2020 OS Project 1 Report

B07902066 資工二 黃禹喆

1. Design:

   There are mainly 2 parts in `main.c`:

   (1) function `scheduler()`: This function is mainly a while loop which keep scheduling child processes. It first sorting the child process according to their ready times. Before entering the while loop, the function assigns the main process itself to CPU 0 to avoid affecting the child processes. In addition, the function creates an extra child process with medium priority (50) and assigns it to CPU 1 (used by all child process) to avoid the situation that when one running process terminates, other child processes which want to run but with low priorities will run. In every round (approximately 1 unit of time) of the while loop, the function first check whether the running process (if exists) finished execution. If yes, call `waitpid()` to wait the process. And then, for those child processes meet their ready times, fork to create them and assign them to CPU 1. Note that since there is an extra child process with medium priority, the child processes after fork with low priorities can not get CPU resource. Then, the function picks next process to run by calling `select_next()`. If the selected process is already running, then do nothing. Otherwise, block the running process by lowering its priority and wake the selected process up by increasing its priority. In the end of this round, the function runs for one unit of time and starts a new round.

   (2) function `select_next()`: This function selects next process allowed to run according to the designated policy:

      a. FIFO: Since FIFO is non-preemptive, if there is a process running, let it keep running and return its id. Otherwise, select one process with the earliest ready time among processes that haven't run.

      b. RR: If there is no process running, select one process with the earliest ready time among processes that haven't run. Otherwise, check if the time quantum timer expires by calculate the time between current and the last time timer expires. If yes, select an unfinished process. If no, let the running process keep running and return its id.

      c. SJF: Since FIFO is non-preemptive, if there is a process running, let it keep running and return its id. Otherwise, select one process with the shortest execution time among processes that haven't run.

      d. PSJF: This is similar to SJF, but it do not check if there is a running process. Select one process with the shortest remain execution time among unfinished processes.

2. Linux kernel version: 4.14.25

3. Compare actual results and theoretical results:
   The time between start time and finish time of each child process looks not bad, since they just keep counting units of time and execute on an independent CPU. Yet, considering that there are still some other processes in the machine using this CPU, it is slightly different between actual time and theoretical time. As for the start times of the child processes, they are more different from theoretical results. The main process has its own “clock” computed from the number of rounds of while loop and the “clock” is not synchronized with child processes. Since the main process has to compute more things, one unit of time in main process is usually longer than that in child processes. And of course, it is longer than one precise unit of time. Thus, the start time of each child process is later than theoretical start time.