IMPLEMENTATION OF PROCESS

Implementation of Process:

Operating system maintains a table (an array of structure) known as process table with one entry per process to implement the process. The entry contains detail about the process such as, *process state, program counter, stack pointer, memory allocation, the status of its open files, its accounting information, scheduling information and everything else* about the process that must be saved when the process is switched from running to ready or blocked state, so that it can be restarted later as if it had never been stopped.

Process Management	**Memory Management**	**File management**
Register program counter Program status word stack pointer process state priority Scheduling parameter Process ID parent process process group Signals Time when process started CPU time used Children's CPU time Time of next alarm	Pointer to text segment pinter to data segment pinter to stack segment	Root directory Working directory File descriptors USER ID GROUP ID

Each I/O device class is associated with a location (often near the bottom of the memory) called the ***Interrupt Vector.*** It contains the address of interrupt service procedure. Suppose that user process 3 is running when a disk interrupt occurs. User process 3's program counter, program status word and possibly one or more registers are pushed onto the (current) stack by the interrupt hardware. The computer then jumps to the address specified in the disk interrupt vector. That is all the hardware does. From here on, it is up to the software in particular the interrupt service procedure.

Interrupt handling and scheduling are summarized below.

1. Hardware stack program counter etc.
2. Hardware loads new program counter from interrupt vector
3. Assembly languages procedures save registers.
4. Assembly language procedures sets up new stack.
5. C interrupt service runs typically reads and buffer input.
6. Scheduler decides which process is to run next.
7. C procedures returns to the assembly code.
8. Assembly language procedures starts up new current process.

Fig:The above points lists the Skeleton of what the lowest level of the operating system does when an interrupt occurs.

Context Switching:

Switching the CPU to another process requires performing a state save of the current process and a state restore restore of a different process. This task is known as context switch. When a context switch occurs, the kernel saves the the context of the old process in its PCB and loads the saved context of the new process scheduled to run. Context switch time is pure overhead, because the system does no useful work while switching. Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied etc.

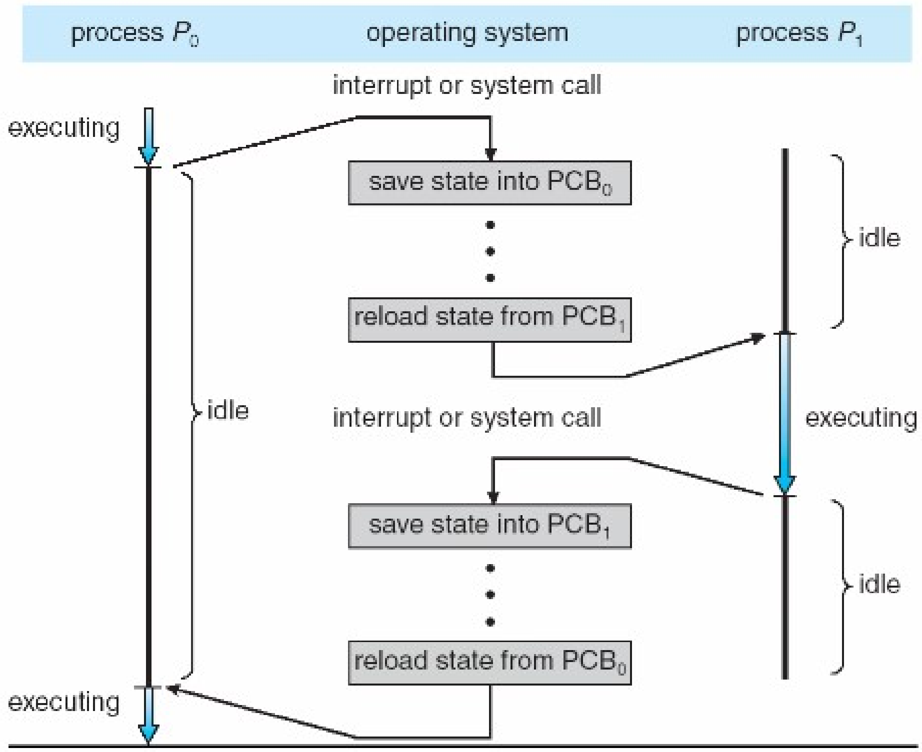

Fig: CPU switch from one process to another

Threads:

A thread is a basic unit of CPU utilization, it comprises a thread ID, a program counter, a register set, and a stack. It shares with other threads belonging to the same process its code section, data section, and other operating system resources, such as open files and signals. A traditional (or heavy weight) process has a single thread of control. If a process has multiple thread of control, it can perform more than one task at a time. Fig below illustrate the difference between single threaded process and a multi-threaded process.

Thread simply enable us to split up a program into logically separate pieces and have the pieces run independently of one another until they need to communicate. In a sense threads are a further level of object orientation for multitasking system.

Multithreading:

Many software package that run on modern desktop pcs are multi-threaded. An application is implemented as a separate process with several threads of control. A web browser might have one thread to display images or text while other thread retrieves data from the network. A word-processor may have a thread for displaying graphics, another thread for reading the character entered by user through the keyboard, and a third thread for performing spelling and grammar checking in the background.