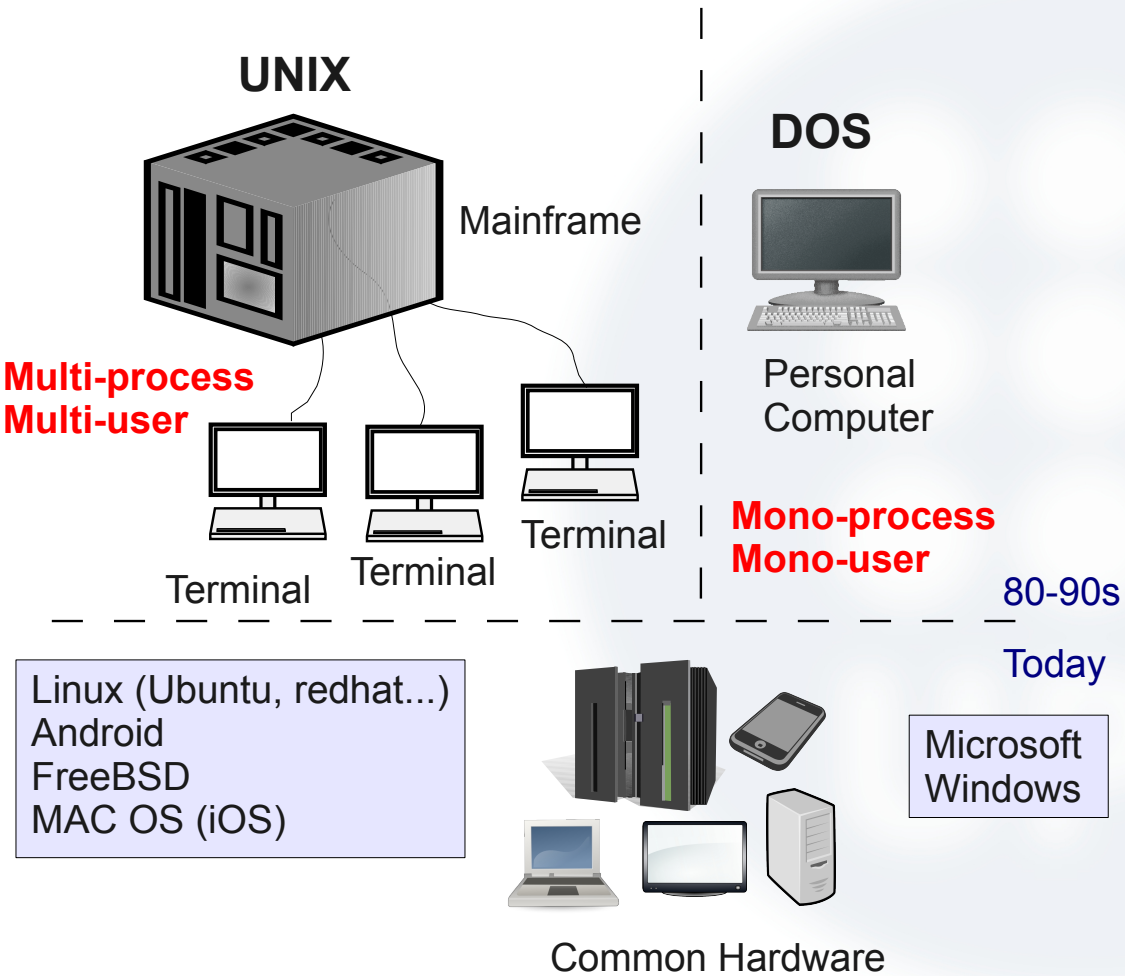
Introduction to Linux

Jose L. Muñoz, Oscar Esparza, Juanjo Alins, Jorge Mata
Telematics Engineering
Universitat Politècnica de Catalunya (UPC)

Outline

Introduction to OS

- An OS has to:
  - Manage the resources of a computer.
  - Provide an interface for human beings: CLI or GUI.
- OS Evolution:



Basic Unix Features

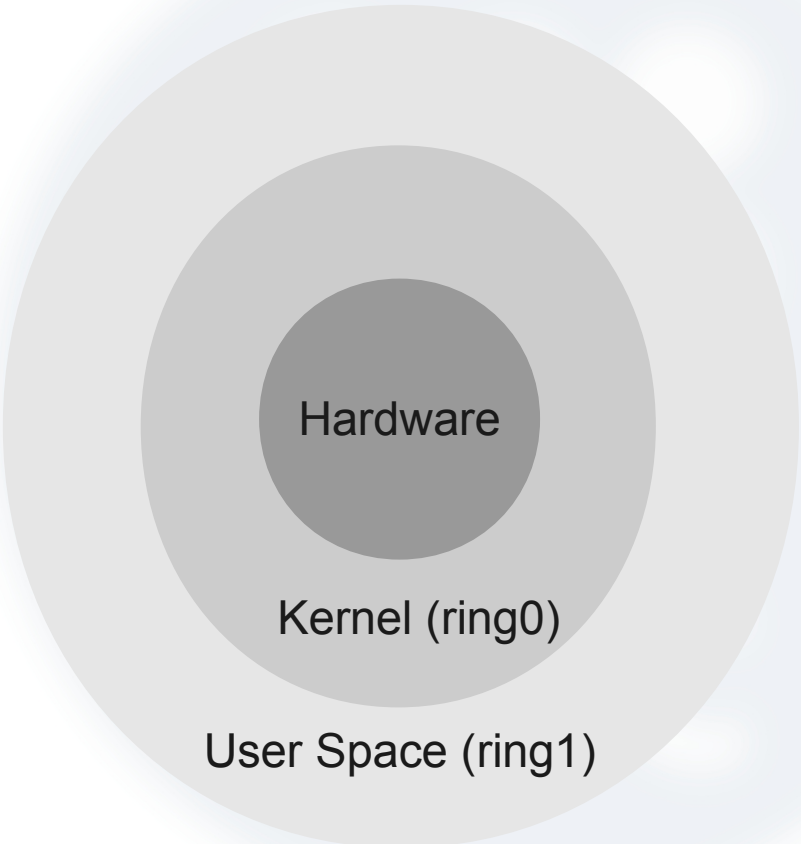
- **Multi-process.** Several processes can run on the same computing environment.
- **Multi-user.** Several users can share the computing environment.
- **Simple Interfaces.** Text as much as possible.

Outline

OS Rings

- **Kernel**. Kernel manages hardware and essential operations with the hardware.
- **User space**. Processes of users.

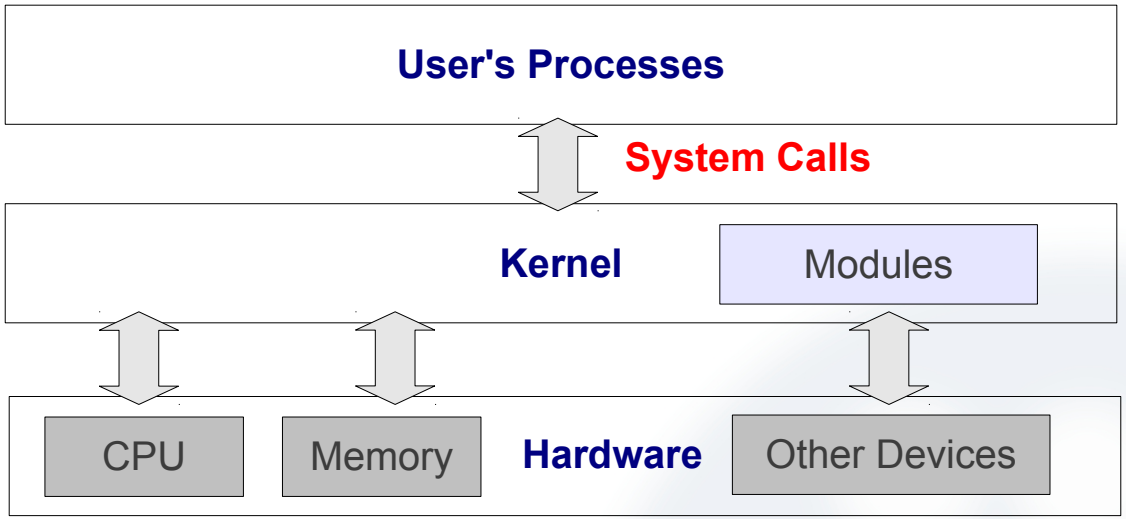
- Interfaces:
  - Hardware to Kernel: **privileged operations** (low level operations). E.g. manage CPU -> CPU scheduler.
  - kernel to user space applications: **system calls**. E.g. ask for creating a new process.



Hardware

- **Central Processing Unit (CPU**). The CPU is responsible executing programs.
- **Memory.** Memory is used to store both program instructions and data.
- **Input/Output (I/O) Devices.** The kernel manages requests from user applications to perform input and output operations and provides convenient methods for using each device.

System Calls



- The kernel provices an interface to user applications managing low level operations with the hardware.
- The kernel runs in “supervisor mode”.
- Unix systems provide a library or API that sits between user programs and the Kernel.
- C library such as glibc provides wrapper functions for the system calls. For example:

```
int kill(pid_t pid, int sig);
```

Modules

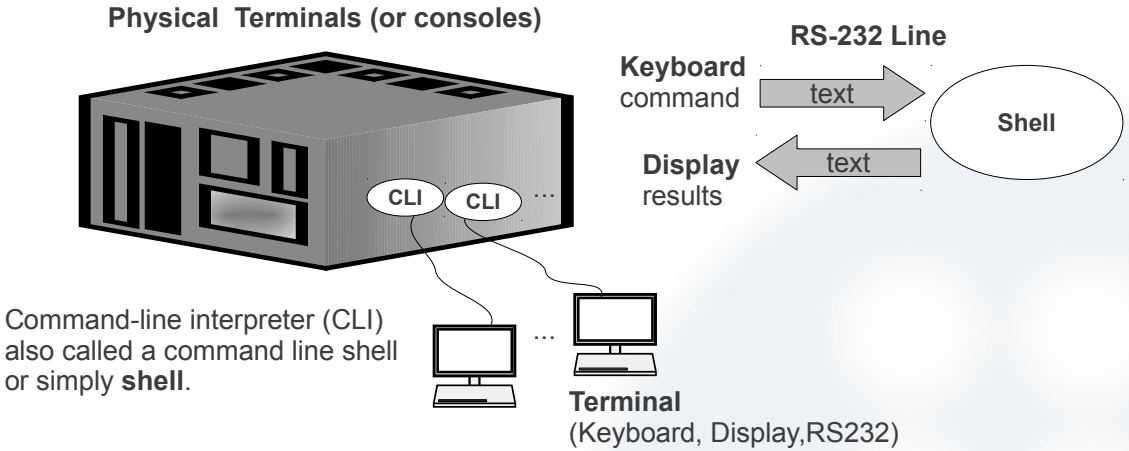
- The Linux Kernel is a **monolithic hybrid kernel**.
- Monolithic means the kernel is alone in supervisor mode.
- Hybrid means that kernel extensions, called modules, can be loaded and unloaded into the kernel upon demand while the kernel is running.
- Device drivers are one type of module.
- When you build a Linux kernel you can decide if you include a certain module inside the kernel (statically compiled module) or if you allow this module to be loaded at run time by your kernel (dynamic module).
- The commands to list, insert and remove modules are: `lsmod`, `modprobe` and `rmmod`.

Outline

Interacting with the OS

- There are two types of interfaces:
  - **Graphical User Interface (GUI).**
    - Require a graphical server (e.g. X server).
    - Processes are typically applications that have graphical I/O using mouse, keyboard, touch screens, etc.
  - **Command Line Interface (CLI).**
    - Does not require a graphical server.
    - Processes are commands that have only textual I/O.
    - The I/O of the commands is related or “attached” to a terminal.
    - The terminal is also attached to a command line interpreter or shell.
    - Types of terminals: Physical terminal, Virtual Console and Terminal Emulator.

CLIs



Virtual Consoles

TTY
<Ctrl><Alt><F1> **/dev/tty1**
<Ctrl><Alt><F2> **/dev/tty2**
…
<Ctrl><Alt><F6> **/dev/tty6**
No need for a graphic server

Terminal Emulator
or pseudo-terminal

xterm (classical from X)
gnome-terminal (GNOME)
konsole (KDE),etc.

<Ctrl><Alt><F7> **/dev/pts/X**
Graphic server running

Outline

`su`

- The `su` command stands for “switch user”.
- Useful for changing the user without having “to relog“.
- It allows you to become another user or execute commands as another user:

```
$ su telematic
```

- You are prompted to enter **the password associated with the account to which you are switching**.
- To exit: type exit or **`Ctrl-d`**.
- Per-command `su`:

```
$ su user -c command
```

- However, `su` is potentially dangerous.
- E.g. people knowing the password of the root.

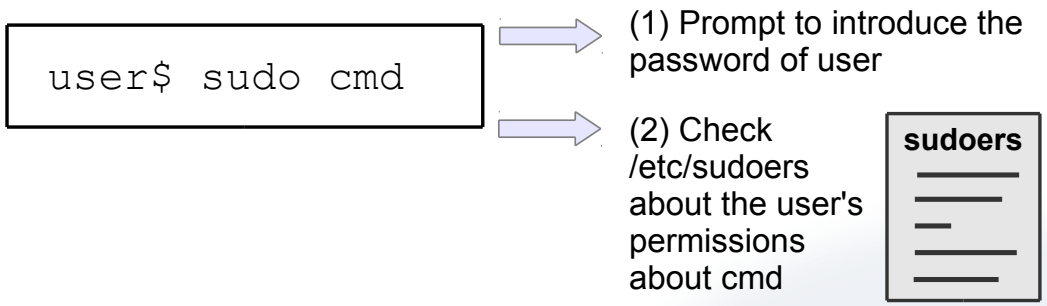
sudo I

- Using the sudoers file (/etc/sudoers), system administrators can define which users or groups will be able to execute certain commands (or even any command) as root.
- The advantage is that none of these users will have to know the password of root.
- The `sudo` command **prompts to introduce the password of the user that is executing `sudo`.**

```
$ sudo command
```

- Additionally, if your user is configured as system administrator in the sudoers file you can get a shell as root.

sudo II



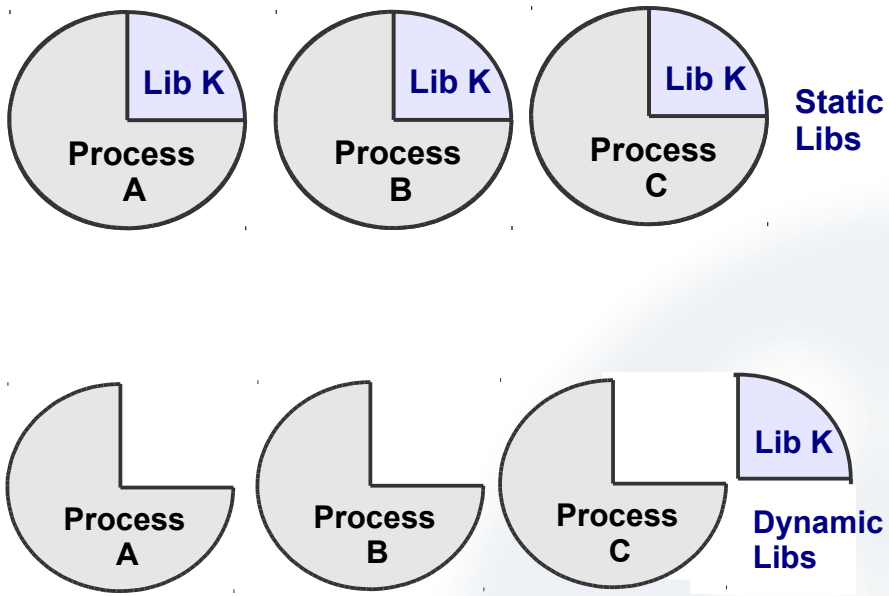
- To become root you can type:

```
user$ sudo -s
root#
```

- **Note. We will use $ to mean that the command is being executed as a regular user and # to mean that the command is being executed as root.**
- You can use the command `whoami` to know which user you are.

Outline

Types of Libraries I



- **Static libraries** provide a set of functions that are included at compile-time into the final executable file.
- **Dynamic libraries** allow its set of functions to be referenced in user code and defined in a shared library to be resolved at run time, that is to say, when the program is loaded to become a process in the system.

Types of Libraries II

- Static libraries pros and cons:
  - Anything used from these libraries is available (with the correct version) before the program is executed and this avoids dependency problems.
  - The main **drawback** is that the size of executables is bigger (more disk and memory).
- Dynamic libraries pros and cons:
  - Shared library code is not present in the executable.
  - Load time may be also reduced (library code might be in memory).
  - Libraries can be updated without updating applications.
  - The main **drawback** is that they usually establish complex relationships between the different packages of software installed in a system (library versions etc.).

Types of Libraries III

- A typical “hello world” program in C:

```
/* Hello World program */
#include<stdio.h>
main()
{
   printf("Hello World\n");
}
```

- We can compile and execute the program:

```
$ gcc -o hello hello.c
$ ./hello
Hello World
```

- By default the `gcc` compiler creates dynamic executables.

Types of Libraries IV

- You can check dependencies with the `ldd` command.

```
$ ldd hello
linux-vdso.so.1 =>  (0x00007fff5e7ca000)
libc.so.6=>/lib/x86_64-linux-gnu/libc.so.6 (0x0..)
/lib64/ld-linux-x86-64.so.2 (0x00007fca8f089000)
```

- We can view the size of our executable with the `du` (disk usage) command:

```
$ du hello
12      hello
```

Types of Libraries V

- Compare this when we statically compile the same program:

```
$ gcc -static -o hello.static hello.c
$ ./hello.static
Hello World
$ ldd hello.static
    not a dynamic executable
$ du hello.static
860     hello.static
```

- The advantages of dynamic executables are clear.
- But the management of the different libraries on the system results in a challenge colloquially known as "dependency hell".

Sofware Packages

- A package of software tracks where all its files are, allowing the user to easily manage the installed software: view dependencies, uninstall, etc.
- Generally, in Linux, software packages copy their executables in /usr/bin, their libraries in /usr/lib and their documentation in /usr/share/doc/package/.
- There are multiple different package systems, two main are:
  - Red Hat Packages (**.rpm** files).
  - Debian Packages (**.deb** files).
- In Debian (Ubuntu) we use deb packages.
- With the `dpkg` command we can install and remove deb packages, view the files installed by a package, view the packages installed in the system, etc.

Advanced Package Management I

- The tool `dpkg` does not manage dependencies.
- A new generation of package management systems was developed to make this tedious process easier for the user.
- For the deb packages the tool is called APT.
- With APT, we essentially say “install this package” and all dependent packages will be installed/upgraded as appropriate.
- We need to configure the “repositories” that contain our deb packages.
- For APT, repositories are defined in files in /etc/apt.

Advanced Package Management II

- Typical APT commands:

APT command	Description
`apt-get update`	update the list available packages from online repositories
`apt-get dist-upgrade`	upgrade specified packages (or all installed packages if none specified)
`apt-get install <package list>`	install latest version of package(s)
`apt-get remove <package list>`	remove specified packages from system
`apt-cache list [package list]`	list available packages from repositories

Advanced Package Management III

- APT requires to execute `apt-get update` to update the available list of packages available in online repositories.
- If the update is not executed, APT works with the local cache, which might be outdated.
- In an Ubuntu system, we can type the name of an application in the console and, if it is not installed, the system will tell us how to install it:

```
$ xcowsay
The program xcowsay is currently not installed.
You can install it by typing:

sudo apt-get install xcowsay
```