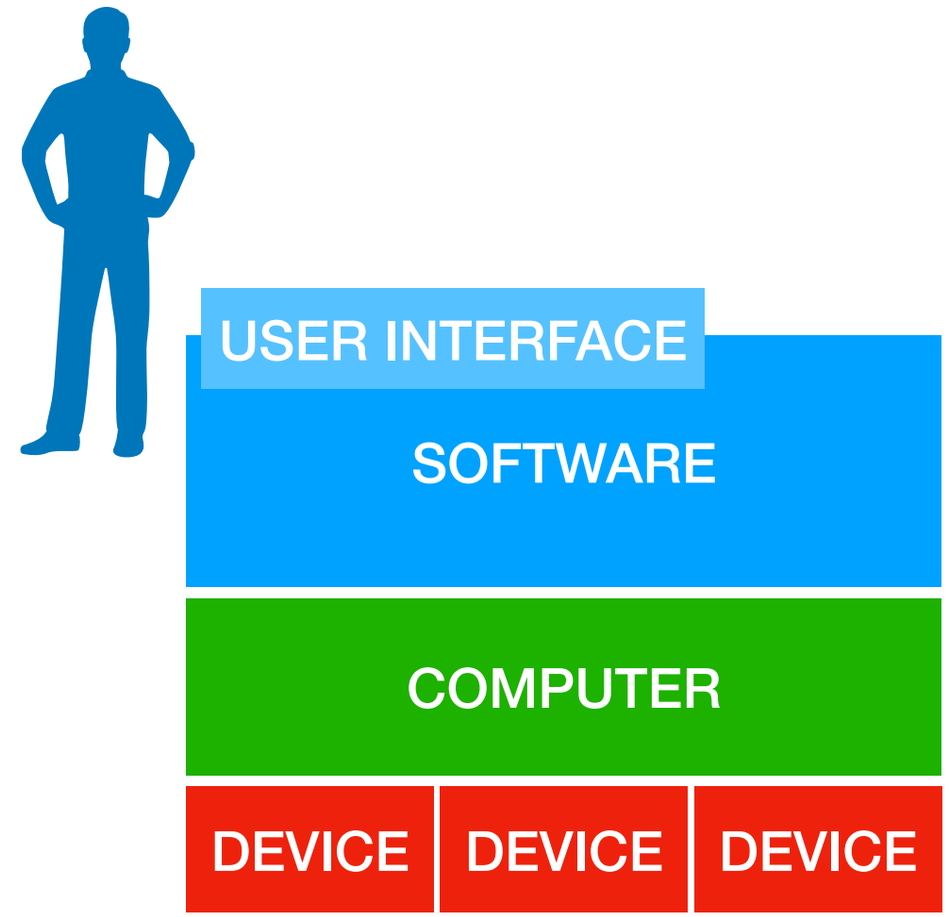
Princípios da Computação

The Unix shell

isep Instituto Superior de Engenharia do Porto

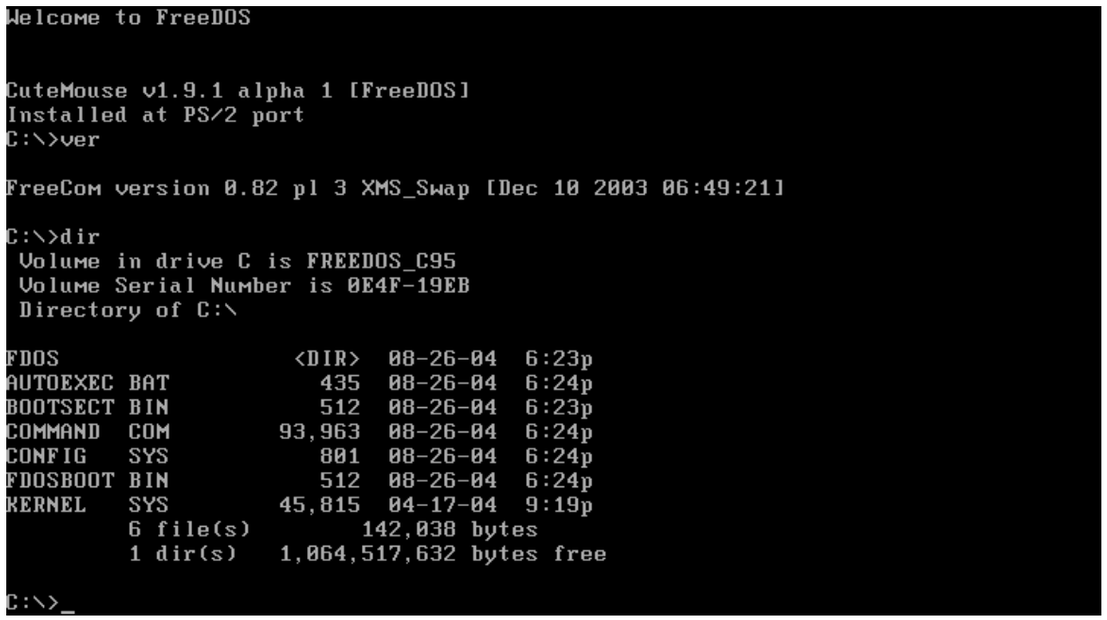
How do computers interact with users?

- If there is a USER, there must be a USER INTERFACE (UI).
  - The UI is software running on a computer.
  - The UI helps the user to operate the computer.

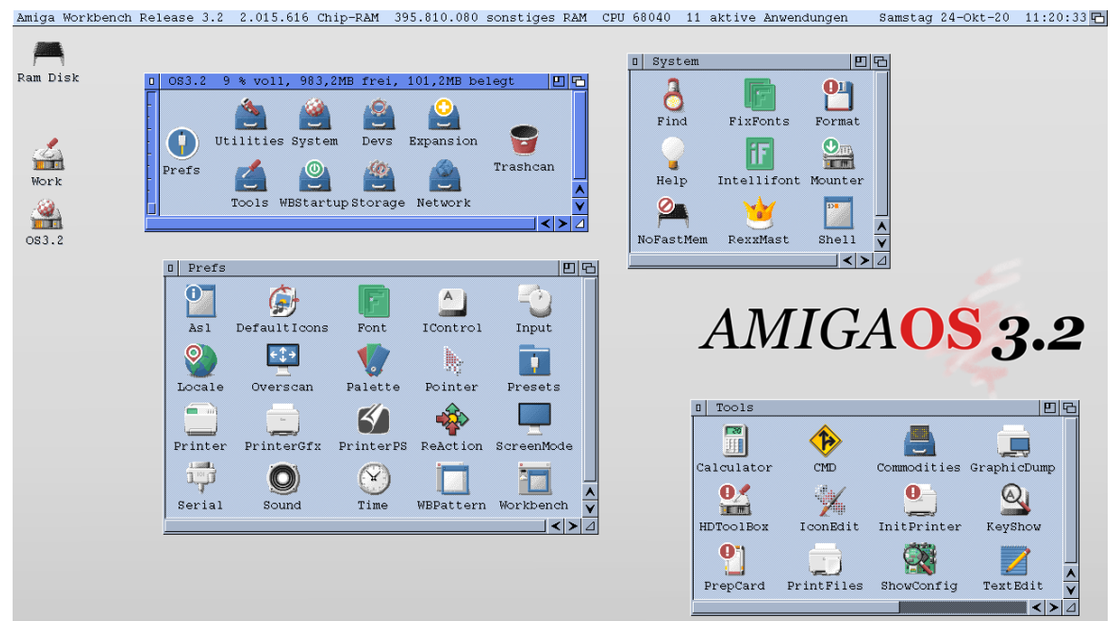


UI: text vs graphics

- Two approaches from the 1960's:



Command Line Interface (CLI)



Graphical User Interface (GUI)

Main goals

- Both approaches share the same goals:

Load a program into primary memory and run it

primary memory = RAM

Manage files stored in secondary memory

secondary memory = magnetic disks, flash cards, etc.

Main goals

Load a program into primary memory and run it

primary memory = RAM

- (CLI) Write the command name and press [ENTER].
- (GUI) Double click or tap on the program icon.

Main goals

Manage files stored in secondary memory

secondary memory = magnetic disks, flash cards, etc.

- Copy, move, rename, delete files.
- Create, copy, move, delete directories (aka folders).
- Manage file permissions (read, write, execute).

The Unix shell

isep Instituto Superior de Engenharia do Porto

Unix user interfaces

- The Unix operating system is completely user interface agnostic, and supports
  - many command line interfaces
  - many graphical user interfaces

The Unix shell

- The original Unix CLI is the **Thompson shell** (sh), created by Ken Thompson (1971).
    - Called *shell* because it is the outermost layer around the operating system.
- Several other shells were developed since then.
    - The most widely used is (possibly) the Bourne Again shell (bash).

Command line

- Any Unix shell accepts a command line, typed by the user.
- The first word in a command line is the **command**.
    - It is the name of a program installed in the computer, or
    - the name of an internal command, implemented in the shell.
- Following words (separated by spaces) specify the required behaviour of the program.
- The command line ends when the [ENTER] key is pressed.

Command line: an example

```
man -a printf
```

- **man** is the name of the command/program that will be executed

- **-a printf** are arguments delivered to the program, which the program uses to determine its behavior (what to do).

Unix file system

isep Instituto Superior de Engenharia do Porto

Files

- Data and programs are stored on persistent media, such as:
  - magnetic hard disk drives (HDD), and
  - solid state drives (SSD).
- The fundamental unit in persistent media is the **file**.
  - Programs are saved in files.
  - Data (images, documents, music, videos) are saved in files.

Files

17_audio_track.aiff	07/05/2021, 17:18	20,7 MB	Áudio AIFF
53182104917_f1fb21419a_o.jpg	12/09/2023, 22:43	3,7 MB	Imagem JPEG
eticket_3835-9249605.pdf	02/10/2023, 15:40	403 KB	Documento PDF
Grade Template.xlsx	Ontem, 16:22	17 KB	Micros...k (.xlsx)
openvpn-dei-config.zip	08/09/2023, 15:18	29 KB	Arquivo ZIP
trail_adventure.gpx	20/05/2023, 18:45	125 KB	Documento
Virtual_Coupling.docx	12/07/2023, 15:47	1,3 MB	Micros...(.docx)

Documents saved in files on a computer's disk.

Directories (or folders)

- **Directories** are special files that make it easier to organise files.
- A directory can store other subdirectories and files.
- Directories are organised in a tree structure.

Directories (or folders)

Directory

Subdirectories

Adventure_photos	:56	--	Pasta
august_at_the_beach	:55	--	Pasta
53182104917_f1fb21419a_o.jpg	12/09/2023, 22:43	3,7 MB	Imagem JPEG
53182167651_fc08bf9d89_o.jpg	12/09/2023, 22:50	3 MB	Imagem JPEG
53182611075_b0cdbe6c2c_o.jpg	12/09/2023, 22:50	3 MB	Imagem JPEG
hiking_in_the_alps	Hoje, 19:56	--	Pasta
53182666208_6c31fd160b_o.jpg	12/09/2023, 22:50	3 MB	Imagem JPEG
53182666253_72ba77b3b4_o.jpg	12/09/2023, 22:50	2,9 MB	Imagem JPEG
GPS_tracks	Hoje, 19:53	--	Pasta
easy_hike_Valongo.gpx	30/06/2023, 19:31	1,1 MB	Documento
trail_adventure_Arga.gpx	20/05/2023, 18:45	125 KB	Documento
wild_trail_Freita.gpx	13/06/2023, 22:26	1,4 MB	Documento
Travel_tickets	Hoje, 19:52	--	Pasta
eticket_3835-9249605.pdf	02/10/2023, 15:40	403 KB	Documento PDF
eticket_3835-9249606.pdf	02/10/2023, 15:40	403 KB	Documento PDF

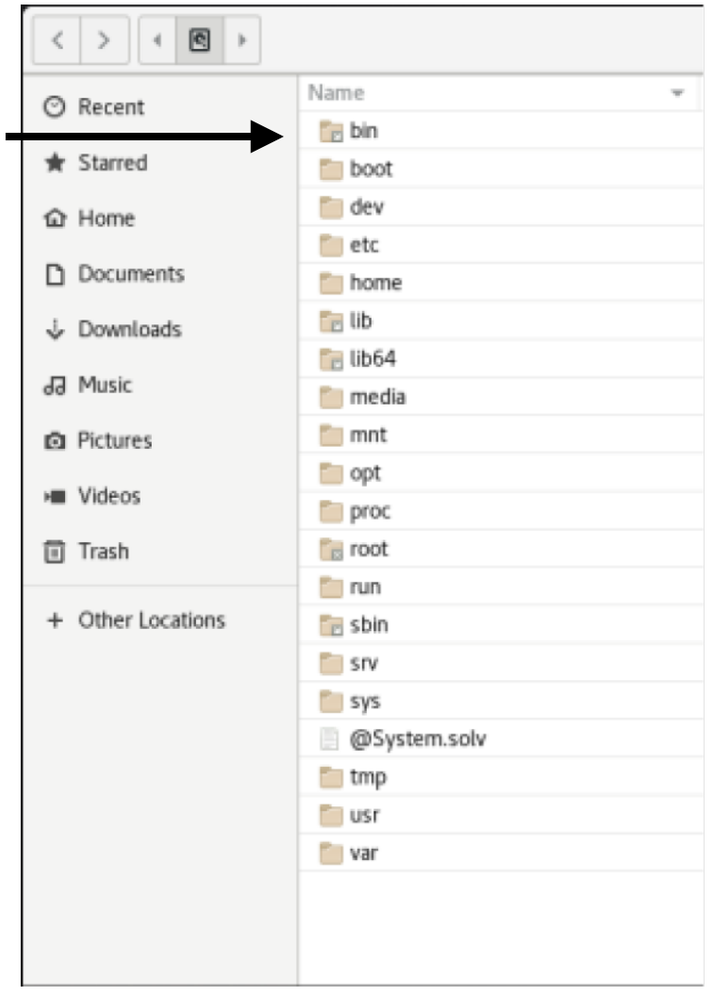
A directory tree

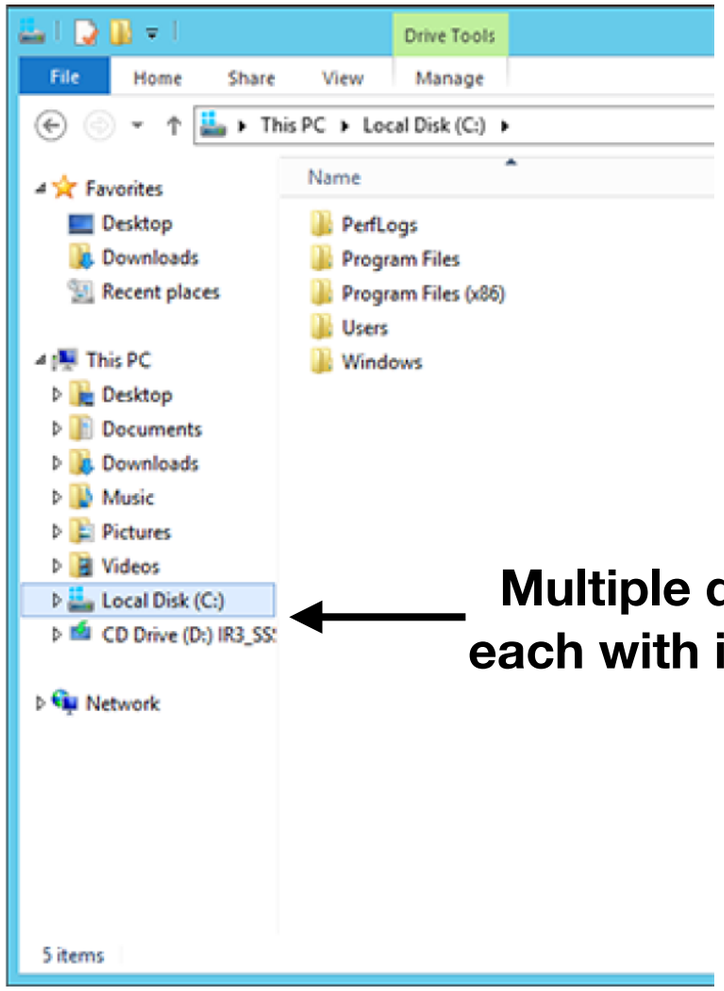
The Unix filesystem

- The Unix file system consists of a single tree.
  - Even if the computer has multiple disks, they are all aggregated into the same tree.
  - Unlike the Windows file system that presents a tree for each disk.

Unix vs. Windows file systems

The whole file system in one single tree

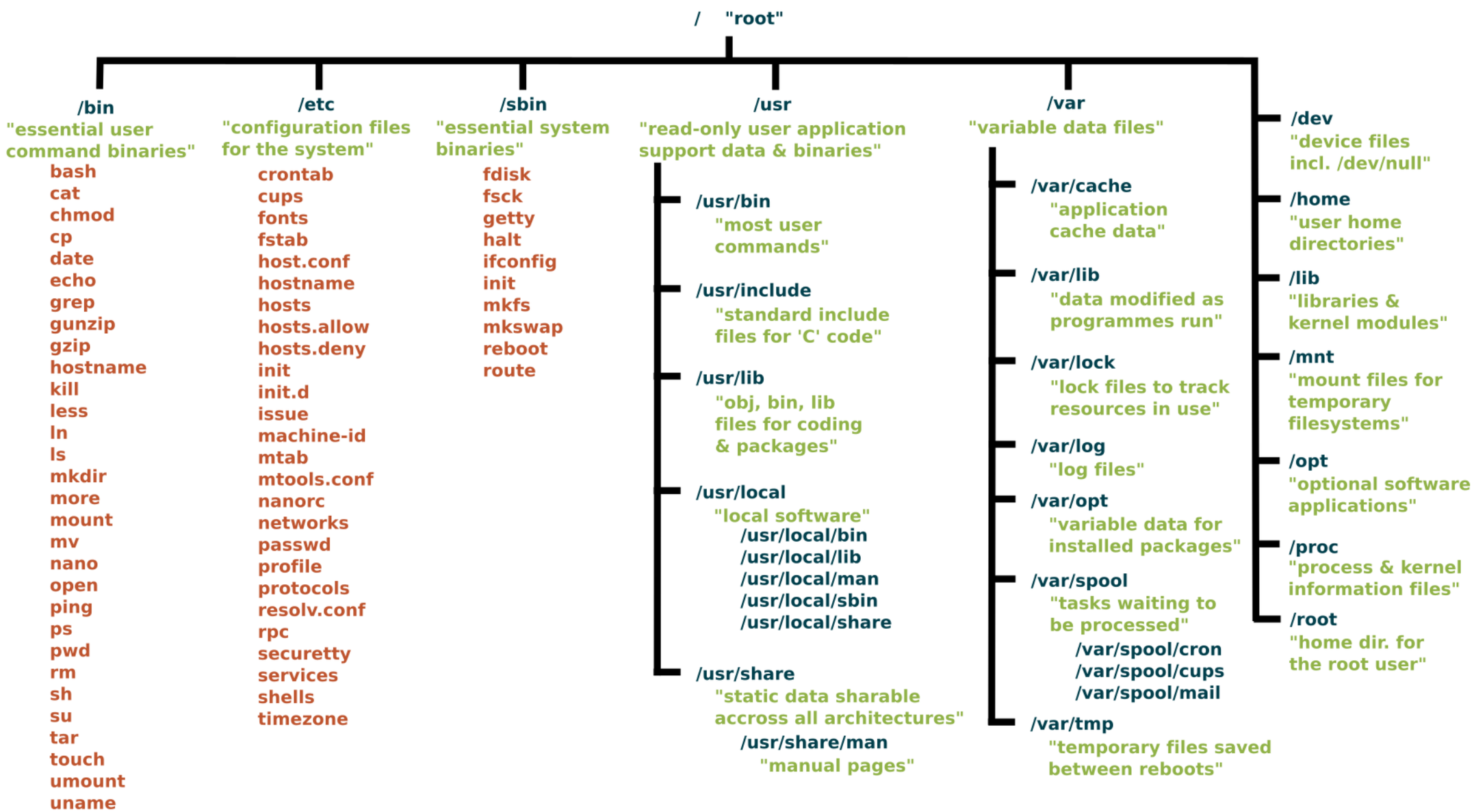




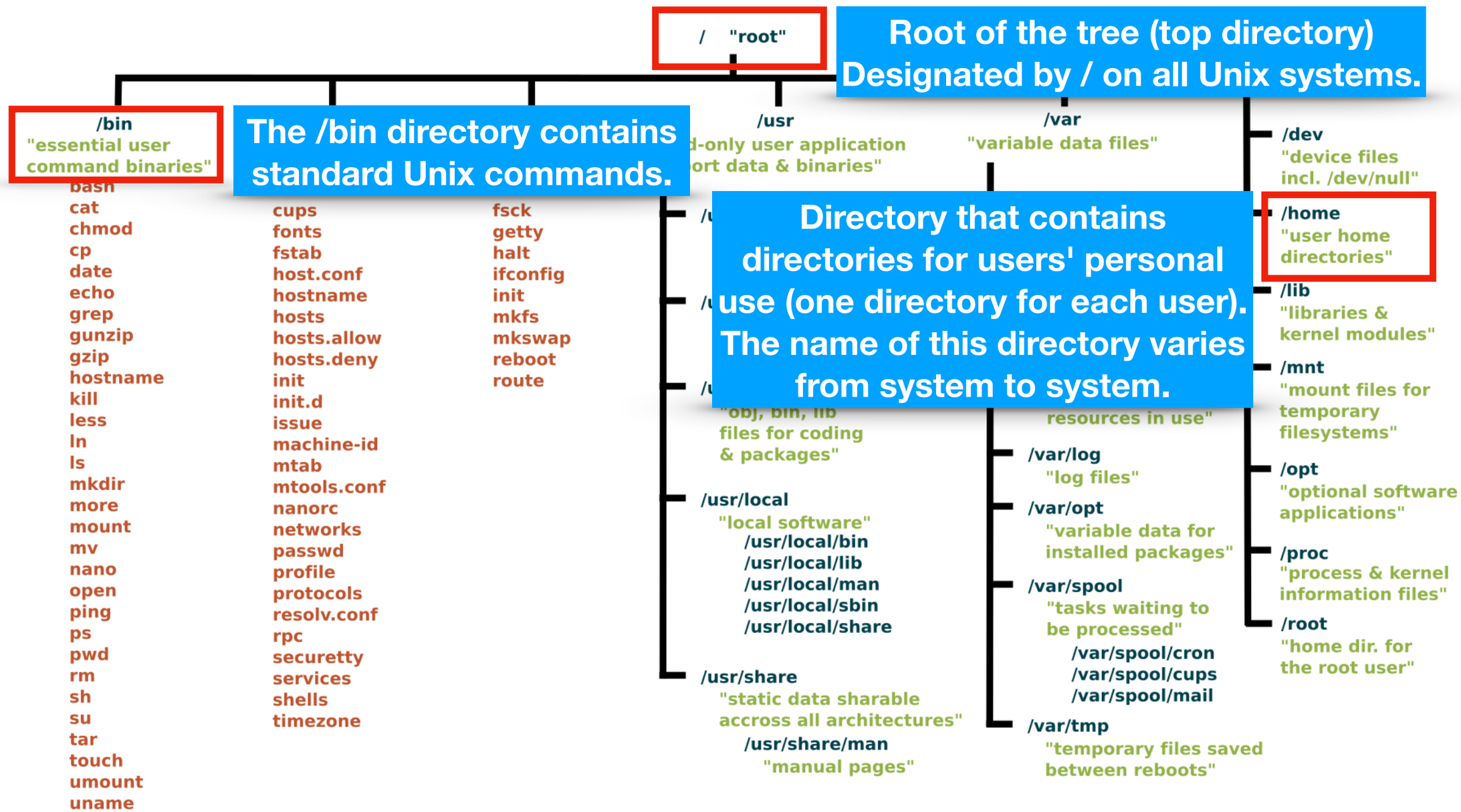
Multiple disk drives, each with its own tree.

Unix **Windows**

Typical Unix directory tree



Typical Unix directory tree



The *root* directory

- The root directory is at the top of the file system.
  - All directories descend from the root.
  - There is nothing above the root.
  - In all Unix systems, the root is called: **/** (slash)

The user's *home* directory

- Each user has a personal directory: the *home*.
    - A user can only create, change and delete directories/files within their home directory.
    - Any regular user is prevented from changing all other directories/files on the file system.
- Home directories are located in a directory whose name varies from system to system.
    - Common names: /home ; /user; /Users ; …

Frequently used file utilities

pwd : working directory name

- The shell always identifies a particular directory within which you are assumed to be working: the **working directory**.
  - When a user starts a shell session, the working directory is set to his home directory.
- The pwd command writes the absolute path name of the current working directory.

pwd : working directory name

```
user@machine ~ $ pwd
/home/user
user@machine ~ $
```

Absolute vs. relative path

- The **path** describes the location of a directory.
- **Absolute path:** describes the full path from the root.
    - It starts with a **/**
- **Relative path:** describes the path from the working directory.
    - It starts with the name of a subdirectory, existing in the working directory, or
    - with the parent (upper) directory symbol: **..**
    - with the current working directory symbol: **.**

Absolute vs. relative path — examples

- Absolute paths:
  - `/usr/bin`
  - `/home/user/documents/prcmp`
- Relative paths:
  - `documents/prcmp`
  - `../downloads`

ls : list directory contents

- It is used to see the contents (files and subdirectories) of a directory.
- It can show the contents of the local directory:
  - **`ls`**
- ...or the contents of a specified directory (absolute or relative path):
  - **`ls`** `/bin`

ls : list directory contents

- You can also use wildcards to filter the results.
  - **?** : any single character
  - * : any number (including zero) of characters
  - **[ ]** : any single character inside the square brackets
  - **{ }** : any single term (separated by commas) inside the curly brackets

ls : list directory contents

- `ls photo_??.jpg`
- `ls *.mp4`
- `ls program.[ch]`
- `ls report.{docx,pdf,odf}`

rm : remove directory entries

- This command is used to delete files.
- You must specify the names of the files to delete.
  - `rm some_file.pdf some_other_file.jpg`
- You can also use wildcards.
  - `rm *.txt`

cp : copy a file or directory

- You can create a copy of a file/directory in another directory.
  - `cp important_file.txt existing_directory`
- Or create a local duplicate with a different name.
  - `cp original_file.docx duplicate_file.docx`
- You can also use wildcards.
  - `cp *.pdf my_pdf_directory`

mv : move/rename a file or directory

- You can move a file/directory to another directory.
  - **mv** `some_file.txt existing_directory`
- Or rename a file/directory to a new non-existing name.
  - **mv** `old_name new_name`

cd : change directory

- This command allows you to change to another working directory.

- The directory is specified in either absolute or relative path.

  - **cd** `/var/log`

  - **cd** `documents/prcmp`

  - **cd** (changes directly to home directory)

mkdir : make a directory

- This command creates a new directory, with a name that does not yet exist.
  - **`mkdir`** `documents/prcmp/week1`

rmdir : remove a directory

- This command deletes an empty directory.
  - **`rmdir`** `documents/prcmp/week1`
- However, the easiest way to delete a non-empty directory in its entirety is using the `rm` command:
  - **`rm`** `-rf documents/non_empty_dir`
  - Check the `rm` manual page for the `-r` and `-f` options.

chmod : change file mode (permissions)

- In the Unix file system, each file/directory has a set of permissions:
  - r : permission to read
  - w : permission to write (or even delete)
  - x : permission to execute

chmod : change file permissions/mode

- This set of permissions is assigned to three distinct types of users.
  - u : (user) owner of the file
  - g : (group) users of the same group of the owner
  - o : (others) all other users
- The whole set of permissions for all types of users is called *mode*.

chmod : change file permissions/mode

```
user@machine Travel_tickets $ ls -l
total 1584
-rw-r--r--@ 1 user  staff  402866  2 Out 15:40 eticket_9249605.pdf
-rw-r--r--@ 1 user  staff  402866  2 Out 15:40 eticket_9249606.pdf
```

Other users permissions (o)

Group permissions (g)

Owner permissions (u)

chmod : change file permissions/mode

- The syntax of chmod is:
  - chmod *mode file*
- The mode can be *absolute* or *symbolic*.

chmod : absolute mode

- An absolute mode is an octal number.
- Each octal digit corresponds to the set of permissions for a type of user.
  - `chmod 754 nice_program` sets rwx r-x r--
  - `chmod 644 poem.txt` sets rw- r-- r--

chmod : relative mode

- A relative mode is described by a grammar:
  - who op perm
- who ::= **u** (owner), **g** (group), **o** (others)
- op ::= **+** (set), **-** (reset), **=** (exactly)
- perm ::= **r** (read), **w** (write), **x** (execute)

chmod : relative mode (examples)

- Raising **x** for **owner** and **group**; all other permissions are left unchanged.
    - `chmod ug+x nice_program`
- Removing **w** and **x** for group and others; all other permissions are left unchanged.
    - `chmod go-wx nice_program`
- All users can only read.
    - `chmod ugo=r nice_program`