Process

By

Amar Panchal

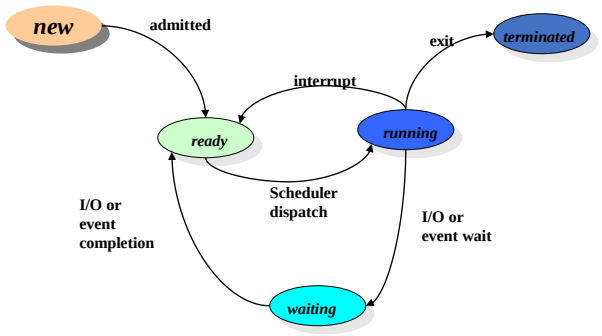
Process Concept

- An operating system executes a variety of programs
  - batch systems - jobs
  - time-shared systems - user programs or tasks
  - job and program used interchangeably
- Process - a program in execution
  - process execution proceeds in a sequential fashion
- A process contains
  - program counter, stack and data section

Amar Panchal - 9821601163

Process State

- A process changes state as it executes.

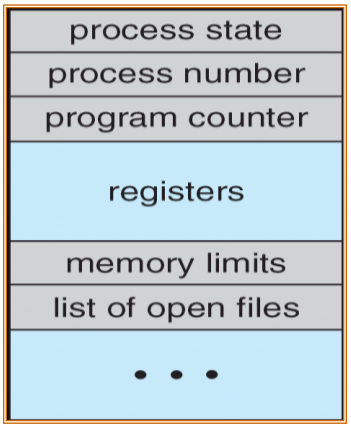


Process States

- New - The process is being created.
- Running - Instructions are being executed.
- Waiting - Waiting for some event to occur.
- Ready - Waiting to be assigned to a processor.
- Terminated - Process has finished execution.

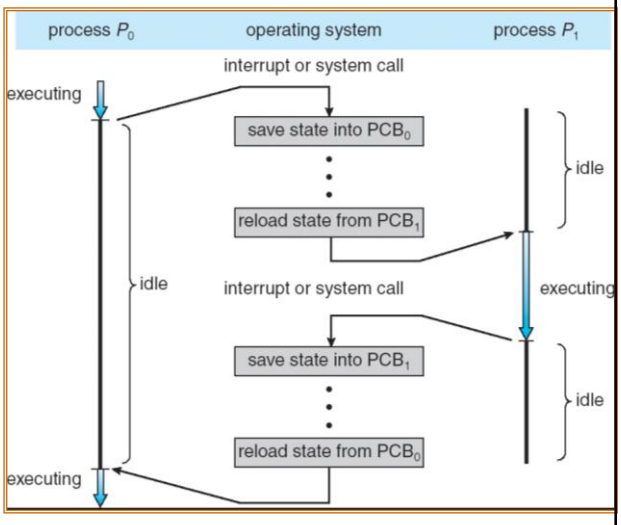
Process Control Block

- **Contains information associated with each process**
  - **Process State - e.g. new, ready, running etc.**
  - **Process Number – Process ID**
  - **Program Counter - address of next instruction to be executed**
  - **CPU registers - general purpose registers, stack pointer etc.**
  - **CPU scheduling information - process priority, pointer**
  - **Memory Management information - base/limit information**
  - **Accounting information - time limits, process number**
    - **I/O Status information - list of I/O devices allocated**



Process Control Block

CPU Switch From Process to Process

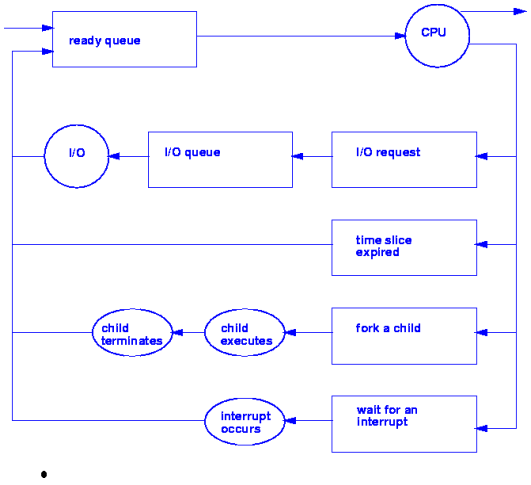


- Code executed in kernel above is overhead
  - Overhead sets minimum practical switching time

Process Scheduling

Process (PCB) moves from queue to queue

When does it move? Where? A scheduling decision



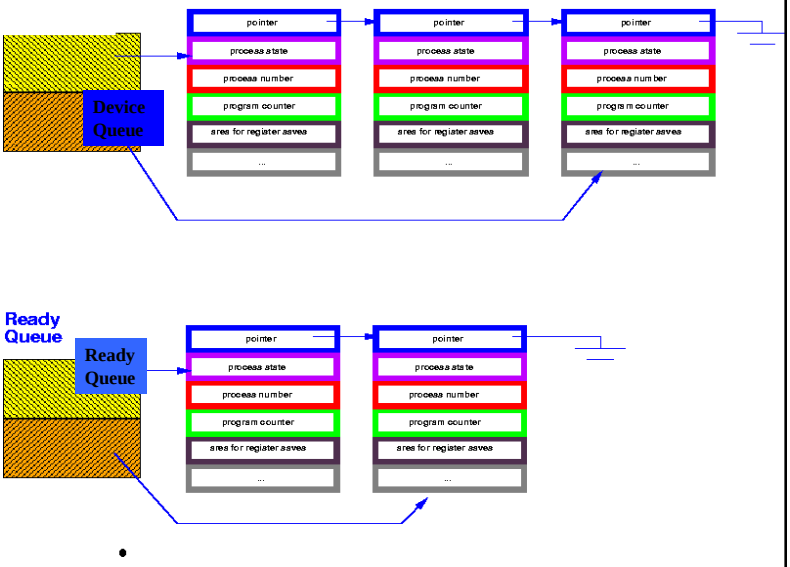
Amar Panchal - 9821601163

Process Scheduling Queues

- Job Queue - set of all processes in the system
- Ready Queue - set of all processes residing in main memory, ready and waiting to execute.
- Device Queues - set of processes waiting for an I/O device.
- Process migration between the various queues.
- Queue Structures - typically linked list, circular list etc.

Amar Panchal - 9821601163

Process Queues



Amar Panchal - 9821601163

Schedulers

- Long-term scheduler (or job scheduler) -
  - selects which processes should be brought into the ready queue.
  - invoked very infrequently (seconds, minutes); may be slow.
  - controls the degree of multiprogramming
- Short term scheduler (or CPU scheduler)
  - selects which process should execute next and allocates CPU.
  - invoked very frequently (milliseconds) - must be very fast
- Medium Term Scheduler
  - swaps out process temporarily
  - balances load for better throughput

Amar Panchal - 9821601163

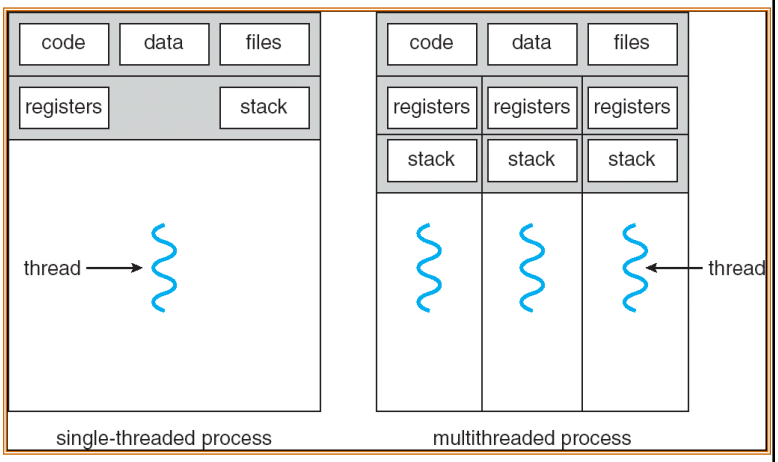
Process Creation

- Resource sharing
  - Parent and children share all resources.
  - Children share subset of parent's resources - prevents many processes from overloading the system.
  - Parent and children share no resources.
- Execution
  - Parent and child execute concurrently.
  - Parent waits until child has terminated.
- Address Space
  - Child process is duplicate of parent process.
  - Child process has a program loaded into it.

Threads

- Processes do not share resources well
  - high context switching overhead
- Idea: Separate concurrency from protection
- Multithreading: *a single program made up of a number of different concurrent activities*
- A thread (or lightweight process)
  - basic unit of CPU utilization; it consists of:
    - program counter, register set and stack space
  - A thread shares the following with peer threads:
    - code section, data section and OS resources (open files, signals)
    - No protection between threads
  - Collectively called a task.
- Heavyweight process is a task with one thread.

Single and Multithreaded Processes



- Threads encapsulate concurrency: "Active" component
- Address spaces encapsulate protection: "Passive" part

Amar Panchal - 9821601163

Benefits

- Responsiveness
- Resource Sharing
- Economy
- Utilization of MP Architectures

Amar Panchal - 9821601163

Threads(Cont.)

- In a multiple threaded task, while one server thread is blocked and waiting, a second thread in the same task can run.
  - Cooperation of multiple threads in the same job confers higher throughput and improved performance.
  - Applications that require sharing a common buffer (i.e. producer-consumer) benefit from thread utilization.
- Threads provide a mechanism that allows sequential processes to make blocking system calls while also achieving parallelism.

Examples: Multithreaded programs

- **Embedded systems**
  - Elevators, Planes, Medical systems, Wristwatches
  - Single Program, concurrent operations
- **Most modern OS kernels**
  - Internally concurrent because have to deal with concurrent requests by multiple users
  - But no protection needed within kernel
- **Database Servers**
  - Access to shared data by many concurrent users
  - Also background utility processing must be done

More Examples: Multithreaded programs

- **Network Servers**
  - Concurrent requests from network
  - Again, single program, multiple concurrent operations
  - File server, Web server, and airline reservation systems
- **Parallel Programming (More than one physical CPU)**
  - Split program into multiple threads for parallelism
  - This is called Multiprocessing

Types of Threads

- Kernel-supported threads
- User-level threads
- Hybrid approach implements both user-level and kernel-supported threads (Solaris 2).

Kernel Threads

- Supported by the Kernel
  - Native threads supported directly by the kernel
  - Every thread can run or block independently
  - One process may have several threads waiting on different things
- Downside of kernel threads: a bit expensive
  - Need to make a crossing into kernel mode to schedule
- Examples
  - Windows XP/2000, Solaris, Linux,Tru64 UNIX, Mac OS X, Mach, OS/2

User Threads

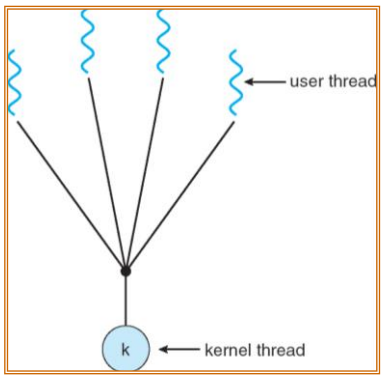
- Supported above the kernel, via a set of library calls at the user level.
  - Thread management done by user-level threads library
    - User program provides scheduler and thread package
  - May have several user threads per kernel thread
  - User threads may be scheduled non-premptively relative to each other (only switch on yield())
- Advantages
  - Cheap, Fast
    - Threads do not need to call OS and cause interrupts to kernel
- Disadv: If kernel is single threaded, system call from any thread can block the entire task.
- Example thread libraries:
  - POSIX Pthreads, Win32 threads, Java threads

Multithreading Models

- Many-to-One
- One-to-One
- Many-to-Many

Many-to-One

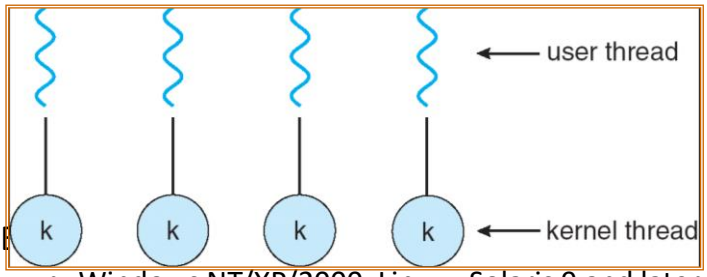
- Many user-level threads mapped to single kernel thread
- Examples:
  - Solaris Green Threads
  - GNU Portable Threads



Amar Panchal - 9821601163

One-to-One

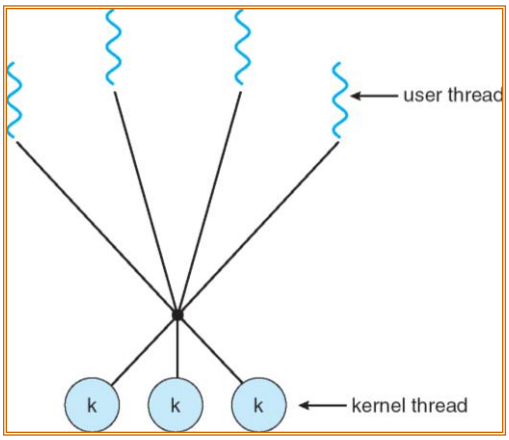
- Each user-level thread maps to kernel thread



- Windows NT/XP/2000; Linux; Solaris 9 and later

Many-to-Many Model

- Allows many user level threads to be mapped to many kernel threads
- Allows the operating system to create a sufficient number of kernel threads
- Solaris prior to version 9
- Windows NT/2000 with the *ThreadFiber* package



PROCESS SCHEDULING

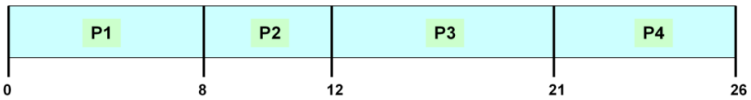
- **Scheduling criteria**
  - a) CPU utilization: Keep the CPU as busy as possible
  - b) Through put: One measure of work of CPU is the number of processes that are completed per time unit called through put.
  - c) Turnaround time: The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU and doing I/O.
  - d) Waiting time: It is the sum of the periods spent waiting in the ready queue.
  - e) Response time: The time it takes for the process to start responding but is not the time it takes to output the response.

Scheduling algorithms

- First Come First Serve:

EXAMPLE DATA:

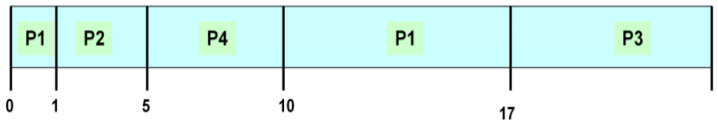
Process	Arrival Time	Service Time
1	0	8
2	1	4
3	2	9
4	3	5



Average wait = ((8-0) + (12-1) + (21-2) + (26-3))/4 = 61/4 = 15.25

EXAMPLE DATA:

Process	Arrival Time	Service Time
1	0	8
2	1	4
3	2	9
4	3	5

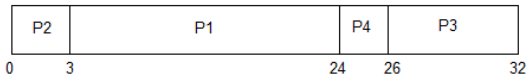


Average wait = ((5-1) + (10-3) + (17-0) + (26-2))/4 = 52/4 = 13.0

Amar Panchal - 9821601163

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,



The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

Amar Panchal - 9821601163

- Round Robin

Process	Duration	Order	Arrival Time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Suppose time quantum is 1 unit.

P1	P2	P3	P1	P2	P3	P1	P2	P3	P2

0 10

P1 waiting time : 4

P2 waiting time: 6

P3 waiting time: 6

The average waiting time(AWT) : (4+6+6)/3=5.33

Problem

Analyze scheduling algorithms for the following five processes given the process CPU burst time, priority (lower number means higher priority) and arrival time.

Processes	Burst Time	Priority	Arrival Time
P1	14	3	0
P2	7	2	2
P3	12	1	4
P4	4	5	6
P5	16	4	8

Problem

(15) a. Draw 3 gantt charts that illustrate the execution of these processes using the following scheduling algorithms: SJF (nonpreemptive), preemptive priority (low numbers are the highest), and RR (quantum=4).
b. What is the turnaround time and average waiting time for each algorithm?

Process	Arrival Time	Burst Time	Priority
P1	0	3	3
P2	2	5	2
P3	5	10	4
P4	7	1	1
P5	10	3	3

Amar Panchal - 9821601163

Problem

6.17 The following processes are being scheduled using a preemptive, round-robin scheduling algorithm. Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an ***idle task*** (which consumes no CPU resources and is identified as P_{idle}). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

Thread	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

a. Show the scheduling order of the processes using a Gantt chart.
b. What is the turnaround time for each process?
c. What is the waiting time for each process?

Amar Panchal - 9821601163