Operating Systems - Report on the Linux Systems

Samuel, Andersson, Johan Dahlberg, Eric Falheim,
Camilla Heiding, Xuan Hoang, Amer Hodzic

Today

Contents

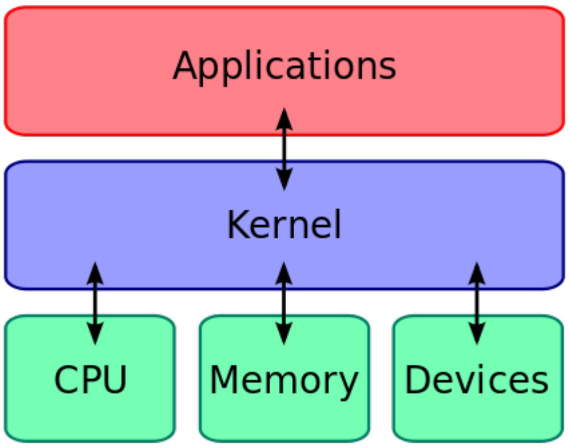
1 Linux History

1.1 The making of Linux

git gud - Linus Torvalds.

1.2 The kernel

The kernel is the core of the operating system and make communication possible to the hardware. It can exist many kernels in the same system. And in the case of a failure, when updating a kernel, you can always boot the system with an older version that was working.



Kernel interfacing hardware and user-space.

1.3 Linux licensing

The Linux kernel is distributed under the GNU General Public License (GPL). This means that anyone can download the kernel for free to use however they choose. If you modify the kernel and make a derivative of it, however, it must be distributed with the same licensing terms. So whenever anyone modifies the code and adds their own standard programs and tools in a packaged form, it could be called a new operating system. This is the reason for the extremely many different "flavors" of Linux.

2 Kernel Modules

A kernel module is a driver that can be loaded into the kernel dynamically, at boot-time or run-time, to make communication between a device and hardware possible (via the kernel). This makes the kernel more lightweight due to the fact that unused modules can be unloaded from the kernel, which frees up space.

All modules can be found in the /lib/modules/ folder. Working with modules is easy in Linux. There are commands you can run from the CLI to load, unload and list modules.

- lsmod - lists the loaded modules.
- modprobe [**-r**] *module_name* - will load or unload named module depending on the flag -r.
- insmod *module_name* - same as using modprobe without -r flag.
- rmmod *module_name* - same as using modprobe with -r flag.

3 Process Management

A process is a program in execution and it has it's own address space. It executes instructions sequentially, but each process can contain many threads. All threads share the same address space, and can therefore communicate via shared-memory.

3.1 Process attributes

A process is associated with a Process ID (PID) on creation, which will be unique for this process. Every process is spawned by a parent process (PPID) with a fork, and on creation the child-process memory space is identical to the parent's, but separate. Who spawned the process and what group they belonged to is stored in Real User ID (RUID), Effective User ID (EUID), Real Group ID (RGID) and Effective Group ID (EGID). The process has it's own priority which can be set from -20 (low) to 20 (high), by default it depends on recent CPU usage. In addition to this every process can be nice to other processes by hugging less resources. This value can be set between -20 to 19. The TTY attribute tells us what terminal the process is connected to.

3.2 Signals

A signal in linux is like an interrupt, which happens due to an event. When a process recieves a signal, it will terminate if it isn't catching it in any way.

- SIGABORT - Process abort
- SIGALARM - Alarm clock
- SIGFPE - Floating point exception
- SIGHUP - Hangup
- SIGILL - Illegal instruction
- SIGINT - Terminal interrupt
- SIGKILL - Kill
- SIGPIPE - Write on a pipe with no reader
- SIGQUIT - Terminal quit
- SIGSEGV - Invalid memory segment access
- SIGTERM - Terminate
- SIGUSR1 and SIGUSR2 - User defined signals

4 Scheduling

5 Memory Management

6 File Systems

6.1 Files

Everything is stored as files on Linux. Even directories, devices, sockets, pipes and symbolic links. When showing information about a file you see something like "-rwx r– r–". The first character determines the file's type. For regular files, a '-' is shown, and for directories a 'd' is shown. The remainder of the string determines the user-group-universal rights. For the example above, the user got read-write-execute rights, whereas the group and everyone else can just read that file. This adds a level of protection to the system so only the right processes and users can modify the file.

6.2 inode

An inode in UNIX-like systems are data-structures for files and directories. It could be simplified to a node in a tree. It would contain information about itself, it's parent-node and a list of child-nodes.

6.3 Mounting

There are no disk-drives in Linux, instead all devices are mounted on mounting points in the file-system. So installing a new hard-disk or inserting a USB-drive is treated the same way. By mounting the device on an inode, you can treat the new device as a part of the whole file-system.

6.4 File System Structure

The filesystem begins at the root /, where folders that divide the system logically is placed. Some examples:

- /bin - Contains binaries for the system. Easily accessed by the system via environment variable $PATH.
- /boot - Files for booting the system, including the kernel.
- /dev - Contains the system's devices.
- /etc - Configuration files for the system.
- /lib - Modules, software libraries and information databases.
- /mnt - Mounting point for external devices.
- /net - Mounting point for remote file-systems.
- /home - Home directory containing every user's own home folder on the system.
- /proc -
- /sbin - Binaries used by the administrator and the system.
- /usr -
- /tmp - A temporary directory that is emptied periodically, or when the system is shut down.

7 Input and Output

7.1 STDIN, STDOUT and STDERR

7.2 Redirection and Pipes

8 Interprocess Communication

9 Network Communications

10 Security