LINUX USER GUIDE (I)

The UNIX family of operating systems

Overview

Cristian Vidrașcu
cristian.vidrascu@info.uaic.ro

February, 2024

Overview

- **Introduction and history of UNIX**
  - What is UNIX ?
  - What is Linux ?
  - Brief history of the evolution of UNIX
- **Overview of UNIX systems**
  - The structure of a UNIX system
  - General characteristics of a UNIX system
  - POSIX standard
  - UNIX and users
- **The Linux operating system**
  - The Linux kernel
  - Linux distributions
  - Why Linux ?
- **Bibliographical references**

Introduction and history of UNIX

Outline

- **Introduction and history of UNIX**
  - What is UNIX ?
  - What is Linux ?
  - Brief history of the evolution of UNIX
- **Overview of UNIX systems**
  - The structure of a UNIX system
  - General characteristics of a UNIX system
  - POSIX standard
  - UNIX and users
- **The Linux operating system**
  - The Linux kernel
  - Linux distributions
  - Why Linux ?
- **Bibliographical references**

What is UNIX ?

UNIX is the generic name for a broad family of *command-oriented*, *multi-user*, and *multi-tasking* operating systems, first developed in the 1970s by AT&T and Berkeley University.

Over time, it became a very widespread operating system around the world, used both in academia (in universities and research institutes) and in industry (in companies, non-profit organizations and public institutions).

* * *

What is a *command-oriented* operating system?

The system has a *command interpreter*, *i.e.* a program that has the task of taking the commands entered by the user, executing them and displaying the results of their execution.

Example: in `Windows NT` (*i.e.*, the current `Windows` family) we have a command interpreter, called `cmd.exe`, the "descendant" of the `command.com` interpreter from the days of the `MS-DOS` and the old `Windows 3.x/9x` family of operating systems, all of which were developed by Microsoft. Additionally, more recently, Microsoft has also developed another second interpreter for `Windows`, called `PowerShell.exe`.

Remark: in a UNIX operating system we have at our disposal several variants of command interpreters (*e.g.*, `sh`, `bash`, `csh`, `zsh`, etc.).

What is UNIX ? (cont.)

What is a *multi-user* operating system?

Such a system is characterized by the fact that there are user accounts, which have certain rights and access restrictions to files and other system resources. For this reason, various authentication mechanisms for users (*e.g.*, password-based) and protection of access to resources are used.
In addition, such a system allows several users to connect to the system and work **simultaneously**.

* * *

What is a *multi-tasking* operating system?

In such a system, several programs can be executed simultaneously (*i.e.*, at the same time). Running programs are called *processes*. Such simultaneous execution of several programs is also called *parallel execution*.
Another term used is that of *concurrent execution*, to emphasize the fact that programs in simultaneous execution *compete* for the use of the resources of the respective computer.

Remark: operating systems from the UNIX family are *time-sharing* systems based on priorities (more details in a future course).

What is Linux ?

Linux is a freely distributable UNIX variant with an open-source license for computers based on various hardware architectures (*e.g.*, Intel x86/x64, DEC Alpha, SPARC, PowerPC, MIPS, ARM and for other types of processors, such as those for *embedded systems*).

* * *

The first version of Linux was written in 1991 by Linus Torvalds, when he was a student, on an Intel 80386 PC.

It is currently continuously developed by a team of hundreds/thousands of Linux enthusiasts from around the world, under the guidance of a steering group led by Linus Torvalds.

Figure 1 – The TUX penguin, Linux's mascot.

* * *

Over time, Linux has become the most popular system in the UNIX family.

Brief history of the evolution of UNIX

The first version of UNIX was written at Bell Laboratories in 1969, by Ken Thompson, in assembly language on a PDP-7 minicomputer, being a *mono-tasking* system.
In 1971, the second version of UNIX was written in assembly language on a PDP-11.
Also in 1971 Ken Thompson created the B language, a simplified version of the BCPL language, with the goal of using it to write system utilities for UNIX in a high-level programming language.
In 1972 Dennis Ritchie, also at Bell Labs, began to improve the B language, creating the C language, which he then used with Ken Thompson to rewrite system utilities in C.
Thus, in 1973, for version 4 of UNIX, the kernel of the operating system was re-implemented in the C language, by Ken Thompson together with Dennis Ritchie, becoming a *multi-tasking* system, and its internal structure was greatly improved.

The high portability of programs written in C led to a fairly rapid spread of both the C language and the UNIX operating system, both of which gained popularity during the 1970s and 1980s.

Thus, numerous variants of UNIX appeared, made by different companies or universities. Two big contributors were imposed:

- AT&T Company and Bell Laboratories – developed successive versions of SYSTEM V UNIX
- Berkeley University – developed successive versions of BSD UNIX

Note: for more historical details I recommend consulting History of UNIX and History of the C language.

A simplified diagram of the various variants of UNIX can be found here (see also the next slide).

Brief history of the evolution of UNIX (cont.)

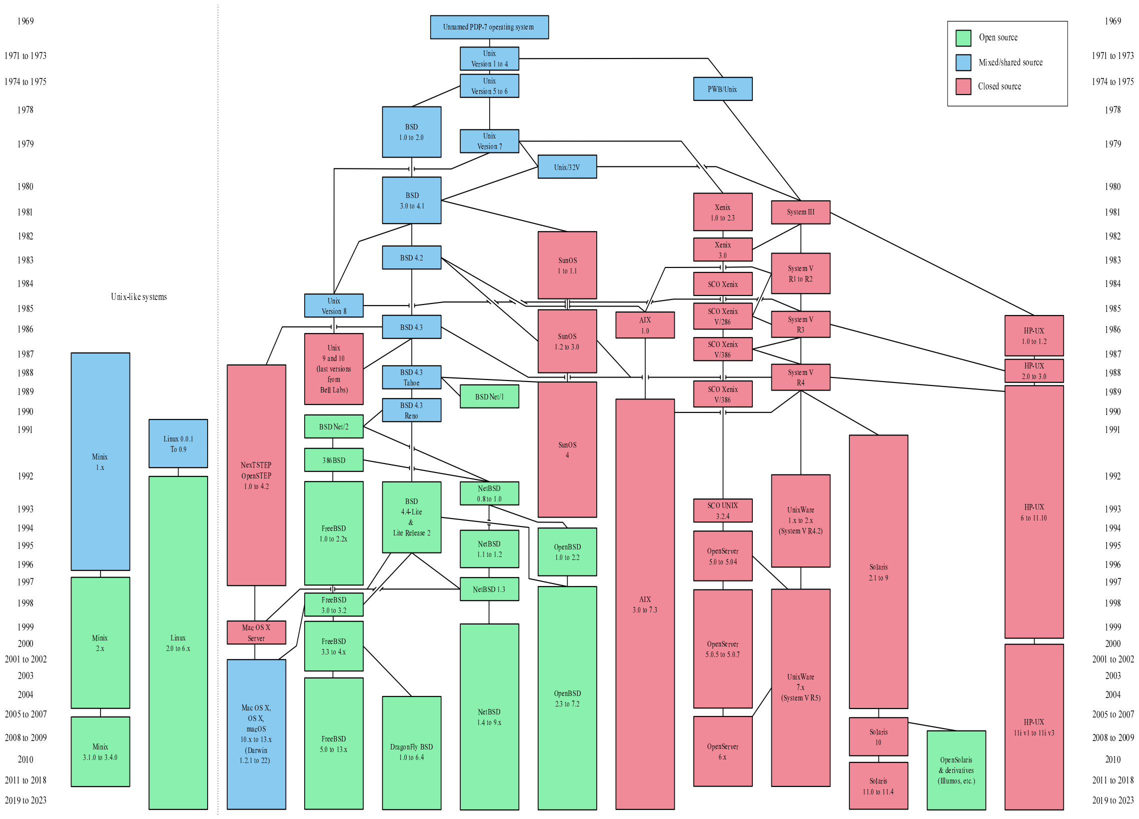


Figure 2 – A simplified diagram of the various UNIX variants. *Credits*: sursa imaginii.

Overview of UNIX systems

Outline

Introduction and history of UNIX
- What is UNIX ?
- What is Linux ?
- Brief history of the evolution of UNIX

Overview of UNIX systems
- The structure of a UNIX system
- General characteristics of a UNIX system
- POSIX standard
- UNIX and users

The Linux operating system
- The Linux kernel
- Linux distributions
- Why Linux ?

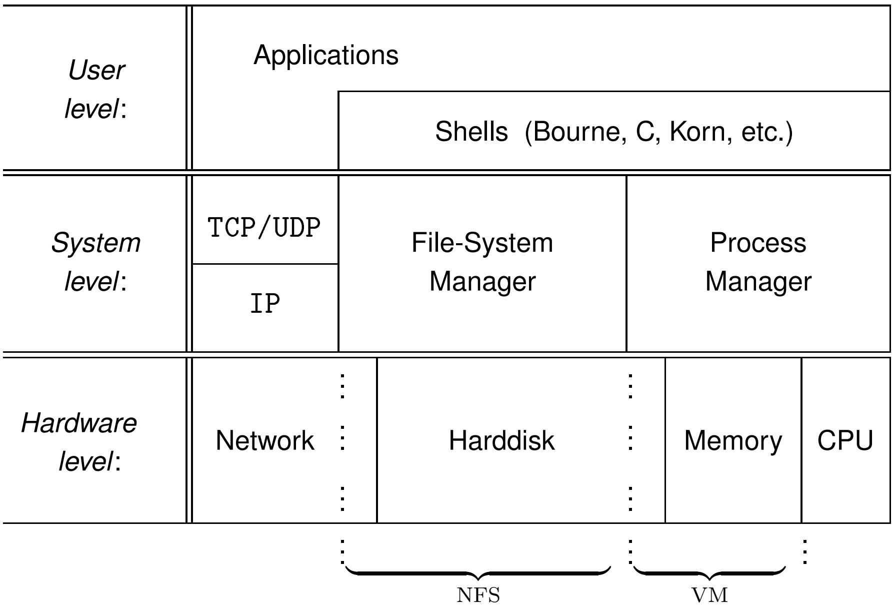
Bibliographical references

The structure of a `UNIX` system

In short, a system from the `UNIX` family is composed of:

- a **kernel**, which has the role of managing memory and low-level I/O operations, as well as planning and controlling the execution of various tasks (processes).
- a set of basic utilities such as:
  - different shells (*i.e.*, command language interpreters)
  - file manipulation commands
  - system activity (processes) management commands
  - communication commands between different users or systems
  - text editors
  - language compilers (C, C++, etc.) and a link-editor
  - general software development utilities: debuggers, archivers, source files management tools, generators of lexical and syntactic analyzers, etc.
  - various *filter* utilities, etc.

The structure of a `UNIX` system (cont.)



User level:

Applications

Shells (Bourne, C, Korn, etc.)

System level:

`TCP/UDP`

`IP`

File-System Manager

Process Manager

Hardware level:

Network

Harddisk

Memory

CPU

NFS

VM

Figure 3 – Simplified structure of a `UNIX` system.

General characteristics of a UNIX system

- The main concepts that UNIX relies on are the **file** concept and the **process** concept.

* * *

- The **file system** is logically organized in a *tree hierarchy*, based on the idea of a *directory* (*i.e.*, a “container” of files):

  As in Windows, the file system is organized as a recursive tree of directories that may contain subdirectories and actual files.

  Unlike Windows, in UNIX we have only one logical tree, and its single root is referred to by the name “/” . And the ’/’ character is used as a separator for subdirectory paths.
  In UNIX, filenames can be up to 255 characters long, can contain any number of ’.’ characters, and are *case-sensitive* (*i.e.*, a distinction is made between uppercase and lowercase letters).

General characteristics of a UNIX system (cont.)

- The main concepts that UNIX relies on are the **file** concept and the **process** concept.

* * *

- All the **processes of a system** are logically organized in a *tree hierarchy*, based on a mechanism of “*genetic inheritance*”:

  Each process in the system has a process that created it, called the *parent process*, and from which it “inherits” a certain set of characteristics (such as owner, access permissions, etc.), and may in turn create one or more *child* processes.

  Each process is assigned a PID (acronym from *Process IDentification*), which is a positive integer and is unique during the lifetime of that process (*i.e.*, at any given time, no two processes with the same PID exist in the system).
  There is a special process, the one with PID = 0, which is created when the UNIX system is initialized (booted) on that computer. It has no parent process, being the root of the tree of processes that will be created over time (until the computer is turned off).

General characteristics of a UNIX system (cont.)

- It is a **multi-user** system, *i.e.* several users can have simultaneous access to the system at any time, from different terminals connected to the respective system, terminals placed locally or remotely.
- It is a **multi-tasking** system, *i.e.* several programs can be executed simultaneously, by the same user or by different users.
- It is a **command-oriented** system, *i.e.* there is a *command interpreter* (sometimes also called a *shell*) that has the role of taking the commands entered by the user, executing them and displaying the results of their execution.

  On UNIX systems there are several command interpreters: `sh` (*Bourne SHell*), `bash` (*Bourne Again SHell*), `csh` (*C SHell*), `ksh` (*Korn SHell*), `ash`, `zsh`, etc.
  UNIX shells are more powerful than MS-DOS and Windows command interpreters (`command.com`, respectively `cmd.exe`), being similar to high-level programming languages: they have alternative and repetitive control structures (like `if`, `case`, `for`, `while`, etc.), which allows complex programs to be written as simple files with sequences of commands (called *scripts*).

General characteristics of a UNIX system (cont.)

- For the use of application programmers, a UNIX system provides a so-called API (an “application programming interface”), *i.e.* a set of “access points” to services provided by the kernel, which are callable from applications written in high-level languages (such as C).
  These “access points” are called **system calls**.

  Following efforts to standardize the API provided by early UNIX systems, the POSIX standard was adopted, which describes both the system call interface that we can use to write applications for systems in the UNIX family, and the main commands (*i.e.*, basic utilities) available on systems of the UNIX family, for working on the command line.
  The POSIX standard also applies to Linux as well as Mac OS X (since it has a UNIX-like kernel).
- Another facility provided is the **process I/O redirection** mechanism, the main use of which is a fundamental UNIX concept, that of **filtering**.

  Remark: we will see later examples of basic utilities that perform various filtering.

POSIX standard

Remark: the POSIX standard (an acronym that comes from *Portable Operating System Interface*) is actually a family of standards developed by the IEEE Computer Society for the purpose mentioned on the previous slide, that is, it has evolved over time, having various versions, just like the standards for C and C++ languages.

Note: for details on the evolution of this standard, I recommend consulting About POSIX.

* * *

Important:

In the first part of the semester you will learn to work on the command line with basic utilities and to use the facilities of `Linux` command interpreters. And in the second part of the semester you will learn to write programs for the `Linux` platform.

For this purpose, you will need to install the `Linux` operating system on your personal computer.

Note: you will be able to practice the acquired knowledge also on the `Mac OS X` system, thanks to the POSIX standard, common to the two platforms.

`UNIX` and users

- Each **user**, to be able to work on a `UNIX` system, must have an account on the respective system. The account is identified by a *username* and an associated *password* , which must be provided when connecting to the system (*i.e.*, the system authentication operation, called “*login*” for short). Also, each user account is assigned a number called UID (*User IDentification*).

  There is a special user, called *root* (or *superuser*), with UID = `0`. He has full rights over the entire system, being responsible for system administration.

  The “database” with information about user accounts is stored in the file `/etc/passwd`.

- There are **user groups**, which make it easier to manage user access rights and restrictions to system resources. Each user group is assigned a name (called the *groupname*) and a number called GID (*i.e.*, *Group IDentification*).

  The “database” with information about user groups in the system is stored in the file `/etc/group`.

Connecting remotely to a `UNIX` system is done with some specific commands.
(*Note*: we will discuss these commands in the next practical lesson.)

Outline

Introduction and history of UNIX

What is UNIX ?

What is Linux ?

Brief history of the evolution of UNIX

Overview of UNIX systems

The structure of a UNIX system

General characteristics of a UNIX system

POSIX standard

UNIX and users

The Linux operating system

The Linux kernel

Linux distributions

Why Linux ?

Bibliographical references

The Linux kernel

Linux is, in fact, the name of the kernel created by Linus Torvalds in 1991 and continuously developed by the team coordinated by him.

Note: the results of their work (*i.e.*, the various versions of the Linux kernel), are published on the site www.kernel.org and distributed on different channels (see next slide).

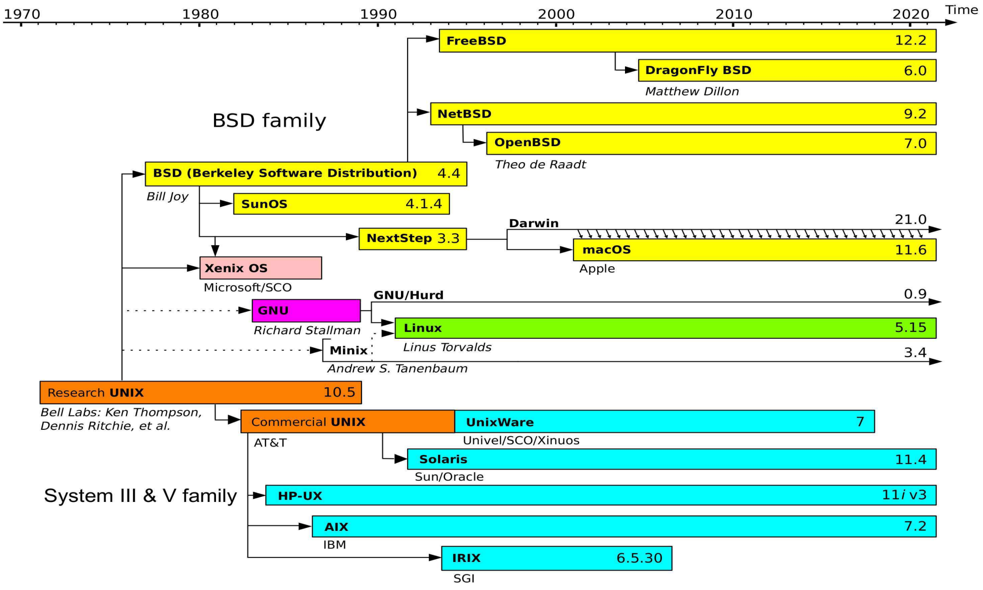


Figure 4 – A simplified diagram of the history of UNIX operating systems. *Credits*: image source.

Linux **distributions**

A *Linux distribution* is an operating system consisting of a **collection of software** based on the Linux kernel and a *package management system*.

That software package management system manages that collection, which will contain, in addition to the Linux kernel, a set of basic utilities developed within the GNU Project, plus various categories of **application programs** and **documentation**. (*Note*: hence the alternative name GNU/Linux operating system, which has also created controversy – for details, see here.)

In addition, that collection can also contain a **graphical interface system**, consisting of: a *window system* (the most used being the *X Window System*) + a *window manager* & *desktop environment* (*e.g.*, GNOME, KDE, Xfce, etc.).

That collection is managed and distributed to its users by a certain **entity**, which can be: a company, a non-profit organization, or even an individual.

* * *

Therefore, distributions are variants of Linux that differ from each other by the entity that manages them, by the version of the *kernel* included, by the software collection management system it uses, and by the suite of core utilities and application programs (each with its own version) that are included in the software collection that forms the respective Linux distribution.

Linux **distributions (cont.)**

The distributions also differ from each other in their appropriate configuration for the target audience of users: *desktop distributions* for home users, *server distributions* for various enterprise roles, distributions for embedded systems and IoT, etc.

Note: for a detailed description of the various types of Linux distributions, you can read here and here.

Some famous distributions, by popularity and longevity:

- *Red Hat*, since 1994;
- *Slackware*, since 1993;
- *SuSE*, since 1994, “derived” from *Slackware*;
- *Debian*, since 1993;
- *Ubuntu*, since 2004, “derived” from *Debian*;
- *LinuxMint*, since 2006, “derived” from *Ubuntu*;
- various specialized distributions: *Gentoo* (2002), *Arch Linux* (2002), *Linux From Scratch* (1999); *Puppy* (2003); *μClinux* (1998), *OpenWRT* (2001); *Android* (2007), *Chromium OS* (2009), etc.

To see the multitude of existing distributions, I recommend consulting “Map of distributions”, which contains a graphical representation of the history of various Linux distributions.

Note: in the past, the list of all available Linux distributions was provided on the official portal www.linux.org . However, the content of the portal has been reorganized, so that currently the site is a forum for discussion on topics related to Linux.
Now, the site DistroWatch provides information about currently available Linux distributions.

Why Linux ?

Which operating system is better, Windows or Linux ?

⟶ Windows *versus* Linux, *the "fanboy wars" on the Internet...*

In reality, this question does not admit an universally valid answer, but the answer depends on a number of factors, such as the specific use of the respective computer.

Thus, the Linux operating system is mostly used for servers – starting from medium and large enterprise environments, and up to the *data centers* of cloud providers and the HPC (*High Performance Computing*) "facilities" present in top500.org. Linux is also predominantly used in the segment of *embedded systems* and IoT (*Internet of things*), as well as in *smartphones*.

On the other hand, in the segment of computers used as *desktops* (both for home users, as well as for workstations of employees in institutions and companies), Windows is the most used operating system.

But both systems can be used equally well as a *desktop*, the choice of one over the other coming down to somewhat subjective factors – the preferences and abilities of the home user (or, of the employer), availability of a particular application for only one of the platforms, etc.

Remark: there are many statistical studies on the net about the "market share", in various segments, of the two systems. In support of the above statements, I will point you here to a single article for you to read: 111+ Linux Statistics and Facts.

Why Linux ? (cont.)

Here are some reasons to learn about Linux in the "Operating Systems" course:

- To also experience working with an operating system other than Windows (with which you have previous experience anyway and which you will use in most of the subjects encountered during the faculty years) and thus everyone will be able to decide, knowingly, which system he prefers to use, Windows or Linux, depending on the usage context as well.
- Teaching and understanding systems programming is easier on the Linux platform than on the Windows platform, due to the simplicity of the POSIX API compared to the Windows API (which has functions with longer names and larger argument lists).
- Teaching and understanding command line work in Linux, using the bash interpreter (or similar), is simpler than on the Windows platform, where we can use only two command interpreters: either cmd.exe, but which is very rudimentary compared to Linux interpreters, or powershell.exe, which is somewhat equivalent, as facilities provided, to Linux interpreters, but instead the syntax of commands is much more complicated (*i.e.*, very long names are used for commands and their options), similar to the difference between the prototype of functions in the Windows API compared to those in the POSIX API.
- The acquired knowledge and skills will also be useful in your future career in IT. Regardless of the role you will be filling (developer / system administrator / *DevOps*), you will need to know how to work on the command line. There are also many open-source software development tools that run on the Linux platform that you, as a programmer, will end up using.

Why Linux ? (cont.)

Here are some reasons why you should install Linux on your PC:

- So that you can work in graphical mode, which is more user-friendly (on the student server you have access only in text mode).
- So that you can also experience privileged commands and more generally have full control over system administration (on the student server you cannot do these things).
- So that you can access and work on a Linux system even during those times when the student server will be overloaded, making it difficult to respond, or even not responding at all, to your requests.
- You will also work on Linux at other courses studied during the faculty years (*e.g.*, in the following semester, in the "Computer Networks" course).

* * *

To install Linux on your personal computer, read very carefully the instructions in the installation guide I have developed, available on the course page ([2]), and follow the steps described in the guide.

Remark: it is recommended to install Linux directly on the hardware, by installing, most often, in *dual boot* mode with Windows; but it can also be the only operating system installed on the computer. Or, if you can't install it directly on hardware, at least install it in a virtual machine under Windows (though you'll suffer the performance penalty of the virtualization solution you're using).

Bibliographical references

Mandatory bibliography

[1] Chapter 1, §1.1 and §1.2 from the book "*Sisteme de operare – manual pentru ID*", by C. Vidrașcu, UAIC Publishing House, 2006. This is available as ebook at the address:

- https://profs.info.uaic.ro/~vidrascu/SO/books/ManualID-SO.pdf

[2] Guide to installing a Linux distribution on your personal computer:

- https://profs.info.uaic.ro/~vidrascu/SO/lectures/Linux/Install_guide.pdf