

Scheduler with IO Operations

Compile with `$ g++ -std=c++11 main.cpp` and run with `$./a.out`

Structs

Process contains name, program code file name, arrival time and last executed instruction.

Instruction contains name and execution time.

IO contains finishing time, wait queue and output file lines.

Data structures

- Processes are stored in a list.
 - `pop_front` method is used to fetch the earliest instruction.
- Ready queue is represented as a list.
 - `pop_front` and `push_back` methods are used to simulate a FIFO queue.
- IO devices are stored in a vector.
 - A specific device is accessed with the index in this vector.
- Instructions are stored in a vector.
 - Then program code file name is mapped to this vector.

Main Loop

- Each iteration of the while loop is a CPU cycle.
- In the beginning of the cycle, current ready queue is printed.
- If there are no processes waiting in the ready queue, then CPU begins to idle.
 - Otherwise next process is fetched and it is executed until there are no instructions left or until there is no quantum left for this CPU cycle.
 - If there is no quantum left then this process is pushed back into the ready queue.
- In both idle state and execution state, total time is incremented either by whole quantum or instruction execution time respectively.

- After increasing total time:
- Completed processes in IO devices are checked and finishing time is updated if wait queue is not empty.
- Processes list is checked if there are any processes that is entered while CPU is busy.
- If a process calls dispM, it is sent to printer.
- If a process calls readM
 - If cache contains the requested block, it is read from cache and LRU block is updated.
 - Otherwise process is sent to hard disk and cache is updated.
- After sending process to IO device, process is added to the wait queue of IO device.
 - If wait queue was empty, finishing time of device is calculated.