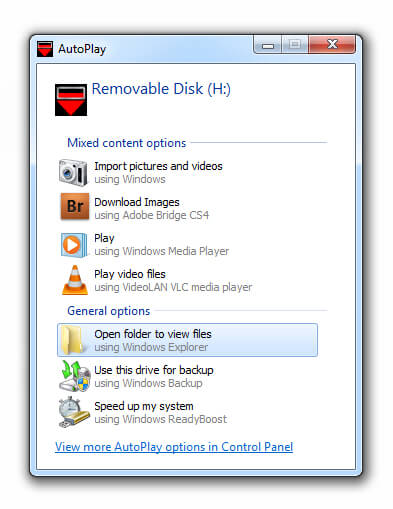
**Operating System Simulation: User Documentation**

**Group 1**

**Introduction**

This program simulates an operating system. The simulated OS performs high level scheduling, memory management, and interrupt handling for a number of running processes. This simulated computer runs with 16Mb of RAM, a timeslice occurs every second, and fetch-execute-check interrupt cycles takes a tenth of a second.



**Program Usage**

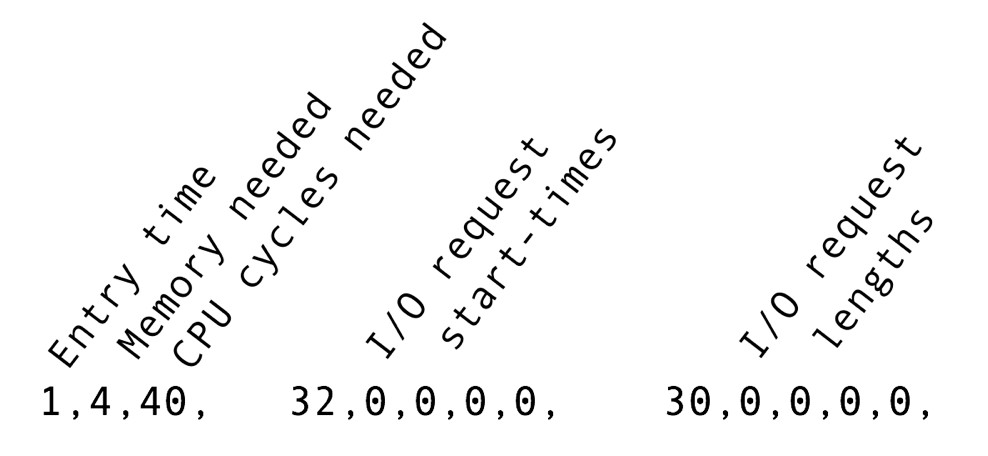
The simulation performs within a text based command-line interface. To open the program, insert the provided USB drive. If the USB drive won’t plug into the computer, turn the computer over. There will be a pop-up… click the option called ‘**Open folder to view files**’.Within this folder, there is the executable program called **simulator.exe** and an accompanying file called **Input.txt** that is used for process entry.

**Double click** the executable to start the program. Alternatively, power users may access the file using the terminal.

**Process Entry**

Provided is a sample text file named **Input.txt**. It contains the data pertaining to nine processes that are fed into the simulation. You can alter this file to or make your own input file named **Input.txt** and place it in the same directory as as the program. Be sure there is only one file named **Input.txt** in the directory.

Inside this text file, processes have comma separated values of the necessary aspects for the simulator. Shown below is a sample process with the values labeled:



If a process needs less than five I/O requests, set the unused request start-times and lengths to zero. Some important value size conditions for the simulation are as follows:

* Entry time must be greater than 0
* Memory needed must be between 1 and 8 MB
* Total CPU cycles needed must be between 10 and 950
* I/O request start times must occur after the process’ entry time
* I/O request lengths must be between 25 and 50 cycles

Each of these values must be an integer. Spaces don’t matter for the program to access the values, they only aid visually in the creation of an **Input.txt** file. Finally, be sure to have commas following every value (this includes after the last process).

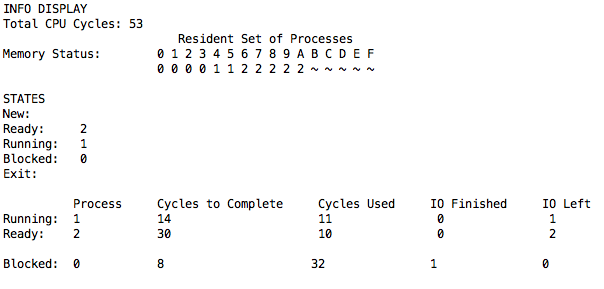
**Simulation Output**

An **info display** for the simulation shows the processes’ current status. It steps forward in time until there is a change in the system that could be:

* *A new process enters the system*
* *A process state changes*
* *An I/O request is made*
* *An I/O request is completed*
* *Memory is allocated or deallocated*
* *A process exits the system*

To step through these changes, type any key and enter it into the command line. If at any time you wish to run the program to its completion, type ‘**r**’ and then press enter.

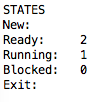
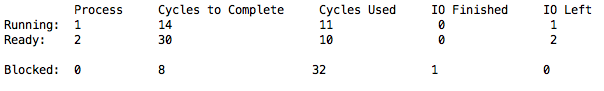
Below is a sample visual of active three processes in the simulation. Seen at the top is the Total CPU Cycles the simulated operating system has completed. Then there are three sections that will be discussed in detail: the **OS Memory**, the **Process States**, and the **Active Processes**.



* The **OS Memory** shows the resident set of processes. This displays the simulation’s memory addresses in hexadecimal as 0-F representing the 16Mb of system memory. Below that, it shows the PIDs of the processes currently in those memory locations.



It can be seen above that **Process 0** takes up the first four frames of memory, **Process 1** takes up frames 4 & 5, **Process 2** takes up the following five frames, while the remaining five frames are empty as signified by the ‘**~**’ symbol.

* The **Process State** diagram lists the five states that a process may be in and then the processes in each state.  
    
  Here is is seen that **Process 1** is running, **Process 2** is ready to run, and that **Process 0** is blocked by an I/O request. When processes enter the OS, they join the the New state and stay there until there is enough free memory to enter the memory manger’s **Round-Robin** system of sharing time slices of CPU time. This is a regular queue where ready processes wait their turn to run.
* The **Active Processes** section is a detailed list of processes in the Ready, Running, and Blocked states.   
    
    
    
  This list shows the PIDs of the processes given to them by the high-level scheduler, the CPU time needed for them to complete, the CPU cycles already completed so far, I/O requests satisfied, and I/O requests outstanding.

**Trouble-Shooting**

If the program doesn’t output the **info display** and outputs a warning of an “Invalid entry,” read the description of the error and reference the section called **Process Entry** to fix your **Input.txt** file.

**Conclusion**

This is a powerful tool for understanding operating systems. **Good luck and have fun!**