

Linux Kernel Internals – an overview

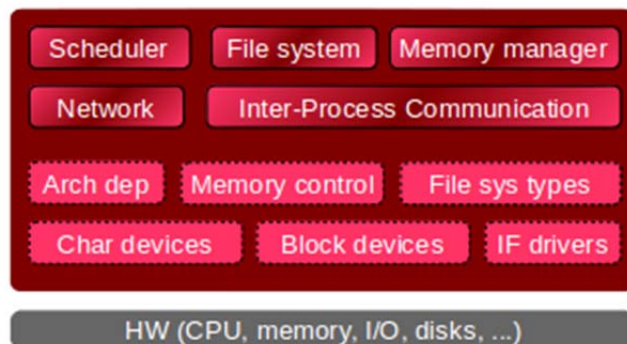
By Anders Törnqvist

The separation between kernel and user space is one of the fundamentals in Linux. This enables for the kernel to run with higher priority and to manage how functionality, resources and permissions are used and distributed over the entire system.

In kernel space execute all the central parts of the system to manage an efficient overall system with excellent throughput, execution performance and responsiveness.

All hardware accesses are managed from within kernel space by a lot of different hardware device drivers. Some drivers are very device specific but most drivers are designed and coded to be as generic as possible and isolates the hardware specific code into small isolated parts.

Most drivers automatically detects hardware existing in, or plugged into, the system and initializes is respectively. The separation of generic and hardware specific code makes it relatively easy to port code to new devices.



The functionality of the Linux kernel is often described to be:

Scheduler

The scheduler handles how the CPU execution resources are used for the entire system. It allocates time for different parts of the system depending on how to make the system efficient.

Data throughput versus responsiveness is often a trade-off that the scheduler works with. This can also be changed ,when building the kernel, or to some extent be tuned during run-time.

Memory Manager

All memory resources such as RAM, flash or just plain I/O is managed by the memory manager. It keeps track of which programs that uses specific parts of the memories and restricts the usage to only the allowed users that has the correct permissions.

A vital part of the memory management s the use of virtual memory spaces meaning that all process get their own virtual memory area into which physical memory is mapped. This enables for efficient memory handling.

If the system runs out of physical RAM the memory manager can use secondary memory, such as a hard disk, if that exists to use as extra temporary memory.

Virtual File System – VFS

This is the functionality that implements the general file system interface to the user space processes. The interface is always the same independently of which the underlying hardware is. VFS contains many different logical file-systems as well as hardware support for physical devices. Much of the file-systems are possible to tune for optimal performance.

Inter Process Communication - IPC

All communication between processes as well as between processes and the kernel are default made via the well defined IPC mechanisms in the kernel.

Most used are:

- pipes/fifos which are data pipes to carry data between processes
- message queues that are used to send events and data
- signals for pure event handling
- semaphores for synchronization. Often used combined with usage of some shared resources

Networking

Many Linux-based devices are network attached but not all, so the kernel code for networking is actually optional to use in your system. With a Linux system the network connectivity is very good since most of the known network protocols and devices, old and new, are supported in the Linux kernel.

The source code for the kernel internals is generally of very good quality. Some documentation exists, such as text files with the source code and a few books. Most recent and detailed "documentation" is always the source code itself.

Contact Anders at

Anders.Tornqvist@fosscentral.se
<http://www.fosscentral.se>