PROCESS UTILIZATION

Read one record from file 15µs

Execute 100 instructions 1µs

Write one record to file 15µs

Total= 31µs

Percent CPU utilization = 1/31= 0.032 = 3.2%

TROUGHPUT

This is a measure of how much work is being performed. It depends on the average length of a process but is also influenced by scheduling policy.

RESPONSE TIME

This is the time from the submission of a request until the response begins to be received. It is a measure for the user’s perspective. Turnaround time is the machines perspective. You want low response time to maximize the number of I/O you can do. Uniprogramming has a longer mean response time than multiprogramming. This is because everything has to be done sequentially.

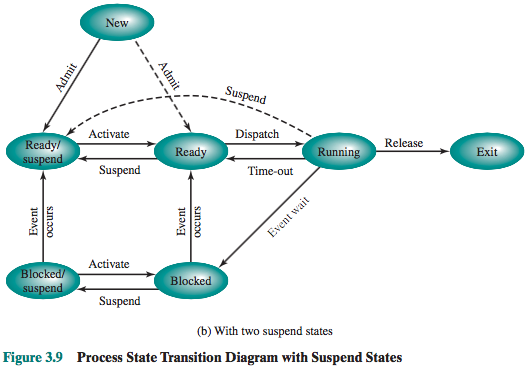
FIVE STATE PROCESS MODEL

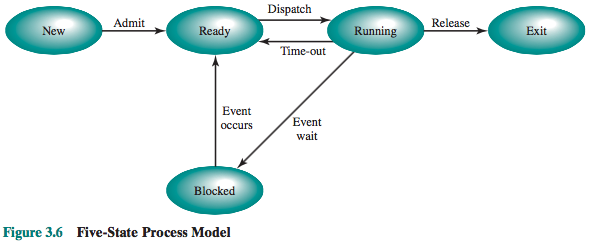
Running: the process is being executed

Ready: a process is prepared to execute

Blocked/Waiting: A process that can’t execute until some event occurs, ex. I/O

New: A new process take a second to get from main memory to the process pool

Exit: A released process, halted or aborted.

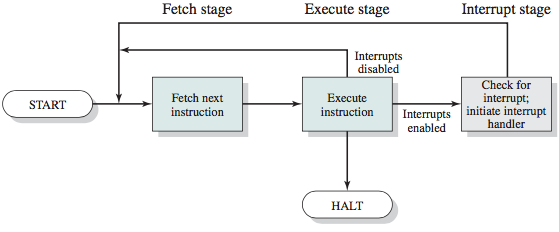
SEVEN STATE PROCESS MODEL

Blocked/Suspend: Process is in secondary memory

Ready/Suspend: In secondary memory but is available for execution

INTERUPTS

Program, Timer, I/O, Hardware failure



MEMORY HIERARCHY

-Inboard memory -Outboard storage

Registers magnetic disk

Cache CDs,DVDs,BR

Main memory

-Offline storage

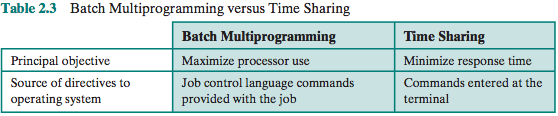
Magnetic tape

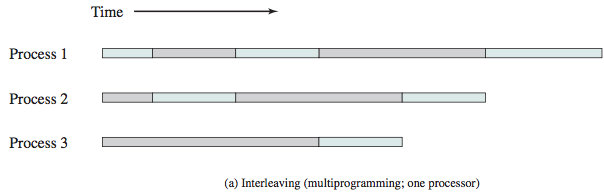
If accessed word is found faster in memory that's a hit, a miss occurs if the accessed word is not found in faster memory.

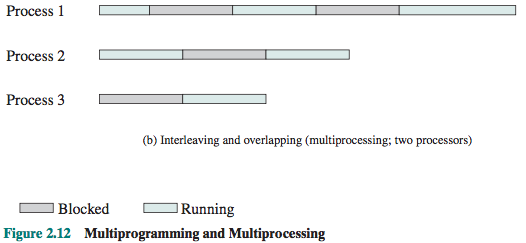
DMA, INTERUPT DRIVEN I/O vs PROGRAMMED I/O

DMA is sent by the processor to go grab blocks of memory and let it know when it has them.

The overall effect is to cause the processor to execute more slowly during a DMA transfer when processor access to the bus is required. Still a multiple-word I/O transfer DMA is more efficient than interrupt of programmed I/O







FIRST FIT is the most likely the best placement algo

PROCESS vs THREAD

Process: A collection of one or more threads and associated system *resources*, ex. A program in execution

Thread: A dispatchable unit of work. Includes processor context(PC and Stack pointer). They are executed sequentially and can be interrupted.

PROCESS IMAGE

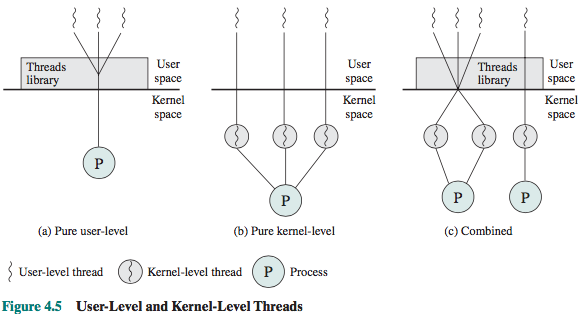
Contains user data(program data), User program, Stack, and a Process control block

EXECUTION CONTEXT

Or processor state is the internal data by which the OS is able to supervise and control the process.

KERNEL MODE vs USER MODE

Kernel mode- Privileged instructions may be executed and protected areas of memory can be accessed

User mode- less access

MODE SWITCHING

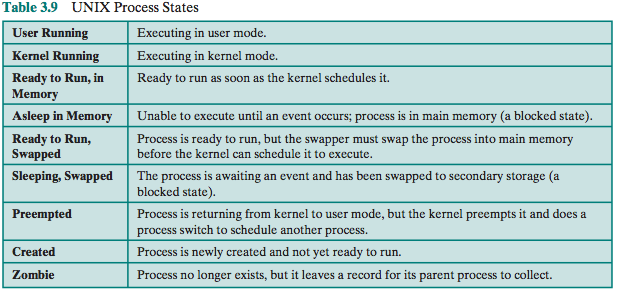
Sets program counter to the starting address of an interrupt handler program

Switches from user mode to kernel mode

SWAPPING

Involves moving part or all of a process from main memory to disk. When none of the processes in main memory is in the Ready state, the OS swaps one of the blocked processes out on to disk into a suspended queue. This is a queue of existing processes. Swapping is an I/O operation.

DISPATCHER vs SCHEDULER

Dispatcher- a program that switches the processor from one processes to another

Scheduler- schedules processes

ULT vs KLT vs LWP

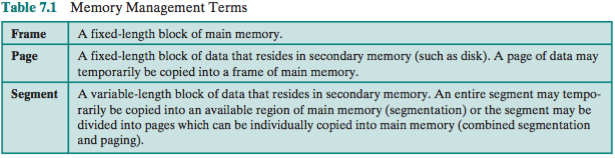
User Level Threads- akk if the work of thread management is done by the application and the kernel is not aware of the existence

Kernel Level Threads- all of the work of thread management is done by the kernel

Light Weight Process- are a mapping between ULTs and kernel threads. Each LWP supports ULT and maps to one kernel thread. They can be executed in parallel on multiprocessors.

JACKETING

A way to overcome the problem of blocking threads. It converts a blocking system call into a non blocking system call.



HIT RATIO

(Hit Rate)(Hit Time) + (Miss Rate)(Miss Time) = AVG

ex. (0.9)(20ns)+(0.1)((0.6)(60ns+20ns)+(0.4)(12,000,080ns)

