Munachimso Aneke

CS 470

Process Management Project 3

RUNNING/ BUILDING PROJECT

Project created to run on Linux. Makefile included in project for compilation. Command to run program: "./process <input file name> < output file name> <quantum number>".

DISCUSSIONS

The main of my program receives 3 arguments – input file name, output file name, and the quantum number. The initial process (PID 0) is initialized. A while loop is used to continuously reading commands from the input file, and the right command action is carried out.

The PCB for this program is implemented using a struct. The struct stores the pid, burst time, parent id, quantum time, and event id as integers; and stores the children id in a vector of integers. The ready and wait queues are implemented with queues and manipulated with the Queue stl functions- pop(), front(), push(). Using queues did make iteration a bit difficult.

The 2 most important functions are the destroy_process() and the timer_interrupt(). The algorithm for destroy_process(), starts with checking if the process to be destroyed is the current process. If it is, then it simply sets the current to the next item on the ready queue. But it if the process to be destroyed is not the current process, it proceeds to check the ready and wait queue until it finds the process. If the process has children, it recursively calls the destroy process function again. The timer_interrupt() is another important function. The timer_interrupt() checks if a process is completed, if the quantum time is 0, decreases the burst and quantum time, and prints out the queue. If the number received is not 0, the interrupt function calls the event() function.

QUESTIONS

1. The RR scheduling is cyclic in nature, each process gets the same amount of time unless it is interrupted or destroyed. The quantum time, has to thoughtfully chosen. If the quantum time is too small there will be way too much context switches.
2. I found creating the destroy process function the most difficult
3. Create process was the easiest to write.
4. I would change the data structure I used to implement the queues. I would rather create a class that implements a queue using array. This would make iterating the queue easier.
5. From the implementation, it seems the best scheduling process will be a multi-level feedback queue, since priority cannot be assigned in RR.