OSSIM - An Operating System Simulator

Richard M. Reese., PhD
Assistant Professor of Computer Science, Tarleton State University
rreese@tarleton.edu

Abstract

Operating Systems concepts are best learned through implementation. However, this can be difficult and time consuming. To support this learning process an OS simulator has been developed that allows students to learn about OS concepts using student-written Java code fragments

Introduction

OSSim is an operating system simulator. It was developed to monitor different core elements of an operating system including:

- User-defined process scheduler
- Memory simulation including heap management
- Process simulation
- File simulation

In addition, the user is able to provide their own Java methods to control the elements of the simulator.

The OSSim can be used to explore the behavior of the operating system. For example, the user can step through each process to see when it has access to the processor, how long it has had access, and where in memory it is located. The use of memory is visually depicted in the simulator based upon which memory allocation method the user chooses.

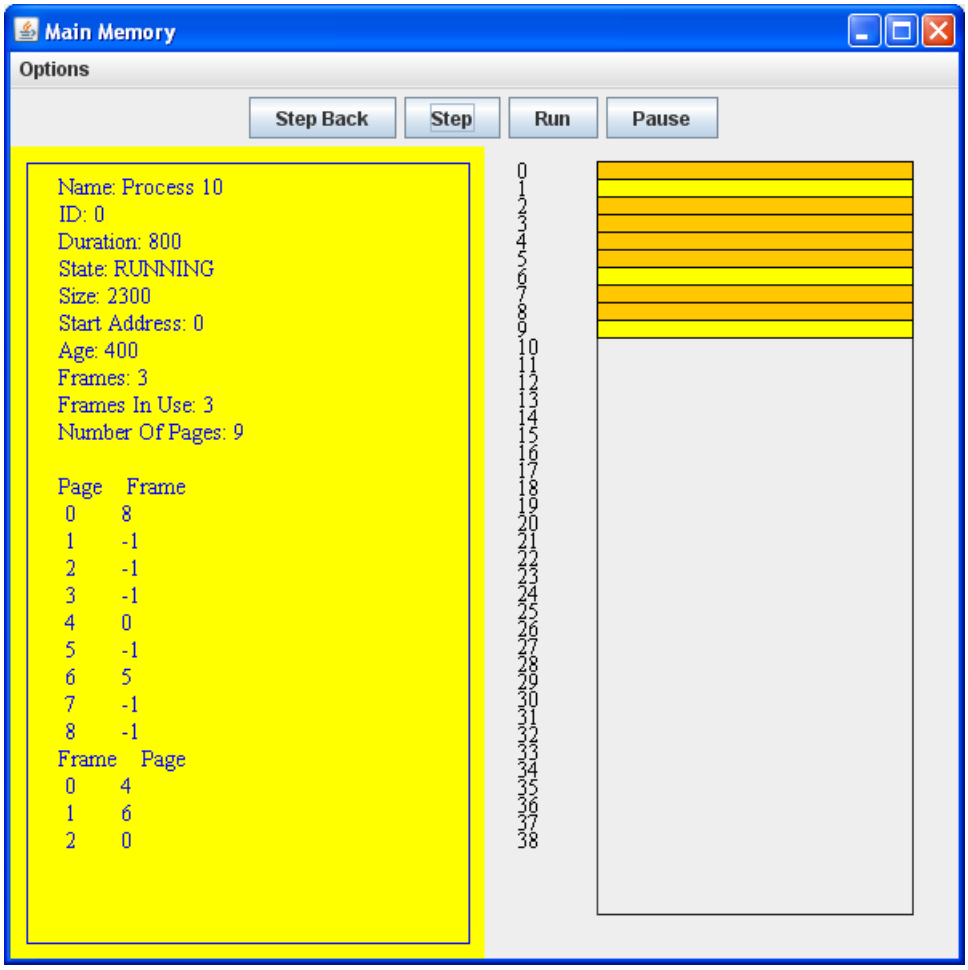
The user is able to replace certain elements of the simulator with their own Java code. This modular approach permits the user to experiment with different implementations of different algorithms. Through this experimentation the user is able to gain a better understanding of the behavioral and performance characteristics of the OS element under consideration.

Using the Simulator

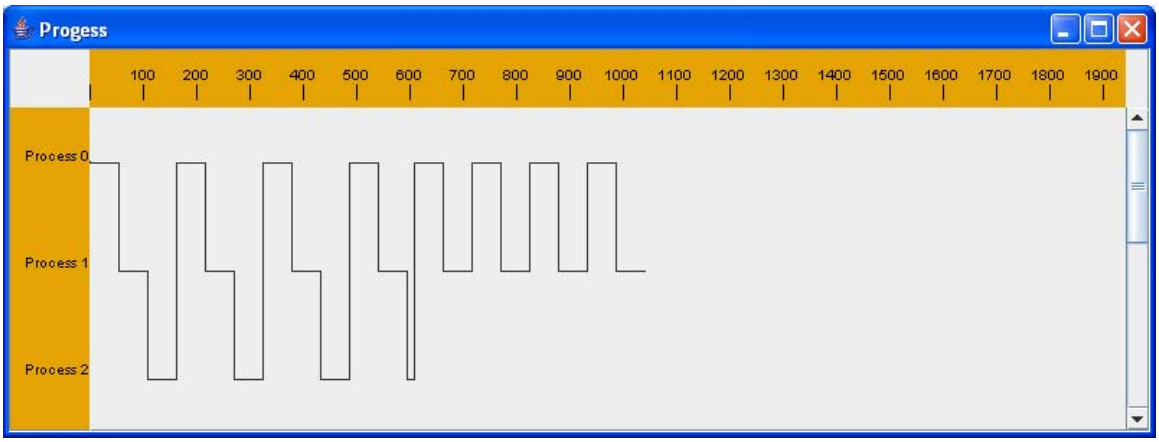
The simulator can be started using default element implementations. If the user choices to replace specific elements, then the appropriate section of the simulator needs to be modified to use that replacement. The present implementation of the simulator requires re-compiling the simulator.

The simulator can be run without the user having to provide element implementations. For example, if no scheduler is provided, the scheduler will default to Round-Robin.

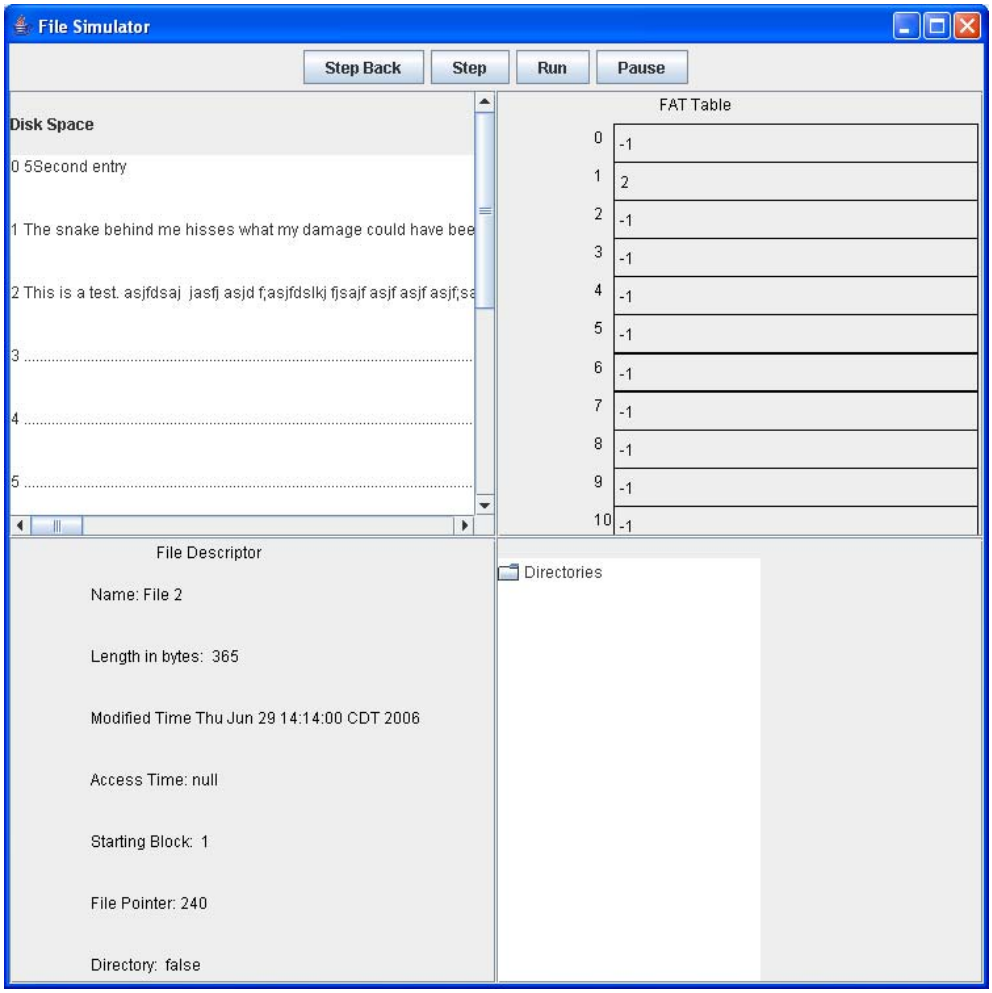
The process simulator will then display the block of memory whether it is contiguous in a segmented mode or virtual memory being manipulated by the process scheduler. As the state of a process changes, the colors of the processes displayed will change, as well, as the process descriptor.



Also displayed, is a progress simulation, showing the time increments that each process obtains.



The user may also create a series of OSFiles objects to write or read from. As the files are created and written, the File Simulator will display the File Allocation Table and the Disk of memory being manipulated, as well, as the file descriptor and directory structure.



In each of these simulations, there are four buttons across the top of the screen, consisting of Run, Step, Back, and Pause. The run button will perform the specified actions. If the user chooses to step through the simulation, after each process time slot has completed or file I/O procedure terminates, the program will pause, allowing the user to view how each process or file is being used. The user may pause or run through the actions at any time. The back button allows the user to step back to the last procedure.

Programming the Simulator

Depending on the element to be replaced, the user can provide there own java implementation. For example, in order to simulate a different number and types of processes the user will need to replace the code used to create the initial set of processes.

The startProcess method is used to create the initial set of processes and start the scheduler.

```
private void startOS() {
        // Specify the OS memory model used
        memoryMode = OSMemoryMode.VMA;

        // Create processes
        OSProcess process = createProcess(null,2,800,2300,"Process 10",3);
        priorityQueue.add(process);
        process.start();

        process = createProcess(null,2,800,4300,"Process 11",5);
        priorityQueue.add(process);
        process.start();

        process = createProcess(null,2,800,1500,"Process 12",6);
        priorityQueue.add(process);
        process.start();

        // Set and start the scheduler
        Scheduler scheduler = new RoundRobinScheduler(100);
        invokeScheduler(scheduler);
}
```

The createProcess method is an overloaded method that permits the user to specify:

- Process - The process thread (null is used typically)
- Priority - The priority, if desired, of the process
- Duration - How long the process should run
- Starting Address - The starting address in memory
- Size - The size of the process
- Name - The name of the process

There are other ways of creating and adding processes to a simulation.

In addition, the user of the simulator is able to model the action of a heap manager and file IO in as similar manner.

Findings

The simulator has not been used in an OS class at present. Its first use will be in the Spring of 2007. At that time the utility of the simulator can be better assessed.