Process States and Memory Management

Learning Objectives

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

- Demonstrate three main states of processes
- Identify allowed and disallowed process state transitions
- Show entry in and exit from waiting state on two types of events
- Demonstrate three placement strategies used in memory management
- Demonstrate virtual memory activity as part of memory management

Tutorial Exercises

Initial Preparation

This tutorial uses the **CPU-OS Simulator** software which simulates the functions of a typical modern CPU and a typical Operating System (OS). To start the simulator, double-click on the simulator icon or name. This starts up the CPU simulator's main window. Next go to the OS simulator's main window by first selecting the **Advanced** tab and then clicking on the **OS 0…** button.

Please read the instructions carefully before you attempt the exercises using the simulator. If at any point the simulator crashes, restart it and carry on from where you left.

LO 1: Investigating *Process States and State Transitions*

In this section we'll look at different process states and the transitions from one state to another. Some of these transitions are allowed and some are not, i.e. illegal transitions.

In the OS Simulator window select the **Views** tab. Click on the **VIEW PROCESS STATES…** button. You'll see the **Process States** window displayed. This window shows a graphical representation of the **Ready Queue** where the process is in Ready State. It also shows a representation of the CPU where the process is in Running State. Also shown is the representation of the **Waiting Queue** where the process is in Waiting State.

In this window check the **Stay on top** and **Animate** check boxes. Back in the OS Simulator window select the **Program** tab and load ***ForeverLooping*** program. Now, select the **Process** tab and create a single process from this program by clicking on the **CREATE NEW PROCESS** button.

In the **Process State** window you'll see a single process at the head of the Ready Queue. This process is represented by a coloured box. Its process number is displayed on the box. Now we are ready to investigate process states. To do this, follow the actions in the table below and make a note of the results of your actions.

Note that the actions will involve dragging and dropping the process box into various areas. If the dragging and dropping action results in a failure then the following message is displayed at

the bottom of the window: ***ERROR: Illegal state transition!** This means the transition is not allowed.

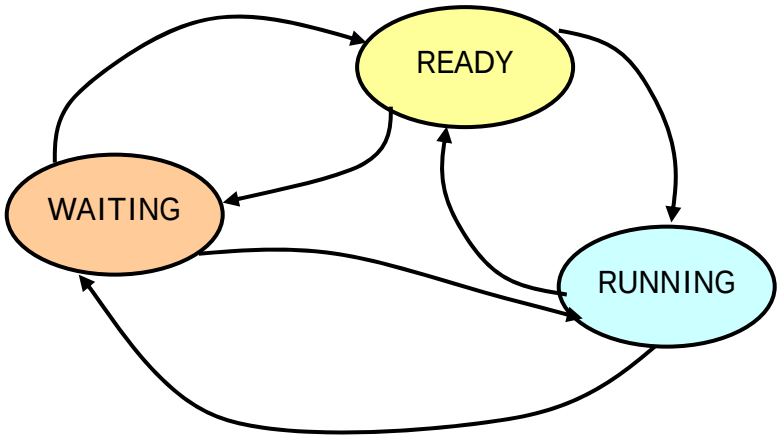
Finally, before you start, in the OS Simulator window check the **Suspend on state change** check box in the **READY PROCESSES** area only and slide the **CPU Speed** slider to the fastest position. The **Suspend on state change** suspends the simulator and displays a message whenever a process changes its state. You'll need to click on the **RESUME** button to continue. Now execute the actions below and fill in the three columns on the right:

Action to take	**Resultant State**	**Success (Y or N)**	**Fail (Y or N)**
1. Drag process to the waiting queue (i.e. put in waiting state)			
2. Drag process to the Process Bin (i.e. terminate it)			
3. Drag process to the CPU 0 box (i.e. run it)			
4. Drag process to the waiting queue			
5. Drag process to the CPU 0 box			
6. Drag process to the Process Bin			
7. Drag process to the Ready Queue (i.e. put it in ready state)			
8. Click on the **RESUME** button in **OS Control** tab			
9. Drag process to the ready queue			
10. Click on the **RESUME** button in **OS Control** tab			
11. Drag process to Process Bin			

Next, based on the results above, fill in only the <u>legal state changes</u> in the table below:

From State	**To State**

The **state transition diagram** below shows process states and ALL the transitions between them. Some transitions are not legal. Remove the arrows that represent the illegal transitions, i.e. the ones that are not allowed by the OS:



LO 2: Investigating *Process States – Waiting Queue and Events*

In the OS Simulator select the **Program** tab and load the program ***OSQueuesDemo***. This program runs on the CPU for a short while and then waits for 3 seconds before running on the CPU again. This repeats forever. Your task is to observe this activity. First, you need to create a single process of this program using the **CREATE NEW PROCESS** button in the **Process** tab.

Next, check ALL three **Suspend on state change** check boxes (there are three of them in OS simulator). Next, start the scheduler by clicking on the **START** button. Make a note in the table below of the sequence of the next six states of the process (Note: You'll need to click on the **RESUME** button in the **Process** tab after each display of the message window just after the state change). The staring state and the expected ending state are filled in for you.

States
1. RUNNING
2.
3.
4.
5.
6. READY

NOTE: If you need to restart, you can clear the process by using the **KILL** button (if the process is running) or the **REMOVE** button (if the process is in Ready Queue or in the waiting queue).

The above example is a demonstration of a process going into the WAITING state as a result of suspending itself for 3 seconds (that's how it was programmed). Next, we are going to look at another method of suspending a program. If a RUNNING program is waiting on an input the OS will force it to go into the WAITING state until it receives the input. To demonstrate this, do the following:

Load the program ***WaitOnReadDemo***. Then create a single process of this program. Make sure all three **Suspend on state change** check boxes are NOT checked. Also make sure the **CPU Speed** slider is in the fastest speed position. Next, in the CPU Simulator window, select the **Advanced** tab and click on the **INPUT OUTPUT…** button. You should now see the **Console** window. In this window click on the **SHOW KBD…** button which will bring up a small window representing a keyboard. Check the **Stay on top** check box on the **Console** window. Back in the OS Simulator click on the **START** button to start the scheduler.

Wait until a **>** character is displayed in the **Console** window. Now check to see what state the process is in (look at the OS simulator window) and make a note of this below:

Using the small **Keyboard** window click on any button and observe what happens in the OS simulator. Make a note below of what state the process goes into immediately after a keyboard button is clicked:

The above will be repeated as many times as you like until you enter a ***** (star) character to terminate this program.

The above demonstration was about the process being suspended by the OS as soon as it waits for an input, i.e. it stays in the WAITING state. As soon as an input is supplied it starts running again until it once again waits for another input. The OS makes sure that all processes waiting for inputs are put into the waiting state. Once the input is provided, i.e. the input/output event is completed, a waiting process comes out of the waiting state.

LO 3: *Looking at Memory Placement Policies*

The above exercises concentrated on the process states and transitions between them. The following exercises are about memory management.

Once a process is created the OS needs to find space in primary memory (i.e. the RAM) for the process's data and code. There are three main methods employed: 1) **First fit**, 2) **Best fit** and 3) **Worst fit**. To demonstrate how these work follow the instructions below:

First, make sure no programs are in memory. Remove any processes which may happen to be in any of the queues (use the **REMOVE** buttons) and also remove all the loaded programs by selecting the program and clicking on the **REMOVE** button in the **Program** tab. We are now starting from scratch!

In the OS Simulator window, select the **Views** tab and click on the **VIEW MEMORY…** button. You'll see the **Main Memory** window displayed. Now, make sure the **Frames** drop-down list is set to 1. Click on the **ADD** button repeatedly until all memory is allocated (needs 10 clicks). We will now proceed to make "holes" in the memory. Please follow the instructions below exactly in the order listed:

Using the **Frame No** drop-down list set it to 2 then click the **REMOVE** button to remove it.
Set the **Frame No** to 3 and remove it.
Set the **Frame No** to 5 and remove it.
Set the **Frame No** to 7 and remove it.
Set the **Frame No** to 8 and remove it.
Set the **Frame No** to 9 and remove it.

You should now have only frames 0, 1, 4 and 6 allocated with gaps of free memory spaces of various sizes between them.

Now, set the **Placement Policy** drop-down list to **First Fit**. Load the program ***ForeverLooping*** if it's not already loaded. Create a single process of it. Observe in which gap in Main Memory its memory is allocated. Now remove the process by clicking on the **REMOVE** button in the **READY PROCESSES** area (Note: Do not remove the program, just the process).

Next, set the **Placement Policy** drop-down list to **Best Fit**. Create a new process and observe in which gap in Main Memory its memory is allocated. Once again remove this process.

Finally, set the **Placement Policy** drop-down list to **Worst Fit**. Create a new process and observe in which gap in Main Memory its memory is allocated. Again remove this process.

Now, before you continue clear the memory by clicking the **RESET** button.

Fill in the table below, against each of the three methods, explaining how each method works:

First Fit placement method	
Best Fit placement method	
Worst Fit placement method	

LO 4: *Looking at Virtual Memory and Swapping*

When a process is created it is also allocated some memory from free memory space in RAM. If this memory runs out of free space then the process's memory will be swapped onto a secondary storage like a hard drive. In this case, the hard drive will be regarded as an extension of the primary memory (i.e. virtual memory). However, when this process is scheduled to run next then its swapped out memory must be brought into RAM first. This action will usually result in another process's memory space being swapped out onto the hard drive in order to make space for the incoming process. The following exercise demonstrates this activity.

In the OS Simulator window select the **Views** tab. Click on the **VIEW UTILIZATION…** button. This will show the **Resource Utilisation** window. Here you can see the CPU and memory utilization as bar charts (both RAM and virtual memory shown). Next, click on the **VIEW MEMORY…** button in the OS Simulator. You'll now see the **Main Memory (RAM)** window. In this window first uncheck the **Paging Enabled** check box. This action is important so make sure it's done!

Next load the program ***OSQueuesDemo*** if it's not already loaded. Create the following three processes of this program with the specified page sizes (see the table below). You can select the process page size using the drop-down list **Pages** in the **Process** tab while you are creating a process. After you create each process observe the values in the **Resource Utilisation** window and note them down in the following table:

Process	Pages	Free	Alloc	Swap
P1	4			
P2	5			
P3	3			

Comment on why P3's memory is swapped out by referring to the table above:

Now, make sure in OS Simulator the **Suspend on state change** check box is selected in RUNNING PROCESS area only. Also make sure the **CPU Speed** slider is in the fastest position. Next, click on the **START** button in the **OS Control** tab. When the message window appears indicating a change in state, make a note of the three memory utilisation values in the table below against the currently running process. Restart the OS by clicking on the **RESUME** button in **OS Control** tab. Carry on doing this for all processes indicated in the table below making a note of the three values in the corresponding columns:

Process	Free	Alloc	Swap
P1			
P2			
P3			
P1			
P2			
P3			

When you finish, uncheck the **Suspend on state change** check box and click on the **RESUME** button. As the processes start running click on the **KILL** button to stop them one by one. When all three processes terminate make a note of the following resource utilization values:

CPU %	Free	Alloc	Swap

The exercises above have been designed to demonstrate the basic principles of OS memory management. Review what has been done in the last section above and make your comments on the way virtual memory management functions in the box below:

Appendix – Program sources used in this tutorial

If you feel adventurous you can copy and paste each of the programs below in the integrated compiler's source editor and then compiling them one by one (use the **COMPILER…** button in the CPU Simulator for this). This tutorial uses the corresponding pre-compiled code downloaded. Use the **NEW** button in the compiler to create a new editor for each of the programs and compile them individually.

```
program OSQueuesDemo          %Start of program
     while true               %Forever loop
          for n = 1 to 15     %Repeat 15 times
               i = 1          %Just something to do!
          next                %End of repeat loop
          wait(3)             %Suspend program for 3 secs
     wend                     %End of forever loop
end                           %End of program


program WaitOnReadDemo
     regvar d integer         %Keep input in a register

     while true
          for n = 1 to 15     %This loop keeps CPU busy
               i = 1          %Just something to do!
          next

          write("> ")         %Display this prompt
          read(d)             %Wait for keyboard input
          writeln(d)          %Display input character
          if d = 42 then      %Test if end of program
               break          %End program if a * character
          end if
     wend
end


program ForeverLooping
     while true               %Do a forever loop
          n = 1               %Does nothing useful but
     wend                     %keeps the CPU busy
end
```