Operating System Project1 Report

B07902031 資工二 黃永雯

1 設計

Overall

四個scheduling method的差異僅在於選取下一個要執行的process的方式，因此我會先在overall說明整體的架構，下方再說明四個分別選擇next process的方式。

1. I first sort all the processes by its ready time, if the ready time of two processes are the same, I use its original ID to sort.

2. Assign the main process (use for scheduling) to CPU 0.

3. Use a while loop to check whether all the processes finishing running. In the while loop, the checking will be as follows:

(1) If there is a process running, check if it finishes (execution time = 0). If it finishes, print its name and pid, and add one to finish number, set the running index to -1 (meaning no process is running).

(2) Then check if the running time of the processes that have not been execute meet the current time, if meeting the current time, then fork a child process and execute, block the processes (set the priority low).

(3) Assign the next process (the method will be described individually below).

(4) Check if the next process is not the current running process, if different (which means that context switch happens), then wake the next process (set the priority

high) and block the running process (set the priority low).

(5) If there is a process running, its execution time minus 1, and if there are any processes haven't finished, the while loop will continue.

FIFO

1. FIFO means that the process comes first will run first, and it will run until it finishes. Therefore, I first check that if there is a process running, then next process will be the current running process.

2. Then if there is no process running, I will check that if there are processes that have been executed but have not run yet, then next process will be the first of them. If not, then next will be -1.

RR

1. I use a queue to implement RR scheduling. All the processes that haven't finished running will be in the queue.

2. If there is no current running process and the queue is not empty, the next process will be the first one in the queue.

3. If there is a process running, then check whether it reaches the time quantum, if it reaches the time quantum, then push it to the back of the queue and pop it from the front of the queue.

4. If it hasn't reach the time quantum, then the next process will be the current running process.

SJF

1. SJF means that the process with the smallest execution time will run first.

2. If there is any process running, the next process will be the current running process.

2. Find the process that is ready and with the shortest execution time to be the next process.

PSJF

1. Similar to SJF; however, step2 of SJF is unneeded.

2. Find the process that is ready and with the shortest execution time to be the next process.

2　核心版本

使用的核心版本為4.14.25

3　實際結果與理論結果比較

以下為實際結果與理論結果的比較，每種scheduling method僅以一個範例表示，其中optimal指的是理論值，real指的是實際值，大於1代表理論值跑得比較快，小於1則代表實際值跑的比較快。

FIFO

```
optimal is 11.527082796 seconds, real is 11.203002397 seconds, ratio = 0.97
1885307
```

RR

```
optimal is 25.052193277 seconds, real is 24.226665107 seconds, ratio = 0.96
7047669
```

SJF

```
optimal is 39.499470382 seconds, real is 40.500849043 seconds, ratio = 1.02
5351698
```

PSJF

```
optimal is 78.384163015 seconds, real is 81.005976854 seconds, ratio = 1.03
3448260
```

以下分別解釋理論值比實際值快以及實際值比理論值快的原因：

理論值比實際值快

This happens because for real-time scheduling, there exists context switch between processes and processes or between processes and scheduler.

實際值比理論值快

This happens because the time unit may be smaller than the time of context switch or child process creation.