PROCESS

• Multiprogramming – computer can do several things at the same time
  • Ex: While running a user program, a computer can also be reading from a disk and outputting text to a screen or printer
  • the CPU is running only one program, in the course of 1 second, it may work on several programs, thus giving the users the illusion of parallelism
• Multiprocessor – true hardware parallelism, which have two or more CPUs sharing the same physical memory

PROCESS

Process – a running program
- a process is an activity of some kind. It has a program, input, output, and a state.
- A single processor may be shared among several processes, with some scheduling algorithm being used to determine when to stop work on one process and service a different one.

PROCESS VS PROGRAM

• The analogy:
  • Program – course schedule, test schedule, books
  • Process – learning and teaching activities in class

PROCESS AND OS

• OS schedules and sends processes to be executed by CPU.
• OS allocates resources for processes.

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
**Process State**

A process changes state as it executes.

- **new**
- **admitted**
- **running**
- **waiting**
- **I/O or event completion**
- **I/O or event wait**
- **interrupt**
- **exit**
- **terminated**

**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.
- 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 (PCB)**

<table>
<thead>
<tr>
<th>Pointer</th>
<th>Process state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Process number</td>
</tr>
<tr>
<td></td>
<td>Program counter</td>
</tr>
<tr>
<td>Registers</td>
<td>Memory limits</td>
</tr>
<tr>
<td></td>
<td>List of open files</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**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.

**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.
### Process Profiles

- **I/O bound process** - spends more time in I/O, short CPU bursts, CPU underutilized.
- **CPU bound process** - spends more time doing computations; few very long CPU bursts, I/O underutilized.

**The right job mix:**
- Long term scheduler - admits jobs to keep load balanced between I/O and CPU bound processes.

### Context Switch

- **Task that switches CPU from one process to another process**
  - the CPU must save the PCB state of the old process and load the saved PCB state of the new process.

- **Time for context switch is dependent on hardware support (1-1000 microseconds).**

### Process Creation

- Processes are created and deleted dynamically.
- Process which creates another process is called a **parent** process; the created process is called a **child** process.
- Result is a tree of processes: e.g. UNIX - processes have dependencies and form a hierarchy.
- Resources required when creating process:
  - CPU time, files, memory, I/O devices etc.

### 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.

### UNIX Process Creation

- Fork system call creates new processes.
- execve system call is used after a fork to replace the processes memory space with a new program.

### Process Termination

- Process executes last statement and asks the operating system to delete it (**exit**).
  - Output data from child to parent (via wait).
  - Process’ resources are deallocated by operating system.
  - Parent may terminate execution of child processes.
    - Child has exceeded allocated resources.
    - Task assigned to child is no longer required.
    - Parent is exiting.
    - OS does not allow child to continue if parent terminates.
    - Cascading termination.