The project’s objective is to create a simple time-sharing system using a round-robin selection algorithm. In this system, all processes have equal priority, the context switching time is assumed to be fixed, and all processes are assumed to fit into the main memory.

Program description

The program is written in Python 3.7. It will first open an input file and read the number of processes, quantum, and processes, then create a list of processes that each element is a list of process id and process time. Also, prepare an audit log for recording the process starting and ending times, and pass them to the round-robin function.

This function first prepares a queue and puts each process into this queue. Then, CPU time will be set to zero and the simulation of the time-sharing system begins. First, CPU time will be incremented by one to represent the overhead for running the dispatcher. Following this, a process will be picked from the queue. Next, CPU time will be incremented by one to represent the overhead for restoring the new process’s context before it can begin running. After this, the current CPU time and running process id will be recorded to an audit log with the log “Process start”. If the process time was equal to or shorter than the quantum, CPU time will be incremented by the process time and the current CPU time and running process id will be recorded to an audit log with the log “Process complete”. Otherwise, CPU time will be incremented by the quantum, the quantum will be subtracted from the process time, CPU time will be incremented by one to represent the overhead for saving the running process’s context, the process will be put back to the queue with remaining process time, and the current CPU time and running process id will be recorded to an audit log with the log “Process unfinished within the quantum. Still need process\_time to complete”. Once the queue got empty, the total CPU time taken will be recorded to an audit log with the log “All processes completed in cpu\_time”.

The round-robin function will return an audit log and this will be written in an output file.

Instruction on running the program

1. Prepare an input text file. The program expects to receive single space separated values. Edit the code with open(‘input.txt’, ‘r’) to your input file name as needed. The input file format is as follows (tab separated in here for visual purpose):

number\_of\_processes quantum

process\_id1 process\_time1

process\_id2 process\_time2

: :

2. Put an input file to the place where the program is located. Run the program, then you will find an output file named “audit\_log.txt” in the same place. (I recommend seeing audit log in the console by uncommenting print(audit\_log) as an output to Windows txt file does not look very clean.) The output file format is as follows:

Time Process\_ID Log

cpu\_time1 id1 log1

cpu\_time2 id2 log2

: : :