**SUMMATIVE ASSIGNMENT**

**NAME:- KRISH DODHIA**

**USN NO:- 22BTRAD020**

**Q1: Explain the First, Best, and worst Fit algorithms.**

Ans:

The First Fit, Best Fit, and Worst Fit algorithms are memory allocation strategies used in computer operating systems to assign memory blocks to incoming processes or programs. These algorithms aim to efficiently utilize available memory and minimize fragmentation.

1. First Fit:

The First Fit algorithm assigns the first available memory block that is large enough to accommodate the incoming process. It starts searching from the beginning of the memory and selects the first block that meets the size requirement. This algorithm is simple and efficient in terms of time complexity since it only needs to traverse the memory once. However, it can lead to significant fragmentation, as smaller memory blocks may be left unused between allocated processes.

2. Best Fit:

The Best Fit algorithm searches the entire memory and selects the smallest available block that can accommodate the incoming process. It compares the sizes of all available memory blocks and chooses the one that minimizes waste. This approach aims to reduce fragmentation by selecting the most appropriate block size. However, it requires traversing the entire memory, making it slightly less efficient than the First Fit algorithm.

3. Worst Fit:

The Worst Fit algorithm assigns the largest available memory block to the incoming process. It searches the entire memory and selects the block with the maximum size. This strategy tends to create larger chunks of free memory, which can be useful for accommodating larger processes in the future. However, it may result in more fragmentation overall, as smaller blocks are left unused. Like the Best Fit algorithm, it requires traversing the entire memory, making it less efficient than First Fit.

**Q2: What are CLI and GUI & What are Shell and Bash?**

Ans:

CLI and GUI are two different types of user interfaces:

1. CLI (Command Line Interface):

A CLI is a text-based interface that allows users to interact with a computer system by typing commands into a command line interpreter or shell. Users input commands, and the system responds with text-based output. CLI interfaces are typically used in operating systems like Unix, Linux, and macOS, as well as in command prompt windows in Windows systems. They are efficient for experienced users who are comfortable with typing commands and navigating the file system using text-based commands.

2. GUI (Graphical User Interface):

A GUI is a visual interface that allows users to interact with a computer system through graphical elements such as windows, icons, buttons, menus, and dialog boxes. GUI interfaces provide a more user-friendly and intuitive experience, as users can interact with the system using a mouse or touch input. GUIs are commonly used in modern operating systems, such as Windows, macOS, and various desktop environments in Linux.

Now, let's discuss shells and Bash:

1. Shell:

A shell is a command line interpreter that acts as an intermediary between the user and the operating system. It reads user commands and executes them by interacting with the operating system's kernel. The shell provides features such as command execution, scripting capabilities, variable handling, control flow structures, and file manipulation. It also supports features like input/output redirection and piping to combine multiple commands. Examples of popular Unix/Linux shells include Bash (Bourne Again SHell), C Shell (csh), Korn Shell (ksh), and Z Shell (zsh).

2. Bash (Bourne Again SHell):

Bash is one of the most widely used command line shells in Unix-like operating systems, including Linux and macOS. It is an enhanced version of the original Unix Bourne shell (sh) and provides improved features and compatibility. Bash supports advanced scripting capabilities, command line editing, history, job control, environment variables, and more. It is highly customizable and extensible, allowing users to configure their command line environment according to their preferences.

**Q3: Calculate competion time, Turn around time, waiting time, and response time for processes?**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PROCESS | ARRIVAL TIME | BURST TIME | COMPLETION TIME | TURN AROUND TIME | WAITING TIME | RESPONSE TIME |
| P1 | 0 | 8 | 8 | 8 | 0 | 8 |
| P2 | 1 | 7 | 15 | 14 | 7 | 14 |
| P3 | 2 | 10 | 25 | 25 | 13 | 25 |

**Q4: Solve for bankers’ algorithm and provide a safe sequence for the processes?**

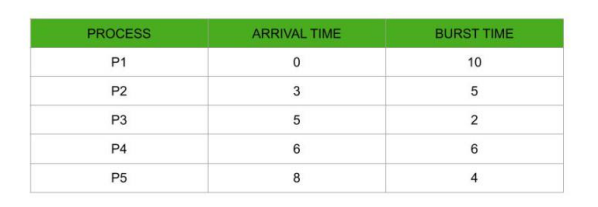
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PROCESS | ALLOCATION | | | MAX | | | AVAILABLE | | |
|  | A | B | C | A | B | C | A | B | C |
| P0 | 0 | 1 | 0 | 7 | 5 | 3 | 3 | 3 | 2 |
| P1 | 2 | 0 | 0 | 3 | 2 | 2 | 5 | 3 | 2 |
| P2 | 3 | 0 | 2 | 9 | 0 | 2 | 7 | 4 | 3 |
| P3 | 2 | 1 | 1 | 2 | 2 | 2 | 7 | 4 | 5 |
| P4 | 0 | 0 | 2 | 4 | 3 | 3 | 7 | 5 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| PRCOESS | NEED | | |
|  | A | B | C |
| P0 | 7 | 4 | 3 |
| P1 | 1 | 2 | 2 |
| P2 | 6 | 0 | 0 |
| P3 | 0 | 1 | 1 |
| P4 | 4 | 3 | 1 |

SEQUENCE OF THIS IS :-

P1 -> P3 -> P4-> P0 ->P2

**Q5: Apply FCFS and SJF on the below data and calculate Gantt chart?**



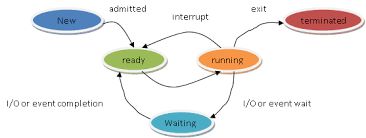
FIRST COME FIRST SERVE

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | IDLE | IDLE | P2 | IDLE | P3 | P4 | IDLE | P5 |
| 0 | 3 | 5 | 6 | 8 |
| 10 | 15 | 17 | 23 | 27 |

SHORTEST JOB FIRST

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P3 | P5 | P2 | P4 | P1 |
| 5 | 8 | 3 | 6 | 0 |
| 2 | 6 | 11 | 17 | 27 |

**Q6: Explain the process State diagram. Also, Draw the Diagram?**



A process state diagram, also known as a process lifecycle diagram, is a visual representation that illustrates the different states a process can go through during its execution in an operating system. It provides a high-level view of the various states and transitions that a process experiences throughout its lifetime.

The process state diagram typically includes the following states:

New: This is the initial state of a process. In this state, the operating system creates the process and performs necessary initialization tasks before transitioning it to the ready state.

Ready: In the ready state, the process is waiting to be assigned the CPU for execution. It is in the main memory and has all the required resources but is waiting for the scheduler to allocate processor time.

Running: When a process is allocated the CPU, it enters the running state. In this state, the process instructions are executed, and it actively utilizes the CPU for its tasks.

Blocked (or Waiting): A process transitions to the blocked state when it is unable to proceed due to the unavailability of a required resource or when it voluntarily relinquishes the CPU. The process waits for the resource or event to become available before it can proceed.

Terminated: The terminated state represents the end of a process's execution. It occurs when a process completes its tasks or is explicitly terminated by the operating system. In this state, the process is removed from the main memory, and its resources are deallocated.