Investigating Process Scheduling

Introduction

Objectives

At the end of this lab you should be able to:

- Explain what software simulators can do
- Use an operating system simulator
- Use the simulator to demonstrate different scheduling policies
- Explain what threads are and how they are created

Dynamic System Simulators

Software-based simulators are created to represent or predict the behaviours of dynamic systems using visualisation techniques or calculated values representing changing system states. Dynamic systems have well defined states and rules, which govern transitions between different states. These simulators have three main stages: input data which can include feedback data; sets of algorithms defining system states and transitions which process the input; output data which can also be fed back to the input (feedback loop).

Following are some of the many areas software-based simulators are employed in

- Supporting educational programmes
- Researching novice techniques and technologies
- Demonstrating ideas and concepts
- Exploring and investigating new ideas
- Developing new products
- Predicting future system behaviours
- Investigating system faults
- Playing games

Operating System Simulator (OSS)

The OSS is a visualisation-based simulator and represents the behaviour of a modern operating system, which itself is a dynamic system, created to support the teaching of operating system concepts. It is part of a set of educational simulators, which also includes a

Language Compiler Simulator (LCS) and a Processor Hardware Simulator (PHS). Together this set of simulators supports the delivery of modern undergraduate modules on operating systems and computer architecture.

This tutorial is about operating systems, so we are only concerned with the use of the OSS. The important operating system concepts such as process and memory management are difficult to directly observe and visualise using commercially available operating systems such as Linux and Windows for obvious reasons. The few educational operating systems are not always suitable to work with, and often require students to have a good level of system programming understanding.

The OSS is specifically developed to provide pedagogic features and is a visualisation-based simulator providing many animated system behaviours, which can be controlled and modified to demonstrate different concepts.

Introducing OSS

OSS provides means for exploring the technology and functionality of operating systems, which enables students to develop a deep understanding of the way the operating systems work. OSS is able to simulate the scheduling and running of sets of processor instructions, known as processes, as if it is a real operating system. It is also able to simulate multi-processing of two or more processes on one or more processors thus simulating multiprocessor systems. The user interface is designed to enable the user to define the various parameters used in the simulations.

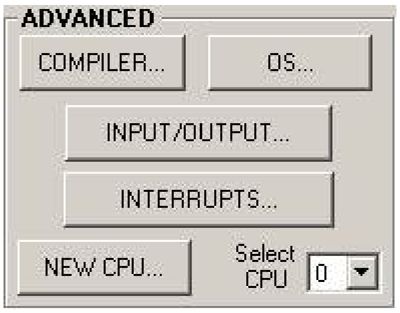
The following sections will describe a series of experiments, which can be carried out using the simulator.

Obtaining and/or installing the simulator software

You need to obtain, and if required, install the simulator software before you can do the exercises.

Running the simulator

Double-click on the executable **CPUSimulator.exe** or the associated icon, if one is made available (usually installation will create one), in order to start running the simulator. On successful load you will see the main simulator window. You access the OSS by clicking the **OS…** button (see the **ADVANCED** view below). You can ignore the rest of the main simulator window.



You can access advanced features of the simulator through the **ADVANCED** view. Click on the **OS…** button to start the operating system simulator.

Image 0 - Advanced view

OSS User Interface Explained

The following are brief descriptions of different elements of the user interface OSS provides. You will learn how to use them as you carry out the experiments.

The interface is divided into several different views. Two of them represent the operating system's ready and waiting queues with each showing the processes in the queue. A third view shows the running process (or processes if multiple processors are used). These views and other related views are described further below.

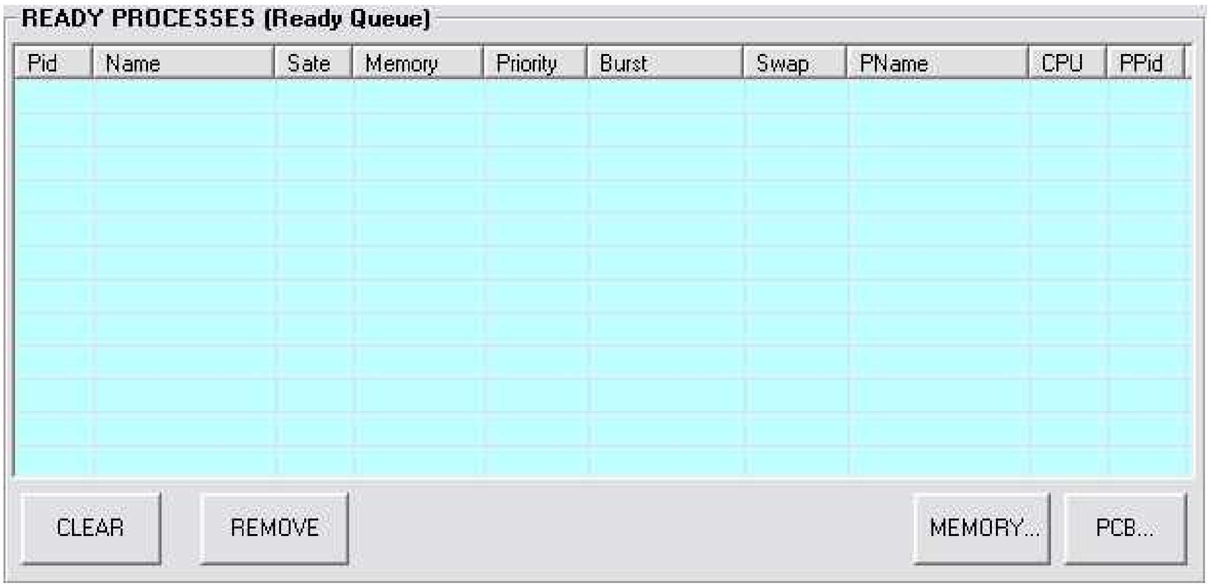
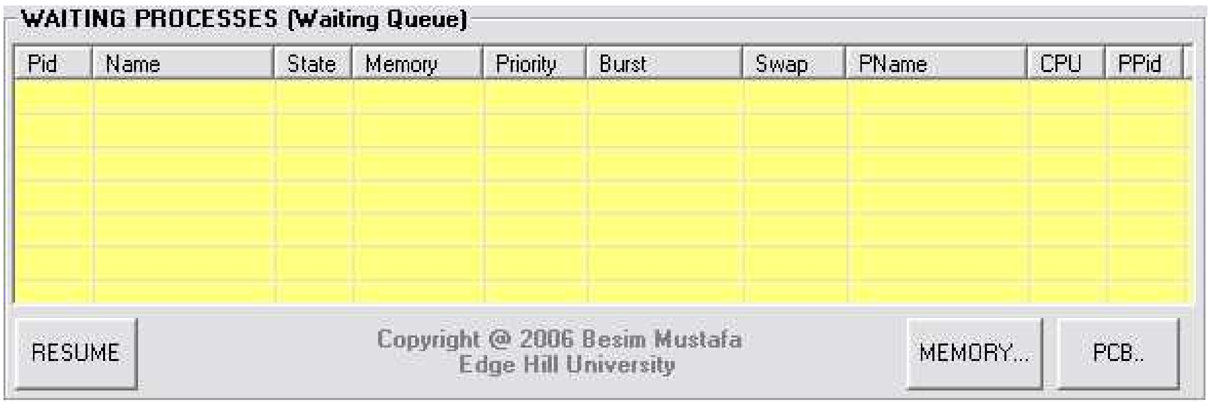


Image 1 – Ready queue view

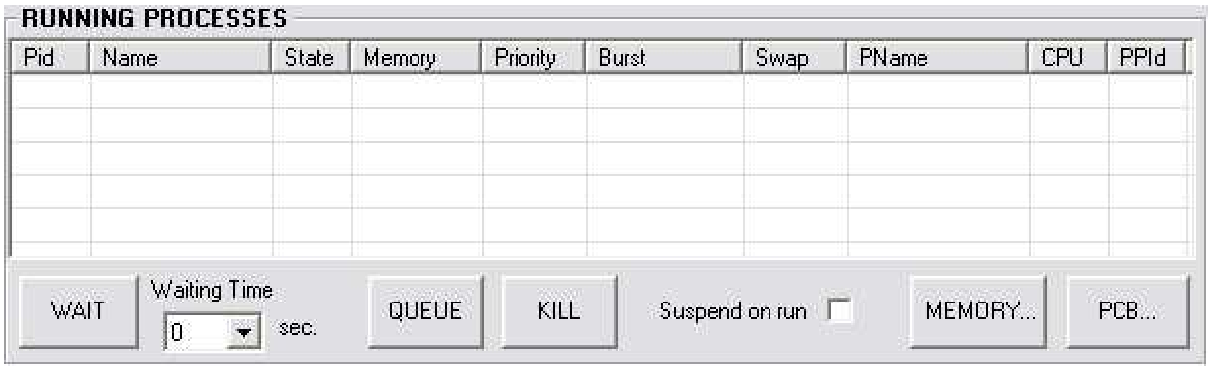
The ready queue view displays those processes, which are currently awaiting dispatching by the operating system (i.e. waiting to be run). Once a processor is available (i.e. no more running a process), a process in this queue will be selected to run next (this selection depends on what the current scheduling policy is – see below).


WAITING PROCESSES (Waiting Queue)
RESUME
Copyright @ 2006 Besim Mustafa
Edge Hill University
MEMORY...
PCB..

Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPid

Image 2 – Waiting queue view

The waiting queue view displays those processes, which are currently waiting for an event to complete. This is simulated by holding the processes in this queue for defined time periods. At the end of this period a process moves to the ready queue, which can be observed in the ready queue view.



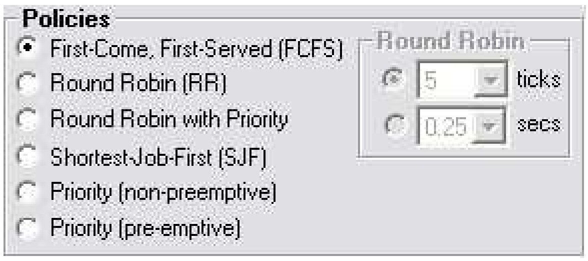
Pid	Name	State	Memory	Priority	Burst	Swap	PName	CPU	PPId

Waiting Time 0 sec.

Suspend on run ☐

Image 3 – Running process view

The running process view displays the process currently running (i.e. the processor is executing this process' instructions). The currently running process can either run until it naturally terminates or is pre-emptively moved back to the ready queue after a defined period of time. This depends on the type of scheduling policy selected (see below for the policy options supported by the simulator).

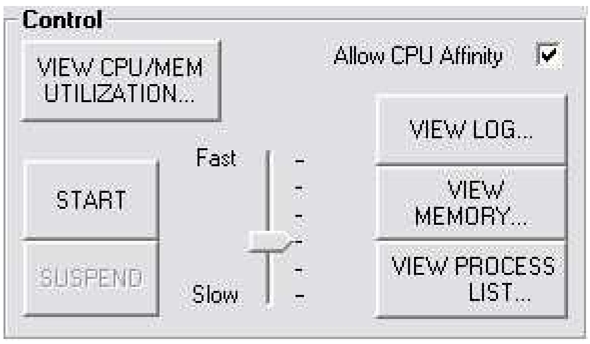


[✓ First-Come, First-Served (FCFS)]
Round Robin (RR)
Round Robin with Priority
Shortest-Job-First (SJF)
Priority (non-preemptive)
Priority (pre-emptive)

Round Robin: [✓ 5 ticks] 0.25 secs

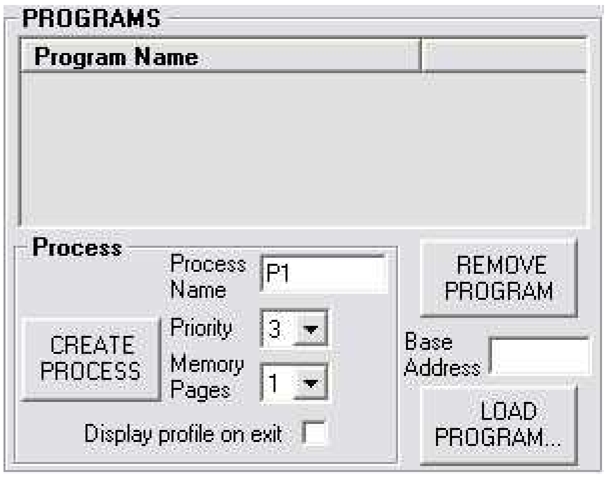
This view presents various scheduling policies supported by the simulator. The user selects one, which is then used when the scheduler is started. The Round Robin policies also need to specify time slot values.

Image 4 – Scheduler policies view



This view presents the scheduler control, which starts and suspends the scheduler simulation. You can also view the log, the main memory, list of processes, CPU and memory utilization information. You can also adjust the speed at which the process instructions are executed by using the slider control.

Image 5 – Scheduler control view



This view presents the list of programs the simulator can run. The processes are created (or instantiated) from these programs. You can specify process name (a default is available), process priority and the number of memory pages the process should be allocated. You can also load or a previously saved program and remove a program in this view.

Image 6 – Programs view

Simulator-based Exercises

Now, let’s start using the simulator. You may wish to first familiarize yourselves with the various views as described above. So, spend a few minutes doing this. Ask your tutor(s) if you need any assistance. You may ignore any part of the simulator not mentioned in this document.

Loading programs and creating processes

OSS requires one or more programs to be loaded before it can run the simulations. You do this by clicking on the LOAD PROGRAM… button in the PROGRAMS view (see Image 6 above). After successfully loading a program you need to create one or more instances of it as processes. Follow the steps below to do this

1. Click on **LOAD PROGRAM…** button
2. Select the file **FORNEXTLOOP.sas** - you'll see the entry FORNEXT appear in the **PROGRAMS** view in **Program Name** column
3. Click on **CREATE PROCESS** button

➢ ***Can you explain the result of action 3 above?***

Running processes

Once a process is created it is a candidate for scheduling to run by the operating system. In a real operating systems the scheduler is always active looking for the next process, if one is available, to run as soon as the CPU becomes available. In our simulator the user must manually start the scheduler. To start the simulator do the following

Click on the **START** button in the **Control** view (see Image 5 above)

- ***Can you explain what happened?***
- ***What kind of information is displayed about a process?***

You may wish to increase the speed of process execution by using the slider control towards the **Fast** label. After a short while the process will naturally terminate.

Operating system log

The simulator maintains a log of events of interest. To show the log do the following

Click on the **VIEW LOG…** button in the Control view (see Image 5)

- ***What sort of information is displayed in the log window?***

Please note that this is specific to this simulator. Real operating systems may or may not maintain similar information.

Investigating different scheduling policies

The simulator provides different scheduling options. These are selectable by the user in Policies view (see Image 4 above). The following exercises are designed to investigate different scheduling algorithms.

- ***What is an algorithm?***

1. Select the **First-Come, First -Served (FCFS)** policy
2. Create two processes
3. Set the speed of execution of processes to **Fast** (slider at top)
4. Start the scheduler
5. Wait until all processes normally terminate

- ***Explain the behaviour of this scheduling policy.***

6. Select the **Priority (non-preemptive)** policy
7. Create a processes with priority 3
8. Create a process with priority 2
9. Create a process with priority 4
10. Create a process with priority 3

- ***Explain the behaviour of this scheduling policy.***

11. Set the speed of execution of processes to **Fast** (slider at top)

12. Start the scheduler
13. Wait until all processes normally terminate
14. Select **Round-Robin** scheduling policy
15. Select "10 ticks" as the time slot
16. Create three processes
17. Start the scheduler
18. Wait for a short time (2 seconds?) then click on the **SUSPEND** button in **Control** view (see Image 5).
19. Select a process in the **Ready Processes** view (i.e. the ready queue) and click on the **PCB…** button in the same view.
    - ***What do you see?***
    - ***Observe the number against the PC Register. What is the significance of this number?***
    - ***What other important information do you see about this process in the displayed window?***
20. Now, click the **RESUME** button
21. Wait until all processes normally terminate
    - ***Explain the behaviour of this scheduling policy.***
22. Select **Priority (pre-emptive)** scheduling policy
23. Set speed of execution of processes to near the **Slow** end
24. Create a processes with priority 2
25. Create a process with priority 3
26. Start the scheduler
27. Now, create a process with priority 1
    - ***Which process is running?***
    - ***Which processes are in the ready queue?***
    - ***Explain the behaviour of this scheduling policy.***
28. Wait until all processes normally terminate
29. Select the **First-Come, First -Served (FCFS)** policy
30. Create two processes
31. Set the speed of execution of processes to **Fast** (slider at top)
32. Start the scheduler
33. Now, in the **RUNNNING PROCESSES** view (see Image 3), select Waiting Time of 10 seconds then click on the **WAIT** button.
34. Now observe the behaviour of the process which was running prior to you clicking the WAIT button (you will have to wait a short while).
    - ***Explain the behaviour of this process.***

- ***What state transitions did it go through?***

35. Wait until all processes normally terminate
36. Now, view the log and trace the results of your actions so far.

Additional exercises (optional)

When a process is created, it starts in the ready queue, then, at some stage, it is scheduled to run and starts running. This process can either terminate naturally (its code decides when) or by external intervention, e.g. the user or the OS.

1. Select the **First-Come, First -Served (FCFS)** policy
2. Create two processes
3. Set the speed of execution of processes to **Slow** (slider at top)
4. Start the scheduler
5. Click on the **KILL** button
   - ***What happened?***
   - ***Is another process running?***
6. Wait until the process normally terminates

Remove the program by clicking on the **REMOVE PROGRAM** button in the **PROGRAMS** view (see Image 6).

7. Click on **LOAD PROGRAM…** button
8. Select the file **THREADTEST.sas -** you'll see the entry THREADTEST appear in the **PROGRAMS** view in **Program Name** column
9. Click on **CREATE PROCESS** button
10. Select **Round-Robin** scheduling policy
11. Select "10 ticks" as the time slot
12. Set the speed of execution of processes to **Fast** (slider at top)
13. Start the scheduler
14. Observe what happens. When three processes are created, click on the **SUSPEND** button.
15. From Control view, click on the **VIEW PROCESS LIST…** button.
16. Click on the **PROCESS TREE…** button.
    - ***What do you see? Can you explain what you see?***
    - **Why are there three processes when you created only one?**