

# Assignment 4

---

Please complete a report and upload the corresponding codes.

The files should be uploaded directly without compression **without compression without compression without compression**

The files to be submitted for this assignment are:

1. report.pdf
2. ex3.zip

## 1. CPU Scheduling [50pts]

---

Consider the following single-threaded processes, and their arrival times, estimated CPU costs and their priorities (a process with a higher priority number has priority over a process with lower priority number):

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

Please note:

- Ignore context switching overhead.
- If a process arrives at time  $x$ , they are ready to run at the beginning of time  $x$ .
- Highest response ratio next (HRRN) is a non-preemptive scheduling algorithm. In HRRN, the next job is not that with the shortest estimated run time, but that with the highest response ratio defined as:  $1 + \text{waiting time} / \text{estimated CPU time}$ .
- Newly arrived processes are scheduled last for RR. When the RR quanta expires, the currently running thread is added at the end of the ready list before any newly arriving threads.
- The quanta for RR is 1 unit of time.
- Average turn-around time is the average time a process takes to complete after it arrives.
- SJF is non\_preemptive.
- Priority scheduler is preemptive.

Given the above information please fill in the following table.

Time	HRRN	FIFO/FCFS	RR	SJF	Priority
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
Avg. Turn-around Time					

## 2. Preemptive process scheduling [50pts]

In this problem, we will implement preemptive scheduling.

**Please enable the clock interrupt first.**

In PCB, we add another integer name `tab_scheduled_good` and the initial number is 6. **You need to implement a syscall to let user process set `tab_scheduled_good`.**

When the child process runs for a while, it may call the syscall to modify `tab_scheduled_good`.

In this scheduling algorithm, we need to choose the process with the largest good value to run when scheduling.

That is to say, when a process call syscall to set its `tab_scheduled_good` smaller, it will be preempted by a runnable process with a larger `tab_scheduled_good`.

If processes have same `tab_scheduled_good`, they are scheduled by fifo.

You need to realize this scheduling algorithm by modify `default_sche.c`.

To test your code, you need to run user program `ex3`. Please release annotations in the main function in `ex3.c`. Other contents are prohibited to modify in `ex3.c`. And you should let `user_main` to run `ex3` instead of `rr`.

Please show your design idea, the running sequence of processes, modified code(screen-shot) and the running result(screen-shot) in your report.

Sample:

```
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:414:  
initproc exit.
```