#opcodes = 2^4=16
#pos = 2^12=4096
#sizeOfPos = 16bits=2bytes
#sizeOfMemory=#pos * sizeOfPos
#memoryRange = 000-FFF
#DataRange=0000-FFFF

#PC=3octalDigits=log2(8)=3bits*3=9 bits
-PC is length of mem address!!!
#IR = opcode + address
#IR=Data=memPos=12bits
#pos = 2^9 = 512
#sizeOfAPosition=12bits
#SizeOfMemory= #pos * #sizeOfAPosition
#memoryRange=000-777
#DataRange=0000-7777

Process Control (continued)

Modes of Execution: *User mode (less-privileged mode) -user programs typically execute in this mode. *System mode (more privileged) - also referred to as control mode or kernel mode - kernel of the OS.

KERNEL functions - *Process management (pocess creation,termination,switching,synchronization, mangement of proc. control block). *Memory management (allocation of address space to process, swapping, page/segment management). *I/O mangement (buffer management, Allocation of I/O channels and devices to process). *Support functions (interrupt handling, Accounting, Monitoring).

Process creation (step-by-step): 1) OS assigns a unique process ID to the new process 2) allocates space for the process. 3) initializes the process control block. 4) sets the appropriate linkages 5)creates or expands other data structures.

Process Switching - a process switch may occur anytime that the OS has gained control from the currently running process. Events giving OS control are *interrupt(reaction an asynchronus external event) *Trap(Halding of an error or an exception condition) *Supervisor call (call to an operating system function)

System Interrupt: *due to some sort of event that is external to and independent of the currently running process. *clock interrupt *I/O interrupt *memory fault *time slice - the max amount of time that a process can execute before being interrupted.

Trap: *an error or exception condition generated within the currently running process *OS determines if the condition is fatal - moved to the exit state and a process switch occurs *action will depend on the nature of the error.

If no interrups are pending the Proccessor: 1) proceeds to the fetch stage and fetches the next instruction of the current program in the current process.

If interrupt is pending: 1) sets the program counter to the starting address of an interrupt handler program. 2) switch from user mode to kernel mode so that the interrupt processing code may include priviliged instructions.

Change Of Processor State: 1) save the context of the proc 2) update the proces control block of the process current running. 3) move the process control block of this process to the appropriate que. 4) select another process for exe. 5) update the process control block of the process selected. 6) update memory management data structures. 7) restore the context of the processor to that which existed at the time the selected process was last switched out.

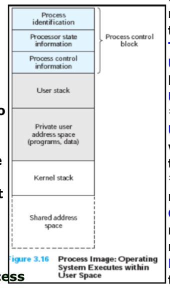


Figure 3.16 Process Image: Operating System Executes within User Space

Security Issues: * An OS associates a set of privileges with each process *typically a process that exe on behalf of a user has the privileges that the OS recognizes for that user. *Highest lvl of prvilige is referred to as admin, supervisor, root.

System Access Threats: *Intruders (hacker) *Malicious Software. Intrusion Detection counter measure: Intrusion detection system(IDS_ comprises of three loical compondents 1)sensors 2)analyzers 3)user interface *IDS are designed to detect human intruder behavior.

Authentication Countermeasure: cosists of two steps 1) Identification 2) verification

Access Control Countermeasure: *implements a security policy that specifies who or what may have access to each specific system resource and the type of access that is permitted in each instance. *mediates between a user and system resource. *A security admin maintains an authorization database. *auditing function montors and keeps a record of user accesses to system resources.

FIREWALL: dedicated computer that: *interfaces with comps outside the network. *has special security precations built into it to protect sensitive files on computers within the network. **Design of a firewall: *all traffic must pass through the firewall. * only authorized traffic will be allowed to pass. *immune to penetration.

Fork() process creation: 1) allocate a slot in the process table for the new process. 2) assign a unique process ID to the child process. 3) Make a copy of the process image of the parent, with the exception of any shared memory. 4) increments counters for any files owned by the parent, to reflect that an additional process now also owns those files. 5) Assings the child process to the ready to run state. 6) Returns the ID number of the child to the parent process, and a 0 value to the child process.

After fork() process creation: the kernel can do: 1) stay in the parent process. 2) transfer control to the child process. 3) transfer control to another process.

Processes and Threads - *have two characteristics. 1) Resource ownership - process includes a virtual address space to hold the process image. *the OS performs a protection function to prevent unwanted interference between the processes with respect to resources. 2) Schedule/-Execution *a process has an execution state (running, ready, etc.)and a dispatching priority and is scheduled and dispatched by the OS.

Threads: *The unit of dispatching is referred to as thread or lightweight process *unit of resource ownership is reffered to as a process or task.
*In an OS that supports threads, scheduling and dispatching is done on a thread basis.
*Most of the state information dealing with execution is maintained in thread-level data structures.

Multithreading - the ability of an OS to support multiple, concurrent paths of execution within a single process.

Single Thread Approach: * a single thread of excution per process, in which the concept of a thread is not recognized, is referred to as a single-threaded approach. *MS-DOS is an example.

Mutlithreaded approaches: *A java run-time environment is an example of a system of one process with multiple threads.

Processes: *The unit of resource allocation and a unit of protection. *A virtual address space that holds the process image. *Protected access to: -processors, -other processes, -files, -I/O resources.

One or more threads in a process: [each thread has]: *an execution thats (running, ready,etc), *saved thread context when not running, *an execution stack, *some per-thread static storage for local variables, *access to the memory and resources of its process(all threads of a process share this).

Benefits of Threads: 1) takes less time to create a new thread vs. new process. 2) less time to terminate a thread than a process. 3) switching between threads takes less time than switching between processes. 4) threads enhance efficiency in communication between programs.

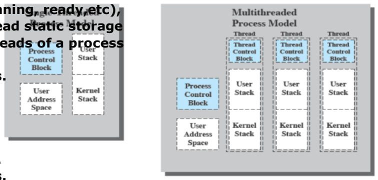


Figure 4.2 Single Threaded and Multithreaded Process Models

Thread use in a single-user system: *foreground and background work, *asynchronous processing, *speed of execution, *modular program structure.

Suspending a process/thread: *involves suspending all threads of the process. *termination of a process terminates all threads within the process.

Thread execution state: *key states(running, ready, blocked), *Thread operations associated with a change in thread state are: 1)spawn 2) block 3) unblock 4) finish.

thread syncrhonization: *It is necessary to synchronize the activities of the various threads. *all threads of a process share the same address space and other resources. *any alternation of a resource by one thread affects the other threads in the same process.

Types of threads: 1) Upper level thread(ULT) 2)Kernel level thread(KLT)

ULT Upper level threads: *all thread management is done by the application. *the kernel is not aware of the existence of threads.

ULT Advantages: *thread switching does not require kernel mode privileges, *scheduling can be application specific, *ULTs can run on any OS.

ULT Disadvantages: *in a typical OS many system calls are blocking -as a result, when a ULT executes a system call, not only is that thread blocked, but all of the threads within the process are blocked.
*In pure ULT stategy, a multithreaded application cannot take advantage of multiprocessing.

Overcoming ULT disadvantages: *Jacketing -converts a blocking system call into a non-blocking system call. *writing an application as multiple proceses rather than multiple threads.

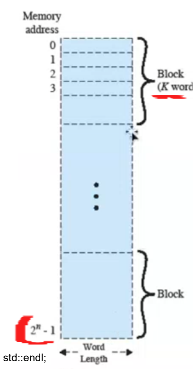
Kernel-Level Threads(KLTs): *Thread management is done by the kernel. *no thread management is done by the application., *Windows is an example of this approach.

KLT Advantages: *The kernel can simultaneously schedule multiple threads from the same process on multiple processors. *If one thread in a process is blocked, the kernel can schedule another thread of the same process. *Kernel routines can be multithreaded.

KLT Disadvantages: The trasfer of control from one thread to another within the same process requires a mode switch to the kernel.

Combined Approaches: *Thread creation is done in the user space. *Bulk of scheduling and synchronization of threads is by the application, *Solaris is an example.

Applications that Benefit: *Multithreaded native applications -characterized by having a small number of highly threaded processes. *Multiprocess applications -characterized by the presence of many single-threaded process. *Java Applications., *Multiinstance applications -multiple instances of the application in parpallel.



(b) Main memory

Blocks in Main Memory = 2 ^ n / K

```
struct operation *pos_ptr = (struct operation *)pos_void_ptr;

    if(pthread_create(&tid[i], NULL, calculator, &operations[i]))

for (int i = 0; i < NOPER; i++)
    pthread_join(tid[i], NULL);
```

```
#include <iostream>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main()
{
   pid_t pid;
   std::cout << "I am the parent process" << std::endl;
   for(int i=0;i<3;i++)
   {
      pid = fork();
      if (pid == 0)
      {
         std::cout << "I am the child process " << i << std::endl;
         if (i==1)
         {
            pid = fork();
            if (pid == 0)
            {
               std::cout << "I am a grandchild process from child process " << i << std::endl;
               _exit(0);
            }
            wait(0);
         }
         _exit(0);
      }
      wait(0);
   }
   return 0;
}
```

Threads:Processes	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:N	Combines attributes of M:1 and 1:M cases.	TRIX

Table 4.2 Relationship between Threads and Processes

Two levels of memory(cache/main)
Hit Ratio: The probability of a word being found in the first level of memory(fastest)
Miss Ratio: 1 - Hit ration

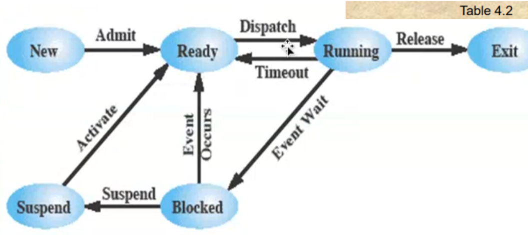
Principle of Locality (cache)- when you transfer information from second level to first level, you will transfer a block of words where that block is. This is because the probability of another word we are looking for is higher. Code tends to be sequential. Increases performance

PERFORMANCE OF A SIMPLE TWO-LEVEL MEMORY

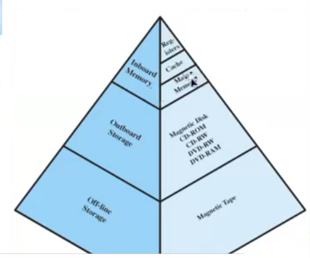
Access time Level 1 (TL_1) = 100 ms
Access time Level 2 (TL_2) = 1000 ms
Hit Ratio = 90 %
Miss Ratio = 10%
Average Access time (AvgT) = HR * TL1 + MR * (TL2+TL1)
= 0.9 * 100 ms + 0.1 * (1000 ms + 100 ms)

```
#include<iostream>
#include<unistd.h>
#include<sys/wait.h>
using namespace std;
int main()
{
    pid_t pid;
    int i = 0;
    for(i;i<3;i++){
        pid=fork();
        if(pid==0){
          break;
        }
        wait(NULL);
    }
    if(i == 0 && pid == 0){
        for(int j=0;j<2;j++){
            pid=fork();
            if(pid == 0){
                break;
            }
            wait(NULL);
        }
    }

    if(i == 2 && pid == 0){
        for(int j = 0; j<2; j++){
            pid = fork();
            if(pid == 0){
                break;
            }
            wait(NULL);
        }
```



(a) With One Suspend State



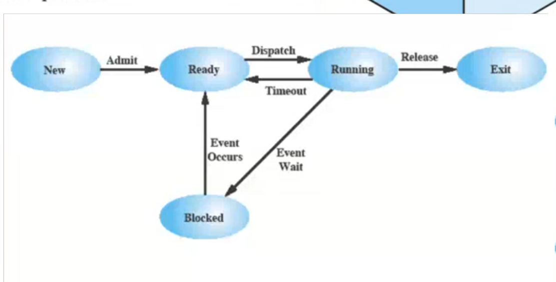
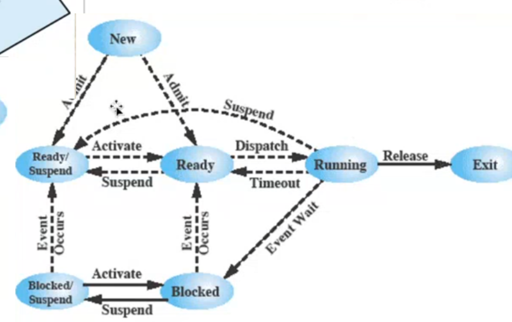
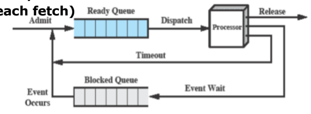


Figure 3.6 Five-State Process Model



(b) With Two Suspend States

*Operating system -exploits hardware resources of one or more processors, most complex pieces of software
-provides a set of services to system users, manages second. mem and I/O dev.
*pc - holds address of the instruction to be fetched next (incr. after each fetch)
PROCESSOR
IR - instruction being executed.
MAR -
MBR -
I/O AR -
I/O BR -
*AC (accumulator) - temporary storage
microprocessor - fastest general purpose proc. proc on single chip, multiprocessor, each chip cont. cores
GPU - provide efficient computation or arrays, physic simulation, comp. on large spreadsheet.
*basic instruc cycle - start -> fetch inst -> execute inst. (repeat if needed) -> halt
*Interupts - interupt the normal sequencing of a proc
provided to improve proc utilization (i/o devices are slower than the proc)
*classes of interrupt(Program, Timer, I/o, hardware failure)
*single interrupt process - dev. controller/sys harw issues inter -> proc finishes current instruc -> proc ack. interrupt -> proc pushes PSW and PC onto cntrl stack -> proc loads new PC val from inter -> save remainder of proc state info -> process interrupt-> restore proc state info -> restore old PSW and PC
*Multiple interrupts - can be done sequentially or nested


(a) Single blocked queue

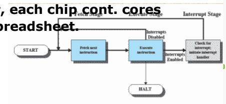
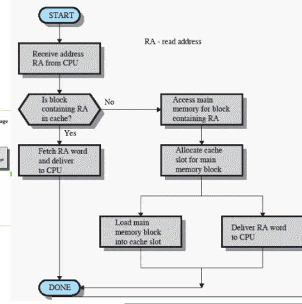


Figure 1.7 Instruction Cycle with Interrupts



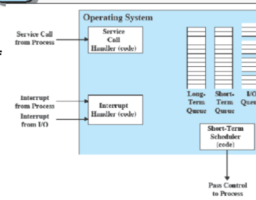
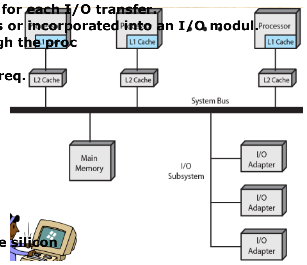


Figure 2.11 Key Elements of an Operating System for Multiprogramming

*MEMORY HIERARCHY major constraints (amount, speed, expense), memory must keep up with proc, cost of mem must be reasonable to other components
*down pyrimid (decreasing cost per bit, increasing cap., increasing access time, decr. freq. of acc. to mem by proc.)
*PRINCIPLE OF LOCALITY - 1) memory references by the proc. tend to cluster 2) data is org. so that the percentage of accesses to each successively lower level is less than that of the lvl above 3) can be applied to more than 2 lvl's
*CACHE 1) invisible to OS. 2) interracts with mem mang. hardware. 3) Proc. must access mem. at least once per instr. cycle. 4) exploits principle of locality with small, fast mem.
*Cache Princ. 1) contains copy of portion of main mem. 2) Proc first checks cache 3) not found? block of mem is read into cache. 4) locality of ref- it is likely that many of future mem. references will be other bytes in block.
MAPPING FUNCTION - Determines which cache location the block will occupy
*Replacement Algorithm - Least recently used (LRU) Algo (effective strat to replace a block that has been in the cache the longest with no refer. *hardware mechanisms are needed to identify the least recently used block *chooses which block to replace when a new block is to be loaded into cache)
*I/O Techniques - When the proc encounters an instr. relating to I/O, it exe that instr. by issuing a comm to the appropriate I/O module
*PROGRAMMED I/O - 1) the I/O module performs the requested action then sets the appropriate bits in the I/O status reg. 2) The Proc periodically checks the status of the I/O module until it determines the instruc. is complete. 3)With programmed i/O the performance level of the entire systm is severly degraded
Interrupt Driven I/O drawbacks - transfer rate is limited by the speed with which the proc can test/service a device. 2)The proc is tied up in managing an I/O transf. - number of instruc. must be exe for each I/O transfer.
DIRECT MEMORY ACCESS (DMA) - performed by a seperate module on the sys bus or incorporated into an I/O modul
1) tranfers the entire block of data directly to and from mem without going through the proc
-proc is involved only at the beg. and end of transfer
- proc exe more slowly during a transfer when proc access to the bus is req.
2) more efficiant than interrupt-driven I/O
Symmetric Muti-Proc (SMP) - stand-alone computer systm with
-two or more similar processors of comparable capabil.
-procesors share the same main mem and are interconected by bus
-proc share access to I/O devices
-all proccessors can perform the same functions
-system is controlled by intergrated
OS that provides interaction between proccessors and their programs
at the job, task, file, data element levels.
Multi-core comp (chip multi-proc) - combines two or more proc(cores) on a single silicon
-each core consists of all components of an indepen. proc.
-multicore chips include L2/L3 cache.



OPERATING SYSTEM - program that controls the execution of applcation programs, interface between application and hardware.
*Services OS - program devl., program exe, access i/o devices, controlled access to files, system access, error detection and response.
Key Interfaces *Instruction set arch. (ISA), application binary interf. (ABI)
Application programming interface(API)
Role of OS - computer is a set of resources for the movement, storage, and proc. of data, the OS is responsible for managing these resources.
Operating system as software 1) functions in teh same way as ordinary comp software. 2) program, or suite of programs, executed by the proc. 3) frequently relinquishes control and must depend on the proc. to allow it to regain control.

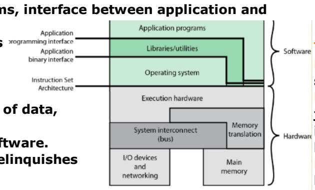


Figure 2.1 Computer Hardware and Software Infrastructure

Serial processing - (earlist computers, no OS, programs interacted directly with hardware, ran with display lights, toggle switches, printer, users had to interact with the computer in 'series')PROBLEMS: scheduling - time allocations could run short or long, wasted computer time. set up-time - a considerable amount of time was spent on just setting up the program to run.
Simple Batch system - early computers were expensive, important to maximize proc utilization. *Monitor - user no longer has direct access to processor., job is submitting to computer operator who batches them together and palces them on an input device. Program branches back to the montior when finished.
Monitor Point of view - monitor controsl the sequence of events,, *resident monitor software always in memory, *monitor reads in jobs and gives control, * job returns control to monitor.
Processor POV - *Proc exe instructions from the mem containing the monitor, *Exe the instructions in the user program until it enxounters an ending or ERR condition., *"Control is passed to a job" means proc is fetching and exe instructions in a user program., *"control is returned to the monitor" means that the proc is fetching and exe instructions from the monitor program.

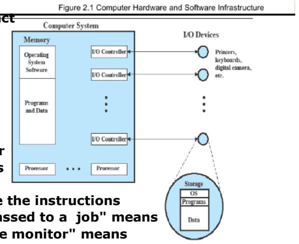


Figure 2.2 The Operating System as Resource Manager

Simple Batch System Overhead - *Proc alternates between execution of user program and execution of the monitor.
*Sacrifices: 1) some main mem is now given over to the monitor, 2) some proc time is consumed by the monitor.
*Despite overhead, the simple batch system improves utilization of the comp.
Uniprogramming - The proc spends a certain amount of time exe, until it reaches an I/O instruction; it must then wait until that I/O instruction concludes before proceeding.
Multprogramming - *there must be enough memory to hold the OS (resident monitor) and one user program, * When one job needs to wait for I/O, the processor cans witch to the other job, which is likely not waiting for I/O *also known as multitasking, *mem is expanded to hold three, four, or more programs and switch among all of them.

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Figure 2.4 System Utilization Example

- Processor is often idle
  - even with automatic job sequencing
  - I/O devices are slow compared to processor

Time-sharing system- *can be used to handle multiple interactive jobs, *Proc time is shared among multiple users. *Mult. users simultaneously access the system through terminals, with the OS interleaving the exe of each user program in a short burst or quantum of computation.

	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

Process - fundamental to the structure of operating systems.
Development of the process - Three major lines of computer system dev. created problems in timing and synchronization that contributed to the dev.
Casues of error: 1)improper synchronization - a program must wait until the data are available in a buffer, improper design of the signaling mechanism can result in loss or duplication.
2) failed mutual exclusion - more than one user or program attempts to make user of a shared resource at the same time. *only one routine at a time allowed to perform an update against the file.
3) nondeterminate program operation - program exe is interleaved by the proc when mem is shared. *the order in whcih programs are scheduled may affect their outcome
4) Deadlocks - it is possible for two or more programs to be hung up waiting for each other. *may depend on the chance timing of resource allocation and release.
Components of a process: *a process contains three components 1) an executable program 2) the associated data needed by the program (variables, work space, buffers) 3) The execution context 'Process State' of the program
The execution context: *is the internal data by which the OS is able to supervise and control the process, * includes the contents of the various process registers, *includes information such as the priority of the process and whether the process is waiting for the completion of a particular I/O event.
Process management: *The entire state of the process at any instant is contained in its context, * New features can be designed and incorporated into the OS by expanding the context to include any new information needed to support the feature.
Memory Management: the OS has five principle storage management responsibilities
Virtual memory: *a facility that allows programs to address memory from logical point of view, without regard to the amount of main memory physically available. *conceived to meet the requirement of having multiple user jobs reside in main memory concurrently.
Paging: *Allows processes to be comprised of a number of fixed-size blocks. *Program references a word by means of virtual address (consists of a page number and an offset with the page), (each page my be located anywhere in main mem.). *Provides for a dynamic mapping between the virtual address used in the program and a real address in main mem.

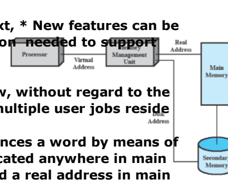
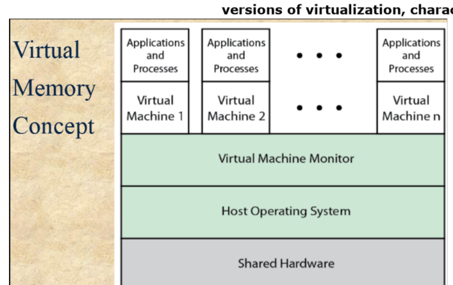
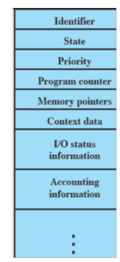


Figure 2.10 Virtual Memory Addressing

Information Protection and Security: the nature of the threat that concerns an organization will vary greatly depending on the circumstances. *The problem involves controlling access to computer systems and the information stored in them.
Scheduling and Resource Management: *key responsibility of an OS is managing resources *Resource allocation policies must consider.
Different Architectural Aproaches: Demands on operating systems require new ways of organizing the OS.
MicroKernel Architecture: *assigns only a few essential functions to the kernel:
Multithreading: *technique in which a process, executing an application, is divided into threads that can run concurrently.
Symmetric Multiprocessing(SMP): *term that refers to a computer hardware arch. and also to the OS behavior that exploits that arch. *several processes can run in parallel. *multiple processors are transparent to the user. *these processors share the same main memory and I/O facilities. *all processors can perform the same functions.*The OS takes care of scheduling of threads or processes on individual processors and of synchronization among processors.
OS Design: [Distributed OS] *provides the illusion of 1) single main memory space 2) single secondary memory space 3) unified access facilities. *State of the art for distributed operating systems lags that of uniprocessor and SMP operating systems.
[Object-oriented Design] *used for adding modular extensions to a small kernel. *enables programmers to customize an operating system without disrupting system integrity. *eases the development of distributed toos and full-blown distributed operating systems.
Virtual Machine and Virtualization: enables a PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS. *a machine can host numerous application, including those that run on different operating systems, on a single platform. *host operating system can support a number virtual machines.
*each has the characteristics of a particular and, in some versions of virtualization, characteristics of a particular hardware.





OS Management of Application Execution: *resources made avail. to mul. appli. *The proc is switched among mul. appli. so all will apear to be progressing. *the proc and I/O devices can be used effeciently.
Process Control block: *contains the process elements. *it is possible to interrupt a running process and later resume exe as if the interrupt had not occured. *key tool that allows support for multiple proc.
Two state process model: state 1) running state2) not running.
Reasons for Process creation: 1)new batch job 2) interactive logon 3) created by OS to provide a service 4) spawned by existing process.
Process termination: *there must be a means for a process to indicate its completion. *a batch job should include HALT instruction or an explicit OS service call for termination. *for an interactive application, the action of the user will indicate when the process is completed (e.g. logoff, quitting an application).
Reasons for process termination: 1)normal completion 2)time limit exceeding 3) memory unav 4) bounds viol 5)protection err 6) arithmetic err 7) I/O failure 8) parent termination
Suspended Processes: *Swapping - involves moving part of 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 on of the blocked process out on to disk into a suspend queue.
Characteristics of a Suspended Process: *the process is not immediately available for exe. *The process may or may not be waiting on an event. *The process was placed in a suspended state by an agent: either itself, a parent process, or the OS, for the purpose of preventing its exe. *the process may not be removed from this state until the agent explicitly orders the removal.
Reasons for suspended process: 1) swapping, 2) Os may suspend a background or utility 3) interactive user request 4) timing, may be executed periodically 5) parent process request, parent may wish to suspend exe of a descendent to modify.
OS CONTROL TABLES
Memory Tables: *used to keep track of both main and secodary memory. *Processes are maintained on secondary memory using some sort of virtual memory or simple swapping mechanism. Must include *allocation of main mem to process *allocation seconndary mem to process *protection attributes of blocks of main or virtual mem. *information needed to manage virtual mem.
I/O tables: *used by the OS to manage the I/O devices and channels of the computer system. *at any given time, an I/O device may be available or assigned to a particular process.
File Tables: *information may be maintained and used by a file management system in which case the OS has little or no knowledge of files. *In other OS, much of the detail of file mangement is manged by the OS itself. *existence of files *location of secondary mem *current status
Process Tables: Must be maintained to manage processes. *there must be some reference to memory, I/O, and files, directly or indirectly. *The tables themselves must be accessible by the OS and therefore are subject to memory management.
PROCESS CONTROL STRUCTURE *OS must know *where the process is located *The attributes of the process that are necessary for its management.
Process Control Location: Process location - *process must include a program to be exe, *process will consist of at least sufficient memory to hold the programs and data of the process. *the execution of a program typically involves a stack that is used to keep track of procedure calls and parameter passing between procedures.
Process Attributes: each process has associated with it a number of attrributes controlled by the OS. *The collection of program, data, stack, attributes is referred to as process image. (location depends on memory management scheme being used.)
Process Image: 1) user data (modifiable part of user space,stack, etc.) 2) User program 3) stack(each process has one or more LIFO stacks assoctated. stack is used to store parameters and calling addresses for system calls.) 4) Process control block (data needed by the OS to control the process.
) Process Identification: *each process is assigned a unique numberic ID, or there must be a mapping that allows the OS to locate the appropriate tables based on the process ID. *many of the tables controlled by the OS may use process ID to cross-reference process tables. *Memory tables may be organized to provide a map of main memory. *When processes communicate with each other the process ID informs the OS of the destination of a particular communication. *when processes are allowed to create other processes, identifiers indicate the parent and descendents.
Processor State information: Consists of the contents of the process registers *user-visible registers *control and status registers *stack pointers
Program Status Word (PSW): contains condition codes plus other status information. *EFLAGS register is an example of a psw used by any OS running on an x86 proc.
Process Control information: the additional info neeed by the OS to control and coordinate the various active processes.
Process Control Block: *most important data structure in an OS *contains all of the information about a process that is needed by the OS. *blocks are read and/or modified by virually every module in the OS. *Defines the state of the OS. *Difficulty is not access but protection. *a bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected process. *A design change in the structure or semantics of the Process control blockcould affect a number of modules in the OS.