Operating System Project 1 Report

B04902012 劉瀚聲
B04902032 馬揚格
B04902040 王郁婷
B04902080 徐琮賀

1 Design

1.1 Main Structure

For each process, it's attributes (ready time, execution time, start time and process id) are stored in a structure `processData`. A structure `processList` is constructed to maintain a list of `processData`, while processes in it are sorted by ready time.

The scheduler process `S` itself is limited to run on CPU 0 with lowest nice value -20 at the beginning. Once a child process `P` is forked, `P` will limit itself to run on CPU 1, and it's nice value is determined by scheduling principle. After finishing setting these property, `P` then executes `./child`, a process that will run million empty iterations for n times, with n passed through `argv[1]`. To make `P` be able to print it's own name, it's name is passed through `argv[2]`.

To schedule, `S` idles a process `P1` and awake another process `P2` by setting nice value of `P1` to 19 and setting nice value of `P2` to -20. Child processes won't compete with `S` for CPU resources because they are affined to different CPU.

1.2 FIFO

We construct two pointers `st` and `ed` pointing to processes in `processList`. They both pointing the first process at the beginning. `st` is maintained to point the executing process (if exists), and `ed` is maintained to point the first unforked process.

`S` checks if the process pointed by `ed` is ready every time unit. Once the child process is ready , `S` forks it, and `ed` moves right. `S` waits non-blockingly for the

process pointed by `st` every time unit. Once the child process terminates, `st` moves right.

A process is awaken if:

(1) It is pointed by `st` and has been forked.

(2) It is forked and `st` and `ed` are pointing the same process.

Actually, the processes between `st+1` and `ed-1` forms the ready queue.

1.3 Round Robin

We construct two pointers `st` and `ed` pointing to processes in `processList`. They both pointing the first process at the beginning. `st` is maintained to point the executing process (if exists), and `ed` is maintained to point the first unforked process, just like that in FIFO.

A counter `cnt` is recorded, starting at zero. `cnt` increases after each time unit. Once `cnt` reaches 500, `cnt` is set back to zero, `S` idles the currently executing process, and let `st` point the next unfinished process and awake it. However, if the currently executing process terminates before `cnt` reaches 500, then `cnt` is also back to zero, and let `st` point the next unfinished process and awake it.

1.4 SJF and PSJF

We construct a pointer `ed` just like before. Also, an integer `st` stores the number of terminated process. Once a process terminates, `st` increases by 1.

As long as we need to decide which process to awake, we choose the forked but unfinished process with shortest estimated remain executing time. The remain executing time of a process `P` is estimated by substracting the original executing time by the real executed time.

In SJF, we need to decide which process to awake when the currently executing process terminates, while in PSJF, we also need to decide when a new child process is forked.