Daniel Maxson

9/14/2015

OS Problem set #1

Question #1

* 1-2s
  + CS-0.2s
* 2-2s
  + CS-0.2s
* 3-2s
  + CS-0.2s
* 1-2s
  + CS-0.2s
* 2-2s
  + CS-0.2s
* 3-1s P3 finishes
  + CS-0.2s scheduler
* 1-2s
  + CS-0.2s
* 2-2s
  + CS-0.2s
* 1-2s
  + CS-0.2s
* 2-2s
  + CS-0.2s
* 1-2s P1 finishes
  + CS-0.2s scheduler
* 2-0.5s P2 finishes

Total time 23.7s

Context switching time = 2.2s

Overhead 2.2/23.7 = 0.093 0.093\*100% = 9.3%

Question #2

Multiprogrammed OS

* P1-10s
  + CS-0.2s
* P2-8.5s
  + CS-0.2s
* P3-3s

Total time 21.9s

Context switching time = 0.4s

Overhead 0.4/21.9 = 0.018, 0.018\*100%= 1.8%

Here batch mode finishes faster. It requires less context switching but more wait time for program 2 and 3, avg 6.3ms, actual 10.2 and 18.9 respectively. Multitasked could be faster if program 1 is waiting for I/O or has had an error and isn’t required to give up CPU time.

Question #3

* Abort - Forcibly terminating a program, usually illegal operations or values and hardware errors
* Fault - potentially recoverable error, page faults stack overflow. Can result in processor state loss
* Interrupt - I/O completion signal
* Trap - a software-generated interrupt cause by a user request or an error

A software interrupt requests a switch to kernel mode so that a user application can request an operating system service be performed.

A hardware interrupt is used to signal that a requested hardware operation is complete.

Question #4

The jump table contains a pointer to the table of system calls, it is used to lookup / find the system call.

Question #5

Overlapping allows the CPU to delegate hardware I/O to the hardware PU and continue processing instead of remaining idle and waiting for hardware I/O to complete. The two ways that I/O can be overlapped are DMA and direct I/O. Direct I/O allows hardware to go out on its own and complete requests and then signal that the request is ready to be processed and handled by the CPU. Direct Memory Access allows hardware to take requests and once completed put the information directly into main memory and passes an interrupt to the CPU

Question #6

1. Call write()
2. OS puts write() put on wait queue
3. OS write() scheduled
4. OS write() sent to disk driver
5. write() received by disk driver
6. write() scheduled on disk
7. write() executed on disk by device controller
8. device controller sends interrupt that I/O write() has been completed
9. device handler receives interrupt and acknowledges
10. device handler interrupts device driver
11. device driver receives interrupt and acknowledges
12. device driver interrupts OS
13. OS receives interrupt and acknowledges
14. OS moves write() from wait queue to ready queue