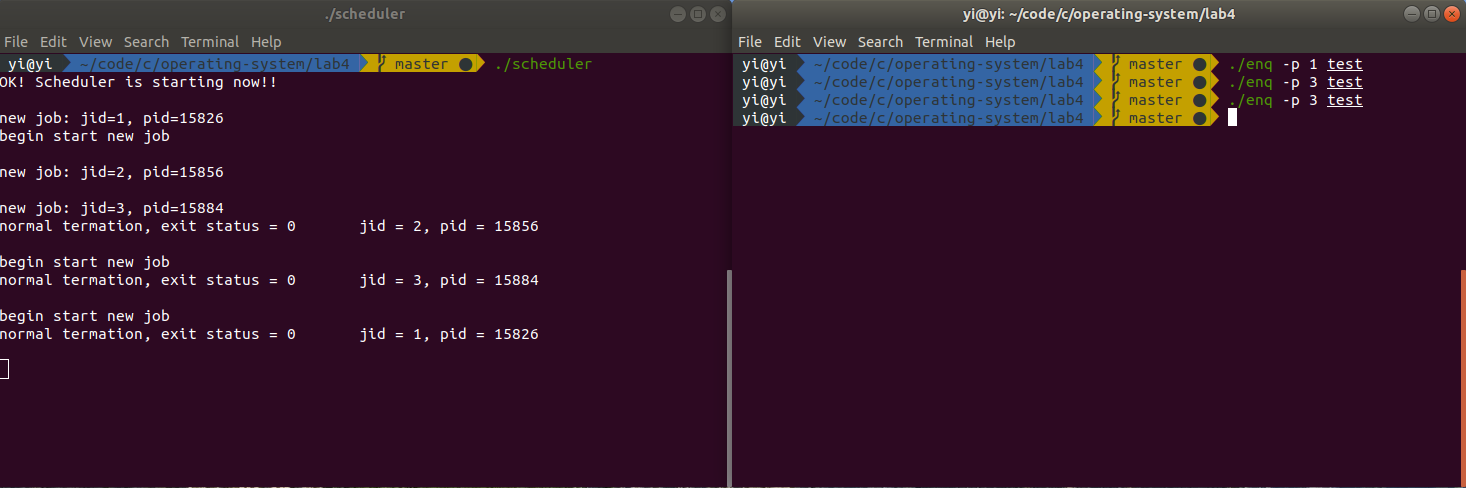


After bug fix: <timeslice long as desired>

as the picture in 4

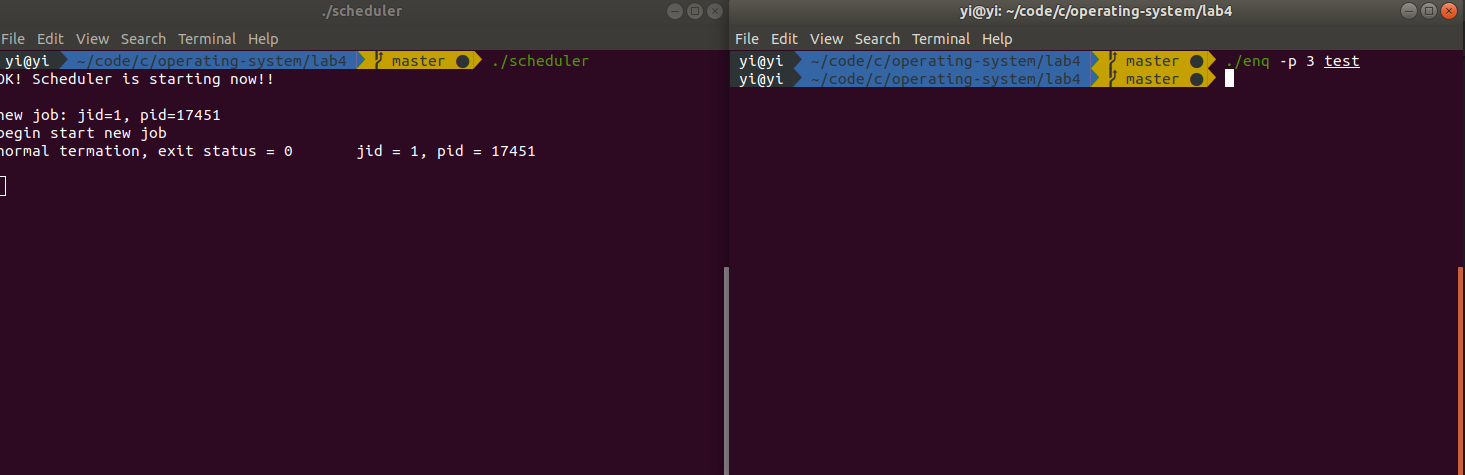
* Run the job scheduler program and analyze the execution of the submitted job:

The job scheduler construct a READY queue, jobs submitted are put in the queue, every time slice, the scheduler iterate all the jobs (iterate the jobs in the queue, then compare the select one from the queue with the current job), get the job with the highest priority and let it run. Job in the READY queue will add one priority after a time slice (the priority will not exceed 3). The running job will return to the default priority after running and clear the waiting time.

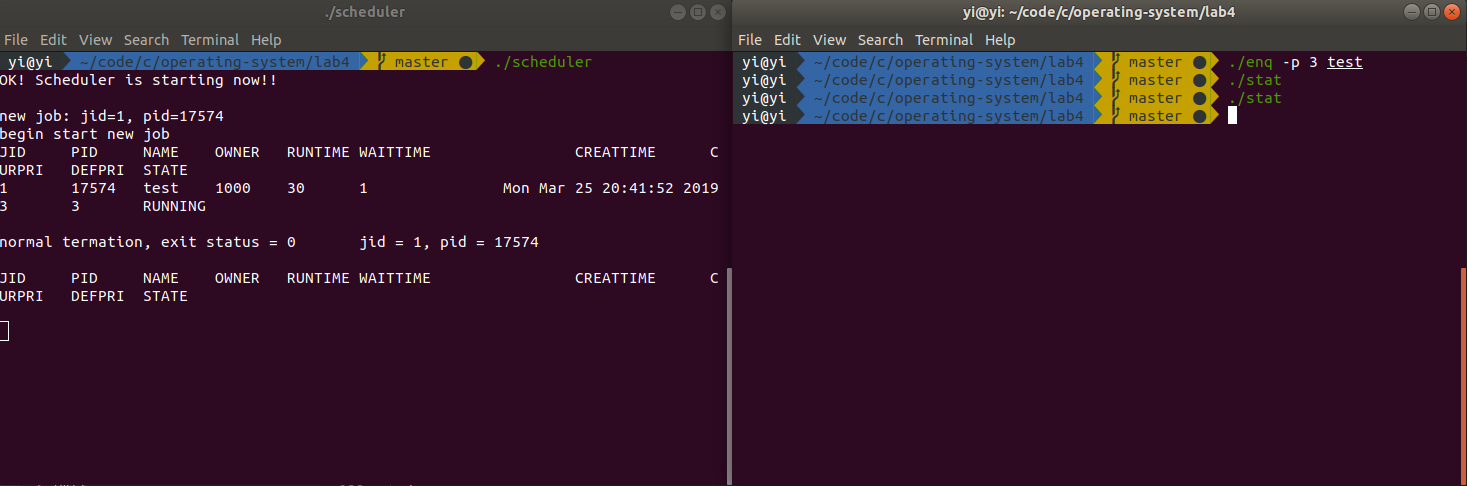


* Understand the process of job scheduling: (Execution results and corresponding code)

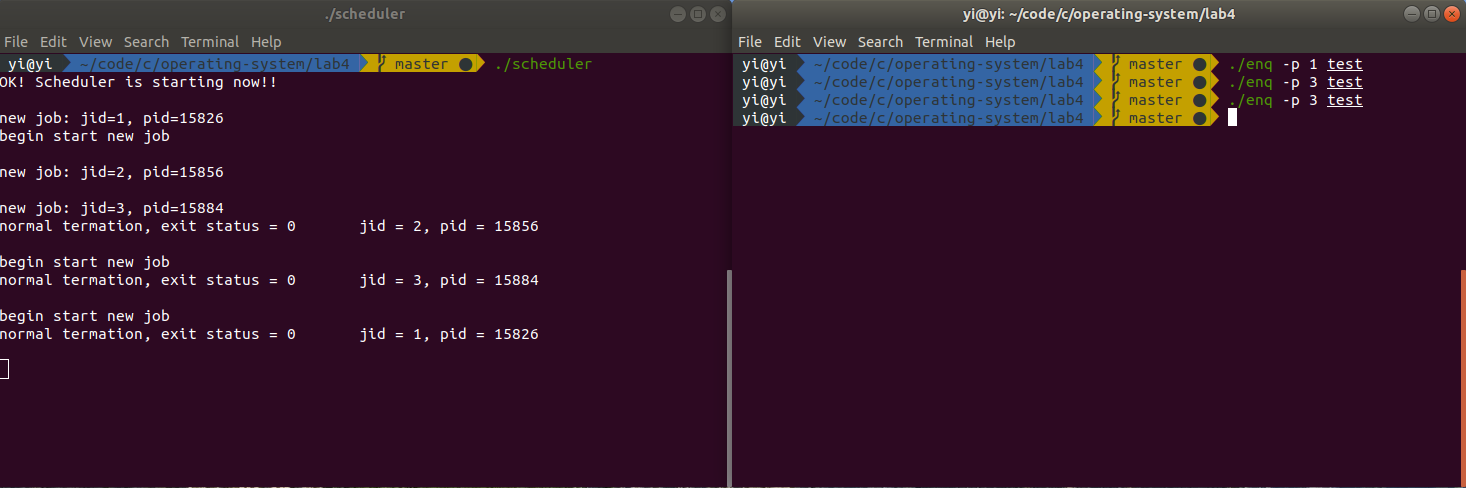
Submit a new job： The do\_enq() will create a new process for it temporarily STOP the process and put the job’s state to be READY, then it will be put in the READY queue.



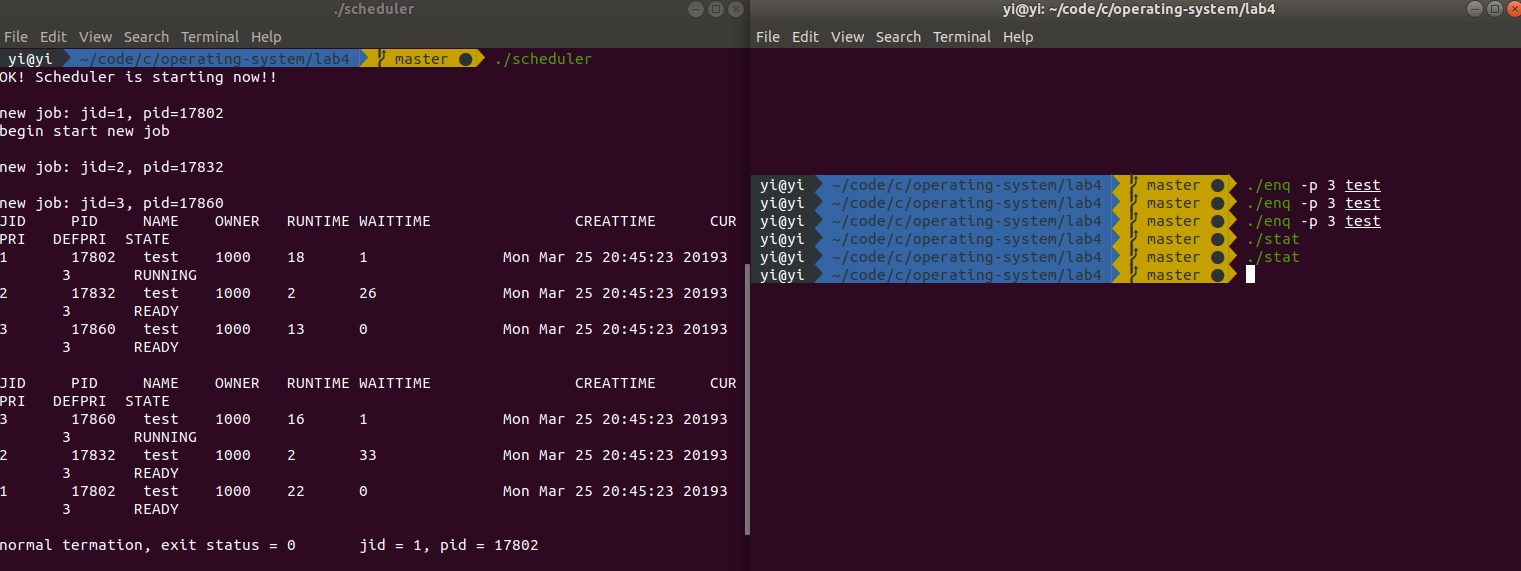
End of job execution: The job will be deleted form the queue.



Job scheduling due to Priority: Iterating all the jobs (iterate the jobs in the queue, then compare the select one from the queue with the current job), get the job with the highest priority and let it run. The priority sets curpri as the first comparing standard and the wait\_time as the second comparing standard, both of which means a higher priority when has a larger value.



Job scheduling due to time slice: The time slice is defined as 100 milliseconds, the running job will continuously run in the timeslice, after the timeslice, job will be rescheduled by the scheduler.



Conclusion:

Through this lab, I get to get a deep understanding of job scheduling. The scheduling divide job into three part, current, next, and the queue. Job must first be selected as next, then it will be put to run in the next timeslice. And the scheduling process is almost all about priority and timeslice.

Submission(directly compress the following files, no more directory):

**-lab4\_report\_studentID.pdf (pdf version report)**

**-scheduler.c (modified code .c file)**