Student Name: Federico Watkins

CSIS 1800: Introduction to Computer Science and Information Systems

Chapter number: 10 Operating Systems

Assignment number: 10

1. Define the terms *multiprogramming* and *timesharing* and describe the relationship between them

|  |  |
| --- | --- |
| **Multiprogramming** | **Time sharing** |
| The technique of keeping multiple programs in main memory at the same time competing for the CPU. | A system in which CPU time is shared among multiple interactive users at the same time |
| Many processes can run at a time in a single CPU using Memory Management to keep track the programs | Working concurrently I a system where multiple users can use the computer by sharing of the computer’s time |
| It is done by scheduling the processes running in a CPU. A program running in the CPU is blocked temporarily giving processing time for other programs | Multiple users have the illusion of exclusive access to the computer via a virtual machine |
| Programs appear to be running at the same time but the CPU executes parts of each program, as determined by the Process Management, completing each one on time. | CPU executes a set of instruction for a specified amount of time and then shifts to the next process to execute its instructions for an allocated amount of time |

1. What constitutes a batch job? Show using an example.

Nowadays a batch job constitutes of a program and instructions regarding the system software and other resources needed to execute the program. It is handled without interaction between user and the program.

Example:

* Files with extension .bat in Microsoft Windows
* Processing a corporation’s monthly payroll, uses specific resources with essentially no human interaction.

1. Describe the evolution of the concept of batch processing from the human operator in the 1960s and '70s to the operating systems of today.

|  |  |
| --- | --- |
| **1960s and ‘70s** | **Systems of today** |
| Programs were put together as batch and it would be scheduled for processing when a computer was in an idle state. The system operator would take care of scheduling the batch processing and the human operator would feed them into the computer. Programs were loaded in punched cards in order to be processed. Program and instructions regarding the system software were given to the human operator | Instructions are given directly to the computer using commands from the operating system. Human operators are not needed for batch processing in modern day computer systems. Programs and instructions regarding the system software are given without interaction between user and program. |

1. If, in a fixed partition memory management system, the current value of the base register is 1024 and the current value of the bounds register is 2048, compute the physical addresses that correspond to the following logical addresses and check the validity of your results:

a. 871

b. 1115

c. 333  
d. 1025

|  |  |  |
| --- | --- | --- |
| **If(logical < register)** | **{execute}** | **Output = Physical** |
| 1. ( 871 < 2048) = true | 871 + 1024 | = 1895 |
| 1. (1115 < 2048) = true | 1115 + 1024 | = 2139 |
| 1. (333 < 2048) = true | 333 + 1024 | = 1357 |
| 1. (1025 < 2048) = true | 1025 + 1024 | = 2049 |

1. Provide detailed descriptions of:

* A process control block

The data structure used by the operating system to manage information about a process. Also known as Task control block, Process descriptor and Switch frame.

Contains the following:

* Program counter
* Memory management information
* CPU registers save area
* I/O (input/output) status information
* Processor information
* Process life cycle current state
* CPU’s Process scheduling information
* A context switch

The exchange of register information that occurs when one process is removed from the CPU and another takes its place. Also known as Task switch and Process switch. It is an essential feature of multitasking operation system.

* PCB saves current contents of register and program counter when process is moved out of CPU.
* New process in now in execution in CPU.
* Process resumes execution, during which time, the contents of the registers from this process’s PCB’s are stored.

1. Demonstrate how does the Page Table keeps track of the relationship between a Virtual Page and a Physical Frame?
   * + Example to demonstrate how the PMT keeps track of page/frame relationship:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | **Memory** | |
| **P1 PMT** | |  |  | Frame | Contents |
| Page | Frame |  |  | 0 | P2/Page2 |
| 0 | 5 |  |  | 1 |  |
| 1 | 12 |  |  | 2 | P1/Page2 |
| 2 | 2 |  |  | 3 |  |
| 3 | 7 |  |  | 4 |  |
| 4 | 22 |  |  | 5 | P1/Page0 |
|  |  |  |  | 6 |  |
|  |  |  |  | 7 | P1/Page3 |
| **P2 PMT** | |  |  | 8 |  |
| Page | Frame |  |  | 9 | P2/Page0 |
| 0 | 9 |  |  | 10 |  |
| 1 | 18 |  |  | 11 | P2/Page3 |
| 2 | 1 |  |  | 12 | P1/Page1 |
| 3 | 11 |  |  | 13 |  |
|  |  |  |  | 14 |  |
|  |  |  |  | 15 |  |
|  |  |  |  | . | . |
|  |  |  |  | . | . |
|  |  |  |  | . | . |
|  |  |  |  |  |  |

1. If, in a paged memory management system, the frame size is 4096 and the following page map table applies to the currently executing process, compute the physical addresses that correspond to the following logical addresses:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Page | 0 | 1 | 2 | 3 | 4 |
| Frame | 3 | 11 | 5 | 8 | 7 |

1. <1, 501>
2. <0, 171>
3. <3, 1048>
4. <4, 1450>
5. <2, 1025>

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Frame #** | **\*** | **Frame size** | **+** | **Offset Value** | **=** | **Physical address** |
| a | 11 | \* | 4096 | + | 501 | = | 45557 |
| b | 3 | \* | 4096 | + | 171 | = | 12459 |
| c | 8 | \* | 4096 | + | 1048 | = | 33816 |
| d | 7 | \* | 4096 | + | 1450 | = | 30122 |
| e | 5 | \* | 4096 | + | 1050 | = | 21505 |

1. **Described how does Demand Paging work**

An extension to paged memory management in which pages are brought into memory on demand, only when referenced. It works via page swap, if the page is already in memory access is complete, if not the page is brought in from secondary memory into an available frame, at which point the access is completed.

1. **Use the following table of processes:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P1 | P2 | P3 | P4 | P5 |
| Service time | 25 | 30 | 35 | 40 | 20 |

**Draw a Gantt chart that shows the completion times for each process using the following CPU scheduling:**

1. First-come, first-served
2. Shortest job next
3. Using round robin CPU scheduling with a time slice of 10.

|  |
| --- |
| Each will take (**service time / time slice**) schedules to complete its execution |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Service Time** | **/** | **Time Slice** | **=** | **Schedules** |
| P1 | 25 | / | 10 | = | 2.5 |
| P2 | 30 | / | 10 | = | 3 |
| P3 | 35 | / | 10 | = | 3.5 |
| P4 | 40 | / | 10 | = | 4 |
| P5 | 20 | / | 10 | = | 2 |

1. **Use the following table of processes:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P1 | P2 | P3 | P4 | P5 |
| Service time | 25 | 40 | 55 | 25 | 30 |

**Draw a Gantt chart that shows the completion times for each process using the following CPU scheduling:**

1. First-come, first-served
2. Shortest job next
3. Using round robin CPU scheduling with a time slice of 15.

|  |
| --- |
| Each will take (**service time / time slice**) schedules to complete its execution |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Service Time** | **/** | **Time Slice** | **=** | **Schedules** |
| P1 | 25 | / | 15 | = | 1.7 |
| P2 | 40 | / | 15 | = | 2.7 |
| P3 | 55 | / | 15 | = | 3.5 |
| P4 | 25 | / | 15 | = | 3.7 |
| P5 | 30 | / | 15 | = | 2 |

Reference:

1. Computer Science Illuminated, Nell Dale and John Lewis, Fourth Edition, Chapter 10.