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%

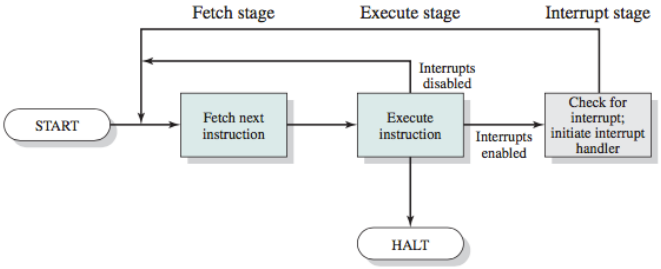
TROUGHPUT
This is a measure of how much work is being performed. It depends on the average length of a process but is also influenced by scheduling policy.

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
Registers
Cache
Main memory
-Outboard storage
magnetic disk
CDs,DVDs,BR
-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

Table 2.3 Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

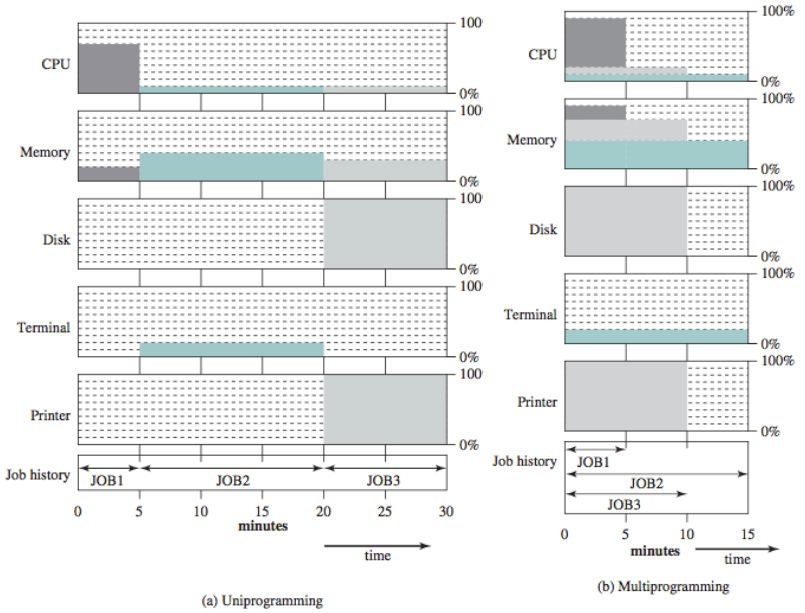


Figure 2.6 Utilization Histograms

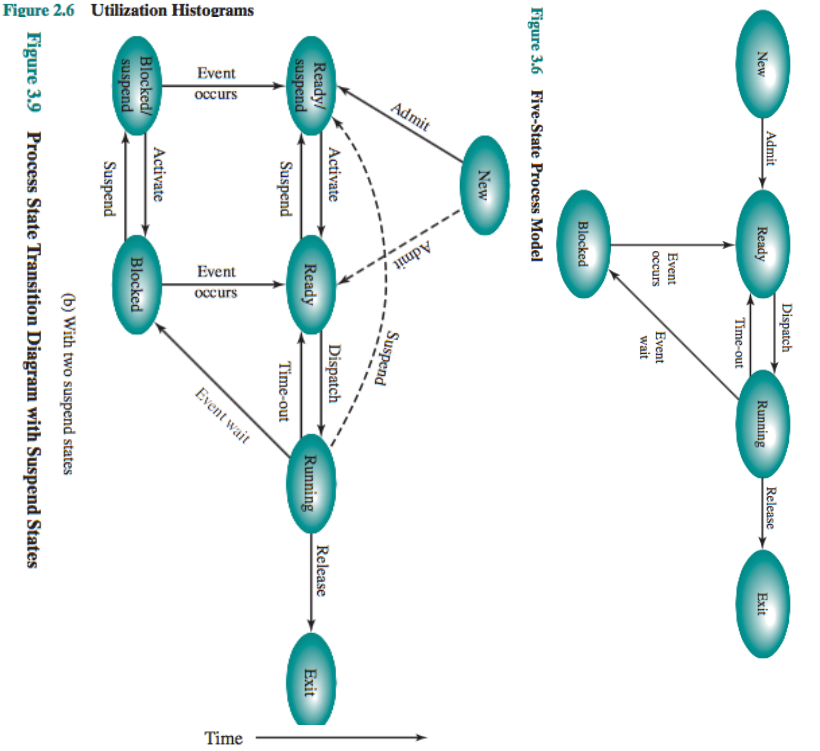


Figure 3.9 Process State Transition Diagram with Suspend States

Figure 3.6 Five-State Process Model

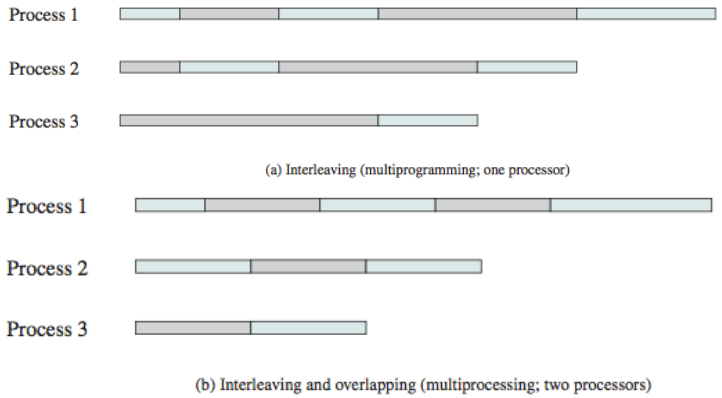


Figure 2.12 Multiprogramming and Multiprocessing

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.

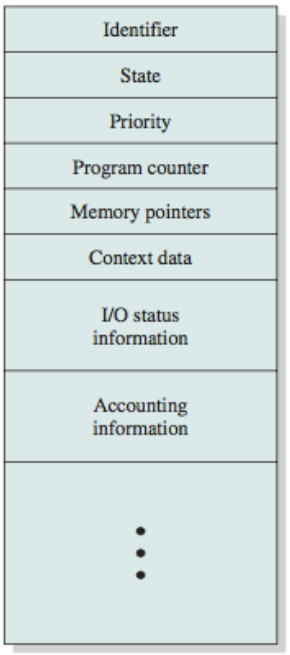


Figure 3.1 Simplified Process Control Block

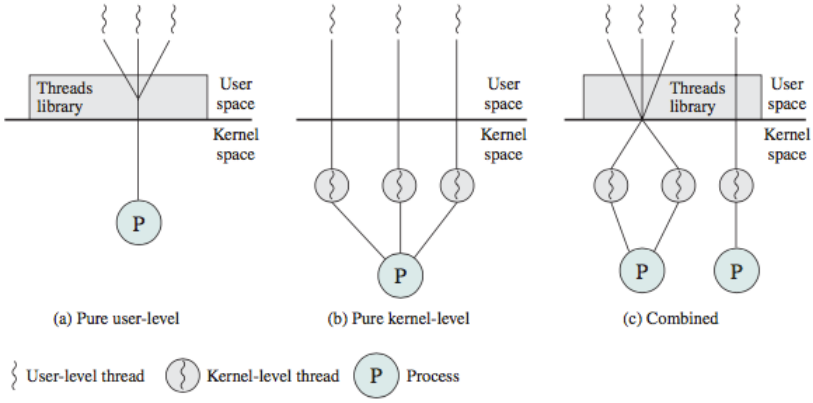


Figure 4.5 User-Level and Kernel-Level Threads

Table 3.9 UNIX Process States

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.