# "UNIT - IV"

## "Kernel Architecture"

The Linux Kernel has the ability to load & unload arbitrary section of Kernel mode on demand. These loadable modules have the full access to all the h/w capabilities of m/c on which they run.

| System mngmt. prg | User Processes | User utility program | Compiler |
|---|---|---|---|
| System shared libraries | | | |
| Linux Kernel | | | |
| Loadable Kernel Modules | | | |

Linux Components - :.

1) Module Management System
   Allow module to be loaded into m/m & to communicate with the rest of Kernel
2) Module loader & unloader
3) Driver Registration System
   New driver availability
4) Conflict Resolution Mechanism
   Reserve h/w resource

(1) Module Management.

Linux have an internal symbol table in Kernel If the system utility cannot resolve any reference in the module by looking them up in Kernel's symbol table, then the module is rejected.

Firstly, the module under utility ask the Kernel to reserve a continuous area

of virtual kernel m/m for the module. The kernel then returns the address, loader utility can use this address to relocate module's m/c code to the correct loading address. A second call then passes the module plus any symbol table that new module wants to export. The module itself is now copied into previously allocated space & the table gets updated.

With a Module requester, the Kernel will inform the management process whenever a process request a device driver, file system etc. The manager process regularly queries the kernel to see whether a dynamically loaded module is still in use & unloads the module.

## 7) Driver Registration.

The module tells its functionalities to the Kernel, the Kernel maintain a dynamic table of all known drivers & provide a set of routines to allow drivers to be added or removed. Kernel makes sure to call module's startup routine & cleanup routine before loading & unloading.

Registration table includes :-

- Device drivers

Include character devices (Printers, terminals etc), block devices (disk drives) network & interface devices.

- File Systems
- Network Protocols (IPX).
- Binary Format

(3) Conflict Resolution
- Prevent modules crashing for h/w
- Prevent Autoprobes
  Auto detect devices
- Resolve conflict among multiple drives trying to access the same h/w.

Kernel have list of allocated h/w resources.

int

"PROCESS"

A process is a basic context within which all user request is serviced within the operating system.

Fork() - Create a process

If a parent process wishes to modify the environment in which a new program is to be run, it can fork & then, still running original program in child process

(a) P-Identity.
It includes P-Id, Credentials, Personality.

(b) Process Environment
Process environment is inherited from its parent & is composed of 2 null-terminated vectors
- Argument Vector
- Environment Vector

Argument vector used to invoke the running program list Command line argument

Environment variable is a list of "Name = Value" pairs.

**Process Context** :- P_Id & environment are ~~created~~ set up when process is created, & not changed until that process exist. A process may choose to change some aspects of its identity if it needs to do so, process context, is the state of running program at any one time, it change constantly.

(i) **Scheduling Context**
The info that the scheduler needs to suspend & restrat the process.

(ii) **Accounting**
Maintain info. of resources being used & total resources used in lifetime

(iii) **File Table**
It is an array of pointers to Kernel File structures.

(iii) **File System Context**
Current root & default directories to be used for next searches are stored here.

(iv) **Signal Handler Table**
Define routine & address space to be called when specific signal arrives

(v) **Virtual M/m Context.**

**Process & Threads** :-
The internal representation of a process & thread is same.

A thread is just a new process that happen to share the same address space as its parents. The major difference arises when a new thread is created by a clone system call.

Fork create new process having its own new process context,

Clone creates new process having its own identity but share the d.s. of parents

## "MEMORY MANAGEMENT"

— Allocate & free physical memory, pages etc
— Virtual m/m handeling which is m/m mapped into the address space.

(i) Physical Memory.

Page allocator.

Use a buddy-heap algo to track physical pages

(ii) Virtual Memory

Maintain address space visible to each process. It also watch page swapping

(a) Swapping & Paging.

Page out policy uses standard-clock algos. Every page is adjusted on each pass of clock. Frequent accessed pages have higher age value. This allow the pager to select pages on LFU policy.

(6) Kernel Virtual Memory

The first section is a static area that contain page-table, helps to do physical to virtual address translation.

(iii) Execution & Loading of User Program

## "File System"

Unix file can be anything capable of handling the i/p or o/p of a stream of data. Devices drivers & interprocess communication also are files

(i) Virtual File System

VFS is designed Object Oriented Principles, It has 2 components ——:

→ A set of definitions about a file object to look like
→ A layer of s/w to ~~map~~ manipulate those objects.

Three main object types ——:

Inode Object
File objects
File System Object.

VFS define a set of operation that must be implemented
Function Table list the address of the actual functions that implement those operations

(ii) The Linux ext2fs File System.

Linux was originally programmed with a Minix-compatible file system, but it was severly restricted by 14-character file name limits & 64 MB size.

Minix file system was superseded by new file system, extended file system. Second fs had improved & covered missing features.

(iii) Linux Proc File System
It is flexible enough, process file system is an example whose contents are not actually stored anywhere but are computed on demand according to user file I/O requests. Linux implements such proc file system, but extends it greatly by adding a no. of extra directories & text files. under file system's root directory.

## "I/O MANAGEMENT"

(i) Block Devices
(ii) Character Device
(iii) Network Device

(i) Block devices — Random access to completely independent, fixed sizes blocks of data
Eg:- Hard disk, CD ROM, Floppies.
It is used to store file systems, but direct access is allowed so that programs can create & repair the file system that device contains.

(ii) Character Device have most other devices.
Eg:- Loudspeaker allow data to be written to it,

but reading not possible. Magnetic tape device allow to seek data from middle of file but mouse can't do it.

(3) Network Devices

User can't directly transfer data to n/w devices but a ~~connection~~ connection to be maintained for communicate. in Kernel's networking subsystem.

## " INTERPROCESS COMMUNICATION"

Communication may be telling other process know that some event has occurred or it may involve transfer of data.

### Synchronization & Signals

Informing a process that an event has occurred is ~~synchronization~~ signal. Limited no. of signals are available & they cannot carry information.

If a Kernel mode process is expecting an event to occur, it will not normally use signal to receive notification of that event, rather it is done through scheduling states & wait-queue structures. Whenever, a process wants to wait in some event to complete, it place itself on wait-queue & tell scheduler that it is ~~available~~ not eligible for execution. The standard LINUX wait-queue synchronizes process communicating with semaphores.

### Passing Data among Processes

Standard UNIX pipe mechanism allows

a child process to inherit a communication channel from its parent, data at one end of pipe can be read at other. Each pipe has a pair of wait-queues to synchronize the reader & writer strains

# SECURITY

(1) Authentication —:
Use of publically readable password file, combined with salt value.
Original password cannot be deducted from password file except by trial & error.
A new alternative, Pluggable Authentication Module (PAM). is based on shared library that can be used by any system component that needs to authenticate users. If a new authentication module is added later, it can be added to configuration file all systems are able to take advantages of that. Also used for password changing.

(2) Access Control
u-id — single user or single set of rights
g-id
Read, Write, Execute Access
Root u-id is privileged one.