Investigating Process Scheduling 2

Introduction

Objectives

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

1. Enter source code in the compiler and compile it.
2. Load the generated code in the CPU simulator's memory.
3. Create processes from programs in the OS simulator.
4. Select different scheduling policies and run the processes in the OS simulator.
5. Explain the differences between pre-emptive and non-pre-emptive scheduling.
6. Locate the PC Register value in a process's PCB when it is in the ready queue.
7. Explain how the PC Register value in PCB is used when in Round Robin scheduling.

Processor and OS Simulators

The computer architecture tutorials are supported by simulators, which are created to underpin theoretical concepts normally covered during the lectures. The simulators provide visual and animated representation of mechanisms involved and enable the students to observe the hidden inner workings of systems, which would be difficult or impossible to do otherwise. The added advantage of using simulators is that they allow the students to experiment and explore different technological aspects of systems without having to install and configure the real systems.

Basic Theory

The different OS scheduling policies are discussed during the lectures on Process Management. This tutorial is based on these lectures.

Lab Exercises - Investigate and Explore

Start the CPU simulator. You now need to create some executable code so that it can be run by the CPU under the control of the OS. In order to create this code, you need to use the compiler which is part of the system simulator. To do this, open the compiler window by selecting the **COMPILER...** button in the current window.

In the compiler window, enter the following simple code in the compiler source editor area (under **PROGRAM SOURCE** frame title):

```
program LoopTest
    I = 0
    for N = 0 to 40
        I = I + 1
    next
end
```

Now you need to compile this in order to generate the executable code. To do this, click on the **COMPILE...** button. You should see the code created on the right **PROGRAM CODE** view. This code needs to be loaded in memory so that the CPU can execute it. To do this, click on the **LOAD IN MEMORY** button in the current window. You should now see the code loaded in memory ready to be executed.

We are now going to use the OS simulator to run this code. To enter the OS simulator, click on the **OS...** button in the current window. The OS window opens. You should see an entry, titled **LoopTest**, in the **PROGRAM LIST** view. Now that this program is available to the OS simulator, we can create as many instances, i.e. processes, of it as we like. You do this by clicking on the **CREATE NEW PROCESS** button. Repeat this four times. Observe the four instances of the program being queued in the ready queue represented by the **READY PROCESSES** view. Make sure the **First-Come, First-Served (FCFS)** option is selected in the **SCHEDULER/Policies** view. At this point the OS is inactive. To activate, first move the **Speed** slider to the fastest position, then click on the **START** button. This should start the OS simulator running the processes. Now, follow the instructions below without any deviation:

1. Make a note of what you observe (you need to concentrate on the two views: **RUNNING PROCESSES** and the **READY PROCESSES** during this period).

2. When all the processes finished, do the following. Select the **Priority (static)** option in the **SCHEDULER/Policies** view. Then select the **Non-**

preemptive option in the **SCHEDULER/Policies/Priority Type** frame. Create three processes with the following priorities (use the **Priority** drop-down list in the **PROGRAM LIST/Processes** view: 3, 2, 4. Slide the **Speed** selector half-way down, then hit the **START** button. While the first process is being run do the following. Create a fourth process with priority 1. Make a note of what you observe.

3. Now, slide the **Speed** selector to fastest position and wait for the processes to complete (or use the **KILL** button to terminate the processes one by one). Next, select the **Pre-emptive** option in the **SCHEDULER/Policies/Priority Type** frame. Create three processes with the following priorities: 3, 2, 4. Slide the **Speed** selector half-way down and then hit the **START** button. While the first process is being run do the following. Create a fourth process with priority 1. Make a note of what you observe.

4. Slide the **Speed** selector to fastest position and wait for the processes to complete (or use the **KILL** button to terminate the processes one by one). Select the **Round-Robin (RR)** option in the **SCHEDULER/Policies** view. Then select the **Non-preemptive** option in the **SCHEDULER/Policies/Priority Type** frame. Create four processes and hit the **START** button. Make a note of what you observe.

5. Wait for all the processes to complete. Go to the compiler window (use the **COMPILER...** button in the **GO TO** frame for this). Click the **NEW** button in the **PROGRAM SOURCE** view and enter the following source code:

```
program LoopForeverTest
    N = 0
    while true
        N = N + 1
    wend
end
```

Compile this code and load it in memory as previously described above. Go to the OS window (use the **OS...** button for this). You should now see an additional entry in the **PROGRAM LIST** view titled **LoopForeverTest**. Select this entry by clicking on it. We are now going to create processes of this program but this time we will assign a lifetime (in seconds) to each. We'll create three processes with the following lifetime values (use the **Lifetime** text box for the values and also select the **Secs** radio button): 10 seconds, 32 seconds, 6 seconds. Now, we are going to select the time slot value. To do this select 4 seconds from the

drop-down list in **SCHEDULER/Policies/RR Time Slice** view. Hit the **START** button and wait until all processes finish.

Open the OS Activity Log window by clicking the **VIEW LOG...** button in the **SCHEDULER/Control & Monitor** view. Observe the relevant log entries and check out the sequence of the running processes and note the time spent by each process during each running state.

6. Now, go to the compiler and enter the following code in the source code editor. You need to click on the **NEW** button to start a new source code tab in the editor:

```
program LoopForeverTest2
   while true
       n = n + 1
       i = i + n
       p = n + i + 5
   wend
end
```

Compile this new source and load in memory. Go to the OS simulator. You'll observe a third entry in the **PROGRAM LIST** view titled **LoopForever2**. Click on the entry **LoopForever** and create a process. Click on **LoopForever2** and create a process of it. There should now be two processes in the ready queue. Make sure the **Round Robin** scheduling is selected, the priority type is **Non-preemptive** and that the **RR Time Slice** is 10 ticks (you can select this from the drop-down list). Put the simulation speed to half-way (using the speed selector).

Finally, go to the CPU simulator window, click on the **SHOW PIPELINE...** button and in the new window check the **No instruction pipeline** check box. Close this window and go back to the OS Simulator window.

Now we are ready for the next experiment. Do the following:

1. Select each process in turn and click on the **PCB...** button. Observe the values of the **PC Registers** in each case and make a note.
2. Click on the **START** button and keep your mouse on the **SUSPEND** button but NOT clicked yet.
3. As soon as the currently running process is put back in the ready queue, click on the **SUSPEND** button.
4. Select the process in the ready queue and click on the **PCB...** button in the ready queue. Make a note of the **PC Register** value.

5. Now, select the **Suspend on run** check box in the **RUNNING PROCESSES** view. Slow down the OS simulation further (say two-thirds down the scale). Click on the **RESUME** button and observe when the queued process is put back in running state at which point the simulation will be automatically suspended. Keep your eye on the **PC Register** box in the CPU simulator window (you may have to re-arrange the simulation windows for this). Now, click on the **RESUME** button in the OS simulator. Make a note of the value in the **PC Register** when it changes to a new value. Comment on this value and explain what is happening.