Project 1

Name : Lin, Ting Feng

Student ID : B07902141

Date : April 28, 2020

1 Design

1.1 CPU scheduling

The scheduler runs on CPU 0, and all other processes run on CPU 1.

1.2 System calls

- **sys_gettime** is used to get the current time. The information is store in a timespec pointer.
- **sys_printinfo** is used to print the message to the kernel.

1.3 Details

1.3.1 util

Several important functions are defined in **util.h**

- **SetCPU** sets affinity CPU of each process to a certain CPU.
- **Setpri** sets the scheduler type(SCHED_FIFO) and the priority of processes.
- **DoProcess** forks the child process when the process is ready, and set affinity CPU of child process to CPU 1.

1.3.2 main

Read the input, sort the process by ready time, and set the affinity CPU of scheduler to CPU 0, then call the scheduling function to deal with the problem.

1.3.3 FIFO

Since the processes are sorted by ready time, we just need to deal with it in order.

1.3.4 RR

We maintain a queue with two pointer. The performing process is always at the front of queue. When a process runs for 500 unit times, we remove it from the front of queue and add it to the end if it hasn't finished yet.

1.3.5 heap

For the rest two scheduling algorithm, we need to maintain a heap, with the following operation.

- **Insert**: Insert a process.
- **Delete**: Pop the "smallest" process.

The smallest process means it has the least remaining processing time. Since heap is a complete binary tree, we just use an array to store it. The time complextiy is $\mathcal{O}(\log n)$ for each operation.

1.3.6 SJF

When there are no processed running, we will see whether the heap is empty. If it is not empty, we pop the smallest process in the heap and run it.

1.3.7 PSJF

Basically the method is the same as SJF, except that every second we need to check whether the smallest process in the heap has the smaller running time than that of current running process, and whether we need to swap them.

2 Kernel version

4.15.0 on Ubuntu 18.04.4 LTS

3 Comparison

We test theorectical results and experimental results by directly looking at the total times when the last process ends with. The value of one unit time will be tested in every testdata. But the value is closed to 0.0017.

theory(s)	1	2	3	4	5
FIFO	4.183339	145.615819	38.596712	5.353536	38.460350
RR	4.184356	16.015411	52.440830	38.431083	38.430580
SJF	23.406431	25.735186	53.683091	18.384835	5.861388
PSJF	41.728220	18.355776	5.855952	23.410317	25.722605

real(s)	1	2	3	4	5
FIFO	4.213839	145.410999	39.518722	6.434021	38.479627
RR	4.192322	16.059127	52.288276	38.766884	42.022623
SJF	23.516220	25.694771	53.816027	18.573791	5.878285
PSJF	41.665333	18.781232	6.463633	23.403948	25.791939

We can see that the results are closed but most of real time are slightly larger than theory time. There are several possible reasons.

- Other unknown processes might use the CPU when testing, which causes the larger real time.
- The scheduler does lots of tasks with system calls, such as forking the child processes, setting the processes' priority, and setting the CPU every process should run on. All of this tasks causes time.
- The UNIT_TIME we estimate might be very precise, it also leads some deviation.