Introduction to Linux Operating Systems

Maximilian Schöbel

Hochschule Bonn-Rhein-Sieg

March 23, 2018

Levels and Layers of Abstraction in a Linux System

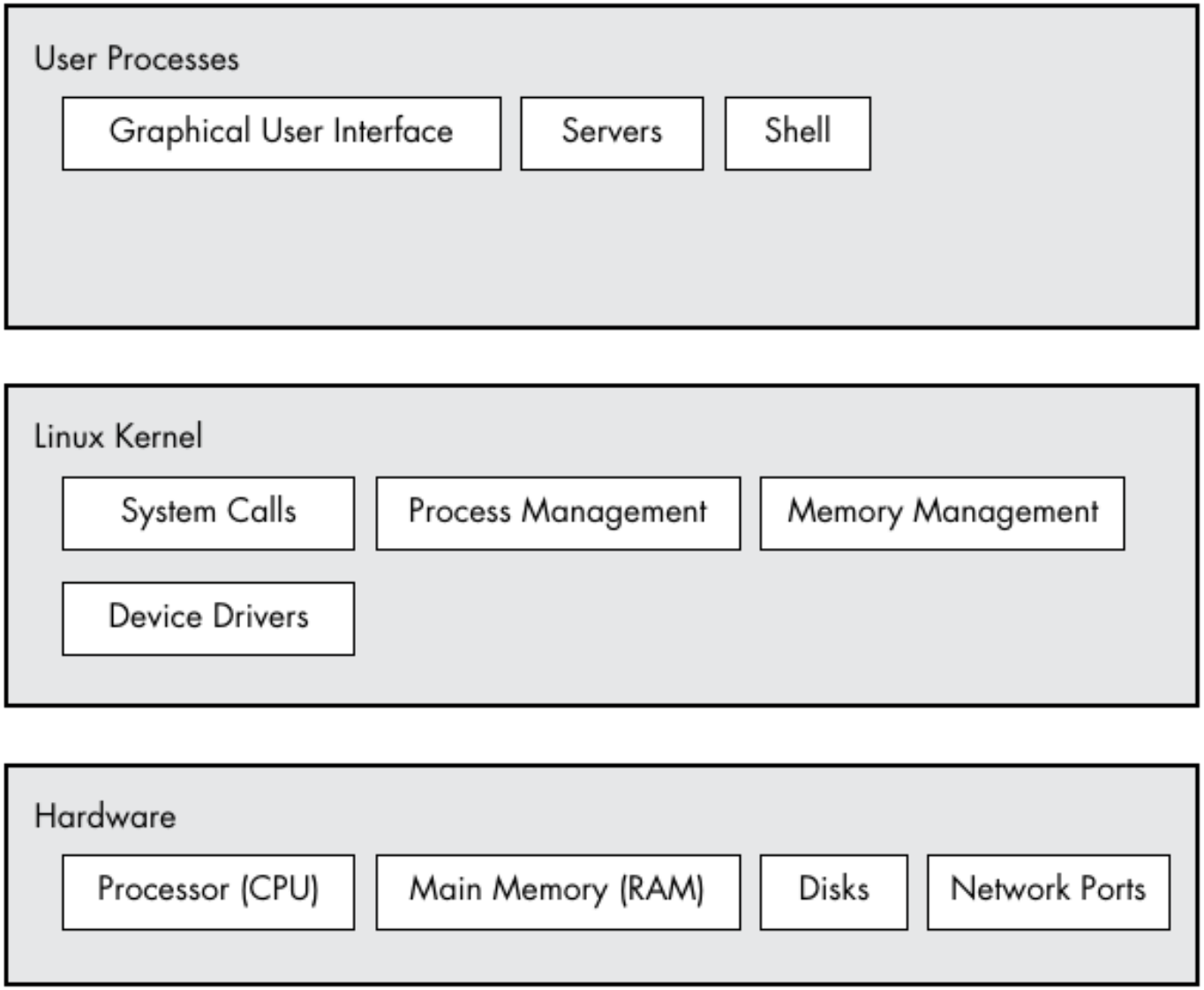


Figure 1: General Linux system organization

The Kernel's Functions

- Process Management
- Memory Management
- Device Drivers and Management
- System Calls and Support

User Space, Users and Groups

- User space: Memory reserved by kernel for user processes. This is where the action happens!
- A user is an entity that can run processes and own files. They exist to support permissions and boundaries.
- Each user can only affect his own processes and files.
- Exception: `root`

GNU/Linux

Linux:

- OS core, written by Linus Torvalds and others.

GNU:

- A set of utilities, small programs, written by Richard Stallman and others.
  `http://www.gnu.org`

The Linux Philosophy of Doug McIlroy

1. Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new features.
2. Expect the output of every program to become the input to another, yet unknown program. Don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.
3. Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

Everything is a File

In linux everything is accessible in a single namespace.

The Directory Tree

Directory	Content
bin	Essential command binaries
boot	Static files of the boot loader
dev	Device files
etc	Host-specific system configuration
home	User home directories
lib	Essential shared libraries and kernel modules
media	Contains mount points for replaceable media
mnt	Mount point for mounting a file system temporarily
proc	Virtual directory for system information
root	Home directory for the root user
run	Run-time variable data
sbin	Essential system binaries
sys	Virtual directory for system information
tmp	Temporary files
usr	Secondary hierarchy
var	Variable data
srv	Data for services provided by the system
opt	Add-on application software packages

Figure 2: Ubuntu directories

Terminal and the Shell

- A *shell* is a program that runs commands.
- Prominent example: The **B**ourne **A**gain **SH**ell.
- Anatomy of a shell command:
  `commandname -a arg1 -b arg2 -c arg3 arg4`
- Standard Input and Standard Output

Shell: Basic commands

- `cat`
- `ls`
- `cp`
- `mv`
- `touch`
- `rm`
- `echo`

Shell: Navigating Directories

Commands:

- `pwd`
- `cd`
- `mkdir`
- `rmdir`

Abbreviating Symbols:

- current directory: `.`
- parent directory: `..`

Shell Globbing (wildcards)

- Globbing: Match simple patterns to file and directory names by the shell.
- prominent shell-glob characters: `*` and `?`
- never ever even think of executing this:
  `sudo rm -rf /`

Shell: Intermediate Commands

- `grep`
- `less`
- `pwd`
- `diff`
- `file`
- `find` and `locate`
- `head` and `tail`
- `sort`

Shell: Symbols

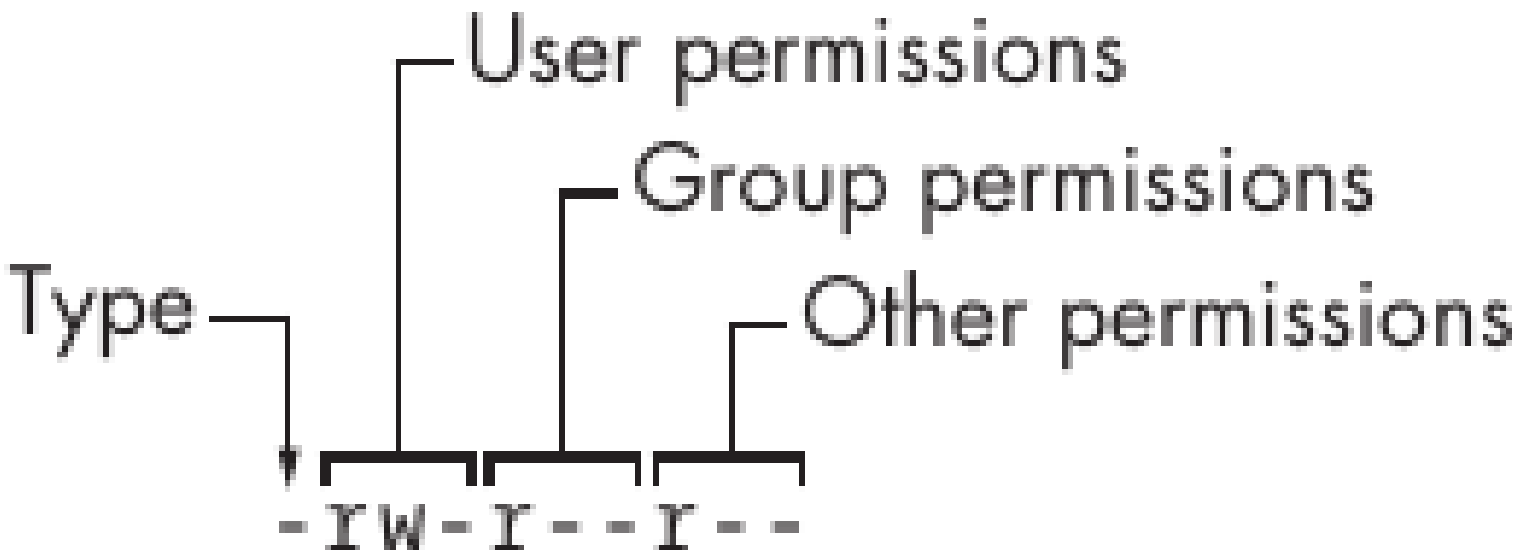
Advanced Features

- dot files (hidden files)
- Environment and shell variables
- The command path
- Getting Online help

Listing and Manipulating Processes

`ps`: a quick listing of running processes in the command line

File Modes and Permissions



Archiving and Compressing Files

Distributions

- Distribution = Kernel + Repositories
- Ubuntu: One of many

Ubuntu Package Managers

Package Installation