**Open Book Test**

|  |  |
| --- | --- |
| Roll No. A016 | Name: Varun Khadayate |
| Program: B-Tech (CSBS) | Division: SY |
| Batch: 1 | Date of Experiment: 29-01-2021 |
| Date of Submission: 29-01-2021 | Grade: |

1. Time slicing may be a scheduling mechanism/way utilized in time-sharing systems. it's also termed as Round Robin scheduling. The aim of Round Robin scheduling or time-slicing scheduling is to allow all processes and civil rights to use CPU. during this kind of scheduling, CPU time is split into slices that are to be allocated to ready processes. Short processes could also be executed within a one-time quantum. Long processes may require several quanta.

**The duration of your time slice or Quantum**

The performance of the time-slicing policy is heavily addicted to the size/duration of the time quantum. When the time quantum is incredibly large, the Round Robin policy becomes an FCFS policy. Too short quantum causes too many process/context switches and reduces CPU efficiency. that the choice of your time quanta may be a vital design decision. Switching from one process to a different one requires a specific amount of your time to save lots of and cargo registers, update various tables and lists, etc.

Consider, as an example, process switch, or context switch takes 5 msec and time slice duration be 20 msec. Thus, the CPU should spend 5 msec on process switching again and again wasting 20% of CPU time. Let the time slice size be set to mention 500 msec and 10 processes are within the ready queue. If P1 starts executing for the primary time slice, then P2 will watch for 1/2 sec; and the waiting time for other processes will increase. the unfortunate last (P10) will anticipate 5 sec if all others use their full-time slices. To conclude setting the time slice.

• Too short will cause too many process switches and can lower CPU efficiency.

• Setting too long will cause a poor response to short interactive processes.

• A quantum around 100 msec is sometimes reasonable.

1. Scheduling will be defined as a group of policies and mechanisms which control the order within which the work to be done is completed. The scheduling program which could be a system software concerned with scheduling is termed the scheduler and also the algorithm it uses is named the scheduling algorithm. Various criteria or characteristics that help in designing a decent scheduling algorithm are:

• **CPU Utilization** − A scheduling algorithm should be designed so the CPU remains busy as possible. It should make efficient use of the CPU.

• **Throughput** − Throughput is that the amount of labor completed in an exceedingly unit of your time. In other words, throughput is that the process executed to many jobs completed in an exceedingly unit of your time. The scheduling algorithm must look to maximize the number of jobs processed per quantity.

• **time interval** − Response time is that the time taken to begin responding to the request. A scheduler must aim to reduce the latent period for interactive users.

• **turnaround** − Turnaround time refers to the time between the instant of submission of a job/ process and therefore the time of its completion. Thus how long it takes to execute a process is additionally a vital factor.

• **Waiting time** − it's the time employment waits for resource allocation when several jobs are competing in a very multiprogramming system. The aim is to attenuate the waiting time.

• **Fairness** − a decent scheduler should ensure that every process gets its justifiable share of the CPU.

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Batch Processing System** | **Real-Time Processing System** |
| 1 | In batch processing processor only needs to busy when work is assigned to it. | In real-time processing, a processor needs to very responsive and active all the time. |
| 2 | Jobs with similar requirements are batched together and run through the computer as a group. | In this system, events mostly external to the computer system are accepted and processed within certain deadlines. |
| 3 | Completion time is not critical in batch processing. | Time to complete the task is very critical in real-time |
| 4 | It provides the most economical and simplest processing method for business applications. | Complex and costly processing requires unique hardware and software to handle complex operating system programs. |
| 5 | Normal computer specifications can also work with batch processing. | Real-time processing needs high computer architecture and high hardware specification. |
| 6 | In this processing, there is no time limit. | It must handle a process within the specified time limit otherwise the system fails. |
| 7 | It is measurement oriented. | It is action or event-oriented. |
| 8 | In this system, sorting is performed before processing. | No sorting is required. |
| 9 | In this system, data is collected for a defined period and is processed in batches. | Supports random data input at a random time. |
| 10 | Examples of batch processing are transactions of credit cards, generation of bills, processing of input and output in the operating system, etc. | Examples of real-time processing are bank ATM transactions, customer services, radar systems, weather forecasts, temperature measurement, etc. |



|  |  |  |
| --- | --- | --- |
| Basis of Comparison | Time-Sharing OS | Real-Time OS |
| Basic | Emphasis on providing a quick response to a request. | It focuses on accomplishing a computational task before its specified deadline. |
| Computer resources | Shared between the users. | No sharing takes place and events are external to the system. |
| Process deals with | More than one application simultaneously. | Single application at a time. |
| Modification of the program | The programs can be modified and written by the users. | No modification is possible. |
| Response | The response is generated within the second, but there is no compulsion. | The user must get the response within the defined time constraint. |
| Switching | Takes place among the processes. | Does not present |



|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Multiprocessing** | **Multiprogramming** |
| 1 | Multiprocessing refers to the processing of multiple processes at the same time by multiple CPUs. | Multiprogramming keeps several programs in the main memory at the same time and executes them concurrently utilizing a single CPU. |
| 2 | It utilizes multiple CPUs. | It utilizes a single CPU. |
| 3 | It permits parallel processing. | Context switching takes place. |
| 4 | Less time is taken to process the jobs. | More Time is taken to process the jobs. |
| 5 | It facilitates much efficient utilization of devices of the computer system. | Less efficient than multiprocessing. |
| 6 | Usually more expensive. | Such systems are less expensive. |

1. **User Mode**

The system is in user mode when the software package is running a user application like handling a text editor. The transition from user mode to kernel mode occurs when the applying requests the help of the package or an interrupt or a supervisor call instruction occurs.

The mode bit is on the brink of 1 within the user mode. it's changed from 1 to 0 when switching from user mode to kernel mode.

**Kernel Mode**

The system starts in kernel mode when it boots and after the package is loaded, it executes applications in user mode. Some privileged instructions can only be executed in kernel mode.

These are interrupted instructions, input-output management, etc. If the privileged instructions are executed in user mode, it's illegal, and a trap is generated.

The mode bit is prepared to 0 within the kernel mode. it's changed from 0 to 1 when switching from kernel mode to user mode.

**Privileged Instruction**

The Instructions that will run only in Kernel-Mode are called Privileged Instructions.

Privileged Instructions possess the next characteristics:

* If any attempt is formed to execute a Privileged Instruction in User Mode, then it'll not be executed and treated as an illegal instruction. The Hardware traps it within the code.
* Before transferring the control to any User Program, it is the responsibility of the package to form sure that the Timer is close to interrupt. Thus, if the timer interrupts then the OS regains the control.
* Thus, any instruction which could modify the contents of the Timer is also a Privileged Instruction.
* Privileged Instructions are employed by the software package to achieve correct operation.
* Various samples of Privileged Instructions include:
  + - * + I/O instructions and Halt instructions.
        + Turn off all Interrupts.
        + Set the Timer.
        + Context Switching
        + Clear the Memory or Remove a process from the Memory.
        + Modify entries within the Device-status table.

1. For real-time systems, the package must support memory board and time-sharing fairly. For handheld systems, the software must provide memory but doesn't must provide time-sharing. Batch programming isn't necessary for both settings.
   1. What are two such problems?
      1. One user can read the private data of another user - privacy.
      2. One user can prevent another user from getting anything done - denial of service.
   2. Yes, if we can ensure that the operating system prevents any sharing of data between users, either for reading or writing, and fairly shares the computer, then we can achieve the same level of security.
2. 1. Mainframes: memory and CPU resources, storage, network bandwidth.
   2. Workstations: memory and CPU resources.
   3. Handheld computers: power consumption, memory resources.
3. A user is better off under three situations: when it is cheaper, faster, or easier. For example:
4. When the user is paying for management costs and the costs are cheaper for a time-sharing system than for a single-user computer.
5. When running a simulation or calculating that takes too long to run on a single PC or workstation.
6. When a user is traveling and doesn't have a laptop to carry around, they can connect remotely to a time-shared system and do their work.
7. A command interpreter is that the a component of a computer package that understands and executes commands that are entered interactively by a person's being or from a program. In some operating systems, the command interpreter is termed the shell.

A command interpreter usually separates the kernel because the kernel is that the central an element of an package. It manages the operations of the pc and thus the hardware - most notably memory and CPU time. There are two sorts of kernels: A microkernel, which only contains basic functionality; A monolithic kernel, which contains many device drivers.

Yes, the user can develop a replacement command interpreter using the system-call interface provided by the software package.

1. JVM interprets the bytecode instructions one at a time. Usually interpreted environments are slower than running native binaries (C, C++), because interpretation process requires converting each instruction into native machine code, every time the program runs. So, Java uses a JIT (just-in-time) compiler to make this interpretation process faster). A just-in-time (JIT) compiler compiles the bytecode for a method into native machine code the first time the method is encountered. This means that the Java program is essentially running as a native application. So, A Java program that is run by a JIT rather than a traditional interpreter typically runs much faster.
2. Examples are:
   1. A matrix multiplication where different parts of the matrix may be worked on in parallel.
   2. A webserver that services each request in a separate thread.
3. A multithreaded system comprising of multiple user-level threads cannot make use of the different processors in a multiprocessor system simultaneously. The operating system sees only a single process and will not schedule the different threads of the process on separate processors. Consequently, there is no performance benefit associated with executing multiple user-level threads on a multiprocessor system.