OS Term Project #1

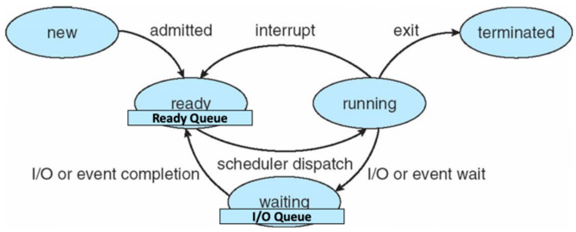
Prof. Eun-Seok Ryu, TA: Jaeyeol Choi

Project 1: **Simple scheduling**

Programming assignment #1 due by Dec. 13. 2023 (11:59pm) KST

1. About Round-Robin CPU Scheduling

Round Robin (RR) scheduling algorithm is designed specifically for time-sharing systems. It is a preemptive version of first-come, first-served scheduling. Processes are dispatched in a first-in-first-out sequence, but each process can run for only a limited amount of time. This time interval is known as a time-slice or quantum. It is similar to FIFO scheduling, but preemption added to switches between processes.



Please implement a simulator for RR scheduling, including the ready queue and I/O queue. The parent process acts as the scheduler.

2. Basic explanations and requirements

-Parent process :

- Parent process creates 10 child processes.
- Assume your own scheduling parameters: e.g., time quantum, and timer tick interval.
- Parent process periodically receives ALARM signal by registering timer event. Students may want to refer to setitimer system call. The ALARM signal serves as periodic timer interrupt (or time tick).
- Parent process manages the ready queue and I/O queue.
    - Elements of the queue should be pointer variables to structures holding information about each child process (e.g. pid, remaining CPU burst time) (think of PCB, though simpler).
- The parent process performs scheduling of its child processes: The parent process accounts for the remaining time quantum of all the child processes.
- The parent process gives time slice to the child process by sending IPC message through msgq.
    - Please note that there is `msgget, msgsnd, msgrcv` system calls, and `IPC_NOWAIT` flag.
- Decreases the remaining i/o burst time of processes in the i/o queue with every time tick.
- Total running time should be more than 1 minute.

-Child process :

- A child process simulates the execution of a user process.
- Workload consists of infinite loop of dynamic CPU-burst and I/O-burst.
- CPU burst is randomly determined at the time of child process creation.
- When a child process receives a start IPC message from the parent process, it is considered to be running.
- The CPU burst time decreases as much as the execution time.
- About I/O operation:
    - At each dispatch, the 'creation' of I/O burst is determined randomly by a certain probability.
    - If an i/o burst is created,
        - The 'start time of I/O' is also randomly determined, as the 'I/O duration'.
        - The 'start time of I/O' come before the end of the time quantum while the process is in running state.

        - When 'start time of I/O' occurs, the next child process in the ready queue executes.
        - Information about the i/o is sent to the parent process via IPC.
    - Please refer to (option 0), (option 1), (option 2), (option 3) in section 3.
- If the CPU burst time of a child process ends before the time quantum, an IPC message is sent to the parent.

- Logging:
  - The following contents are output for every time tick `t`:
  - (1) pid of the child process in running state, (2) list of processes in the ready queue and i/o queue, (3) remaining cpu burst time and i/o burst time for each process.
  - Print out all the operations to the following file: `schedule_dump.txt`.
  - Students would like to refer to the following C-library function and system call: `sprintf`, `open`, `write`, `close`.

3. Optional requirements

- There are three options for creating I/O burst time:
  - Option 0 ***(Penalty)***: Not considering I/O operations.
  - Option 1 ***(Penalty)***: When child process enters running state, whether to perform i/o and i/o duration are randomly determined. After running for the designated time quantum, the child process enters the i/o queue if needed.
  - Option 2: When child process enters running state, 'whether to perform i/o', 'i/o duration', and 'i/o start time' are randomly determined for the process. When the i/o start time is reached, the process must enter the i/o queue, and the parent process immediately dispatches another process from the ready queue.
  - Option 3: When the process is created, i/o start times and each i/o duration are randomly allocated as CPU burst time is allocated. (pre-defining 'total CPU burst time', 'i/o execution from time x_1 to y_1', 'i/o execution from time x_2 to y_2', ...)

- ***(Extra score)*** Implementing priority queue:
  - Implement multi-level queues based on priority.
  - Different policies can be applied to each queue.
  - Refer the slide: `Operating Systems 5_r1.pdf`. pages 28 and 29

- ***(Extra score)*** Writing experimental results in graphs and tables
  - Example: When time quantum parameter changes, how does average response time and total execution time differs?

4. Submission: A .zip file containing **source code**, **report (pdf)**, and **schedule_dump.txt**.

For the report (pdf), the following items should be included:
- Shell scripts (commands) to build and execute the source code.
- A table formatted below:

Whether the code implementation is complete	O / X
The setting of time quantum (which executed without errors)	ex) 50ms
Whether IPC message queue is used	O / X
How the i/o operation is implemented (one of four options in section 3)	ex) option 2
Whether the multi-level queue was implemented (extra score in section 3)	O / X

- Explanation of the code captured image of execution, and analysis of results.

For the source code, the following are recommended:
- Implementation in C/C++, Building through gcc or g++ compiler in a Linux or Windows WSL (WSL2) environment.

Any question: jaeyeol816@gmail.com

Office location: 70526