1.設計

A. System call :

1.get_time : using getnstimeofday() to get the start time and using copy_to_user to write the time into user space buffer.

2.prinf_ans : similar to get_time but adding printk to write 2 time_string to dmesg.

B. main :

1.pcb structure

I use circular link list to maintain the schedule struct which store the process status of fork process.

2. priority setting :

a. for scheduler(parent) :

except idle : set its priority as 10 (2nd order)

idle : set its priority as 1 (3rd order) , only happen with no job in queue

b. for child process unselected : set its priority as 1 (3rd order)

c. for child process selected (only one in same time): set its priority as 99 (1st order)

3. scheduler process : follow is a while loop and finish when all task finish

a. if next job don't ready and no task in queue => idle until timer become ready time

b. check if the are some task ending and remove it from the queue

c. create tasks(fork) whose ready time < timer (using a while loop) , and insert it to the queue (pending in the tail) , at the same time I would set child process's priority as 1

d. base on the queue , select the task and set its priority as 99

FIFO : the head of the queue

RR : head->next of the queue , and let head = head->next

SJF / PSJF: the shortest exec_time in the queue

-------child process which been selected run---------

back to schedule when :

FIFO / SJF : finish

RR : finish or running time = 500

PSJF : timer = next job's ready time

---

e. (only PSJF) renew the exec_time depend on the running time it run

2.核心版本

linux 4.14.25

3.比較實際結果與理論結果，並解釋造成差異的原因

在順序上，結果與理論相同，但在運行時間上，因為 context swicth 的 overhead ，導致部分 TASK 時間並沒有明顯的節省 ex : PSJF 理論上時間要小於 SJF，但部分 TASK 無顯著差異