Resident Monitor

Interrupt may be viewed as follow:

```
IRQ = False //interrupt request is set to false
. . .
While (halt flag not set during execution)
{   IR = (PC);
    PC++;
    execute(IR)
}
```

Resident Monitor

Interrupt may be viewed as follow:

```
IRQ = False
. . .
While (halt flag not set during execution)
{   IR = (PC);
    PC++;
    execute(IR)
    If (IRQ)
    {   .
        .
        .   . .
        .
    }
}
```

Resident Monitor

Interrupt may be viewed as follow:

```
IRQ = True
. . .
While (halt flag not set during execution)
{   IR = (PC);
    PC++;
    execute(IR)
    If (IRQ)
    {   "Context Switching" take place

    }
}
```

Resident Monitor

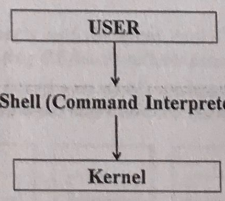
Interrupt may be viewed as follow:

```
IRQ = True
. . .
While (halt flag not set during execution)
{  IR = (PC);
   PC++;
   execute(IR)
   If (IRQ)
   { // Save the current status of the CPU
        Save_All_Registers(); // save in a designated area: "OldState"
     //handle the interrupt
        Restore_PC(IV(IRQ#)); // by loading the "interrupt  response"
        Execute the response
        Restore_PC(OldState) //let the CPU continues with the
                             //execution of the interrupted process
     }
   }
```

Linux Kernel Components

Process management Resource Management File Management,	Device Driver Interrupt Handler	Modules

Linux Kernel Components



OS Evolution

6- Time Sharing
7- Multi-User
8- Multi-programming
   - Spooler (Simultaneous Peripheral Operation On-Line)
   - HASP (Houston Automatic Spooling Program)
9- Real Time
   - Hard
   - Soft

OS Evolution

10- PC Systems
11- Coupled Systems
   - Tightly Coupled
     - *Multi-Processor Systems (Parallel Systems)*
   - Loosely Coupled
     - *Clustered Systems*
     - *Distributed Systems*

OS Evolution

Parallel Systems (Example of Tightly Coupled systems)

Cheaper than equivalent multiple single-processor systems *because they can share*

Peripherals

Mass-Storage, and

Power Supplies

More reliable (Fault Tolerant)

Higher throughput

OS Evolution

Parallel Systems Mode of Operation:

Symmetric

Asymmetric

OS Evolution

Parallel Systems Mode of Operation:

Symmetric

CPUs are peers and, therefore, each CPU performs all tasks including OS functions, user processes, etc.

Each CPU has its own set of registers and cache memory. However they share the RAM.

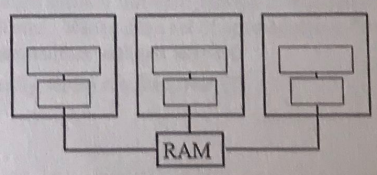
OS Evolution

Parallel Systems Mode of Operation:

Symmetric

CPUs are peers and, therefore, each CPU performs all tasks including OS functions user processes, etc.

Each CPU has its own set of registers and cache memory. However they share the RAM.



OS Evolution

Parallel Systems Mode of Operation:

Asymmetric (Employer-Employee Technique)

CPUs have Employer-Employee relationship.

One CPU in role of "employer" and the rest in role of "employees"

Employer controls the entire system. And it schedules and allocates work to the employee CPUs.

Each employee either looks to the employer for instruction or it has a predefined task

OS Evolution

Parallel Systems New Trends:

Classic Technology (Multiprocessor systems) :

Multiple chips with a single CPU per chip.

Recent Trend in CPU design (Multicore systems)

Single chip with multiple cores.

Core is a CPU with basic functionality, its own registers and cache memory

OS Evolution

Parallel Systems New Trends:

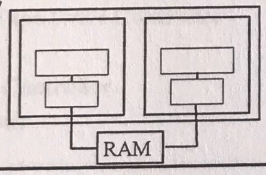
Classic Technology (Multiprocessor systems) :

Multiple chips with a single CPU per chip.

Recent Trend in CPU design (Multicore systems)

Single chip with multiple cores.

Core is a CPU with basic functionality, its own registers and cache memory



Dual-core design with two cores placed on the same chip

OS Evolution

Clustered Systems (Example of Loosely coupled systems)

Two or more individual systems joined together they share storage and closely linked via a LAN or a faster interconnect (InfiniBand).

OS Evolution

Clustered Systems *Modes of Operation*

Symmetric Clustering

a. Each node works on its own task.

b. Nodes monitor each other

Asymmetric Clustering

a. One node is in "Hot-Standby mode" and its function is to monitor the other nodes.

b. Other nodes work on their tasks. If a node fails then the node in "Hot-Standby mode" assumes the task of the failed node.

OS Evolution

An example of Clustered Systems is BEOWULF Clusters

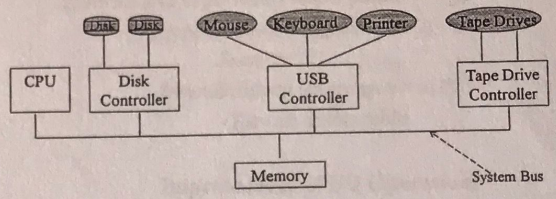
a. It is a cluster of the commodity hardware (some Beowulf clusters are built from discarded personal computers.)
b. Connected via a simple LAN
c. No Single specific software package is required to construct the cluster . Nodes use a set of open-source software libraries to communicate with one another.
d. Typically, nodes run the Linux OS.

OS Evolution

12- Client-Server Systems
13- Handheld Systems

Chapter 2

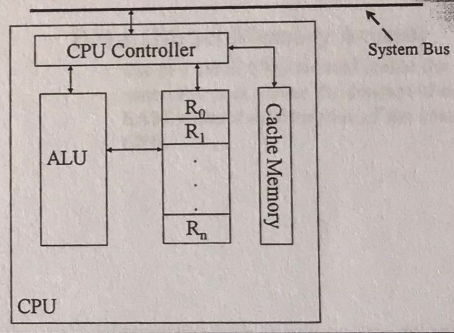
Computer System Architecture



Chapter 2

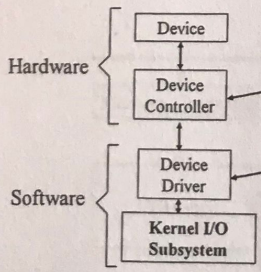
Computer System Architecture
- CPU
- I/O devices
- I/O Device Controllers
  - Disk Controller
  - Printer Controller
  - Tape Drive Controller
- Memory
- Memory Controller
- System Bus

CPU Structure



Tape Drives

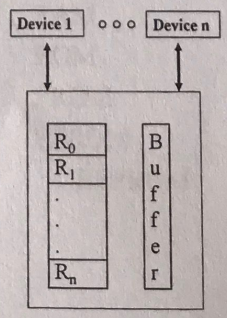
I/O Structure



It permits connection of I/O devices to a computer. One controller per kind.

It is internally custom tailored to specific device But externally presents a uniform interface for the device to the rest of the OS. The interface is per kind

I/O Structure



Device controller Structure

- Device controller moves data from buffer to I/O devices under its control and vice versa .
- Moving data from buffer or into buffer is a command that is expressed by the contents of the registers.