Introduction to Linux-based OS by examples

Dott. Costantino Zazza, costantino.zazza@unitus.it

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

What's Unix (Linux)?

- *Multitasking* Operating System
- *Multi-user* system
- Is a powerful *software* development environment
- Is a *de-facto standard* developed and distributed by most of the ITC companies

Old fashioned Unix implementations:

AIX (IBM)
Digital Unix, **Ultrix** (DIGITAL)
SunOS, **Solaris** (SUN)
Linux (Free)
HP-UX (Hewlett-Packard)
IRIX (Silicon Graphics)

Brief History

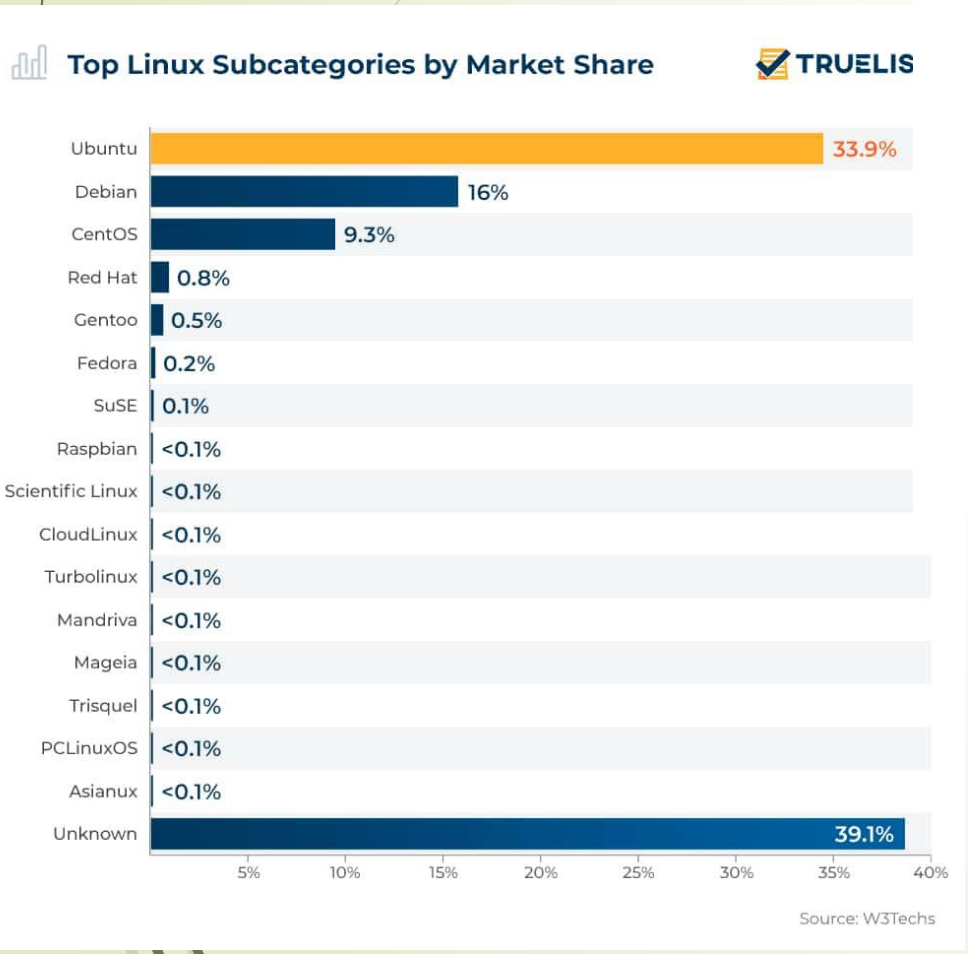
- 1969 - Ken Thompson - PDP-7 (Bell Laboratories)
- 1970 - Thompson - Ritchie PDP-11
- 1973 - Ritchie & Thompson riscrivono il *kernel* in C
- 1975/76 - Diffusione a livello universitario
- 1979 - Prima *release* BSD (Berkeley Software Distribution)

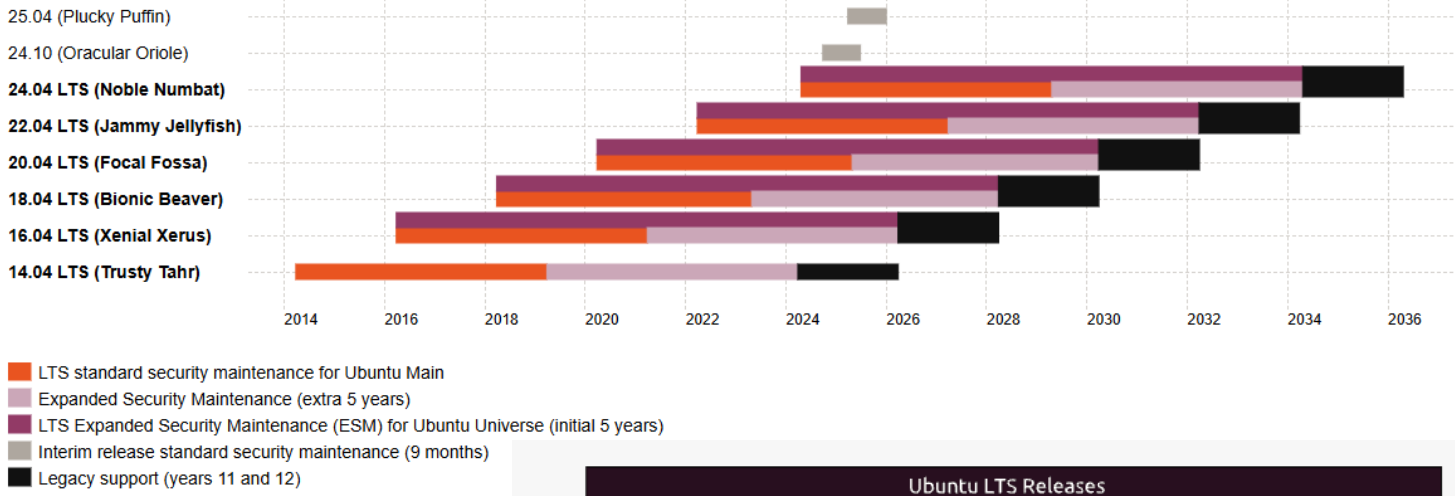
Now UNIX, that is, Linux, is available on almost any computing systems from Supercomputers to your smartphone(*)

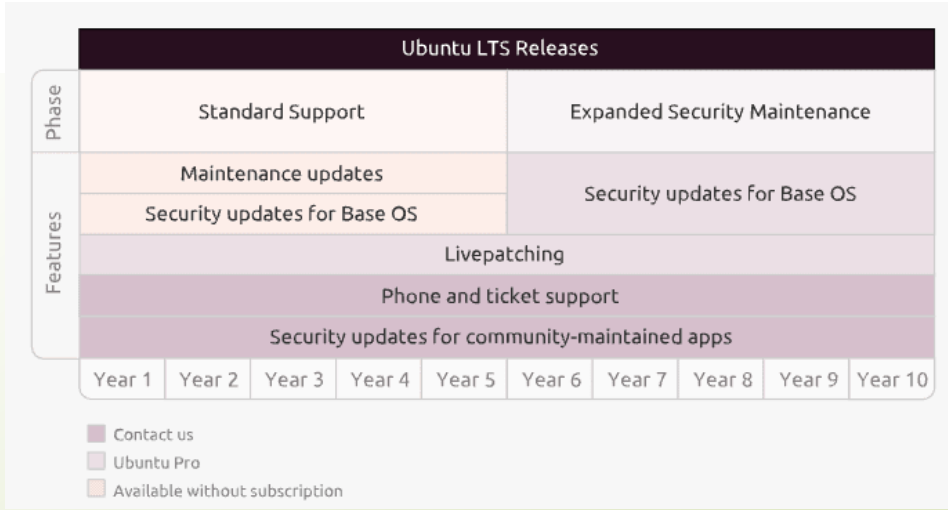
(*) Did you know your Android smartphone is running Linux for ARM processor?

Brief History

For the top 500 most powerful supercomputers, **Linux distributions have had 100% of the marketshare since 2017**. The global server operating system marketshare has Linux leading with a 62.7% marketshare, followed by Windows, Unix and other operating systems.







Brief History

CERN and Fermilab jointly plan to provide **AlmaLinux** as the standard distribution for experiments at our facilities, reflecting recent experience and discussions with experiments and other stakeholders. **AlmaLinux has recently been gaining traction among the community due to its long life cycle for each major version, extended architecture support, rapid release cycle, upstream community contributions, and support for security advisory metadata**. In testing, it has demonstrated to be perfectly compatible with the other rebuilds and Red Hat Enterprise Linux.

AlmaLinux product description

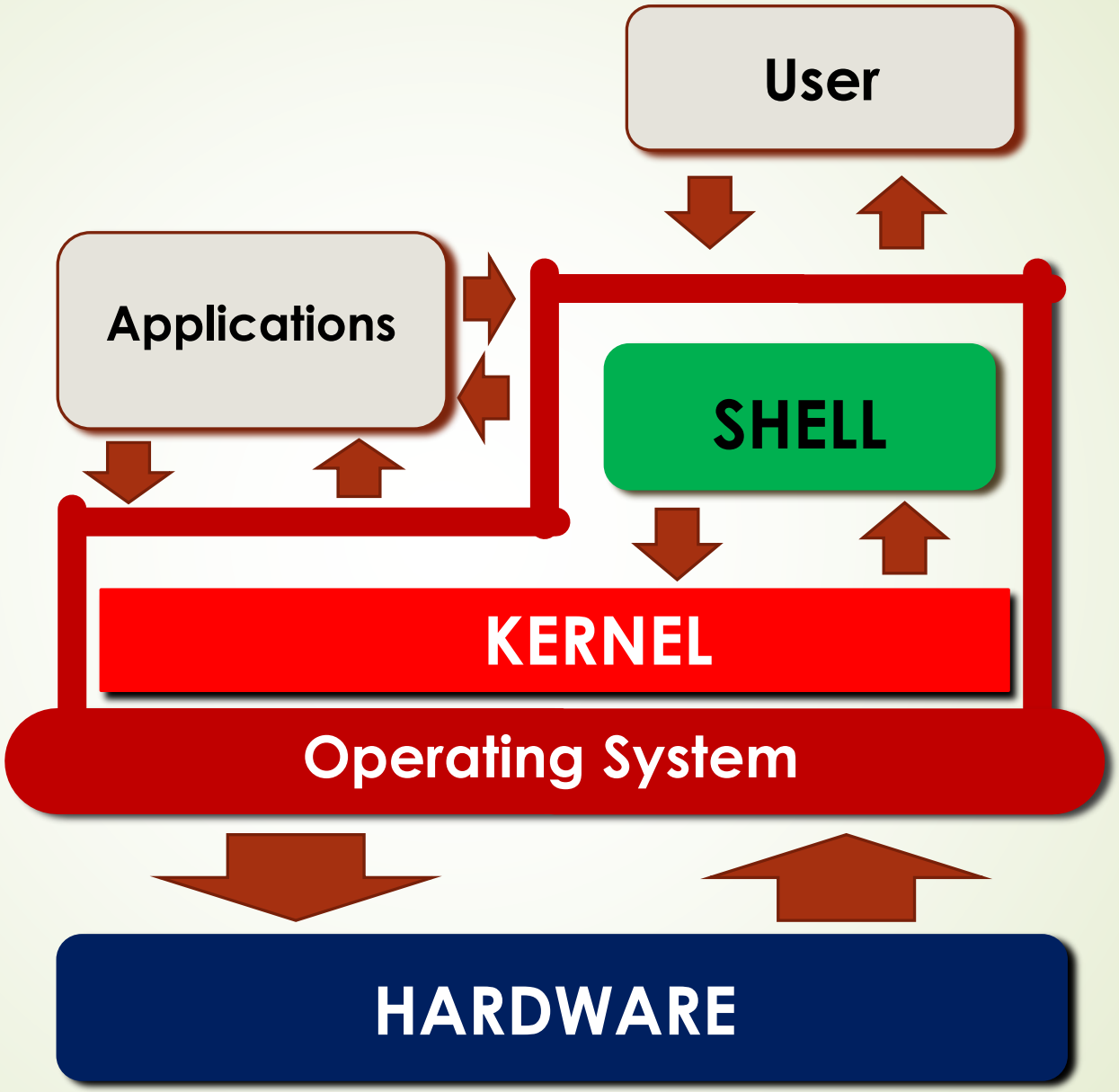
AlmaLinux OS is an open-source, community-driven Linux operating system that fills the gap left by the discontinuation of the CentOS Linux stable release. AlmaLinux OS is a 1:1 binary compatible clone of RHEL guided and built by the community.

AlmaLinux	AlmaLinux OS Foundati
Product	Vendor
9.5	GPL and more
Last version	Licence

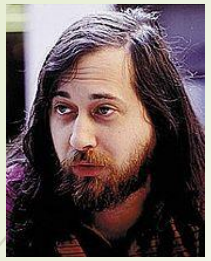
How to automate your product release and end-of-life strategy with Versio.io

Figure: Release, patch and end-of-life cycle of product AlmaLinux (AlmaLinux OS Foundation)

Unix structure



Unix like : GNU/Linux !



R. Stallman

1983 GNU Project / 1989 GNU GPL

…GNU Kernel …

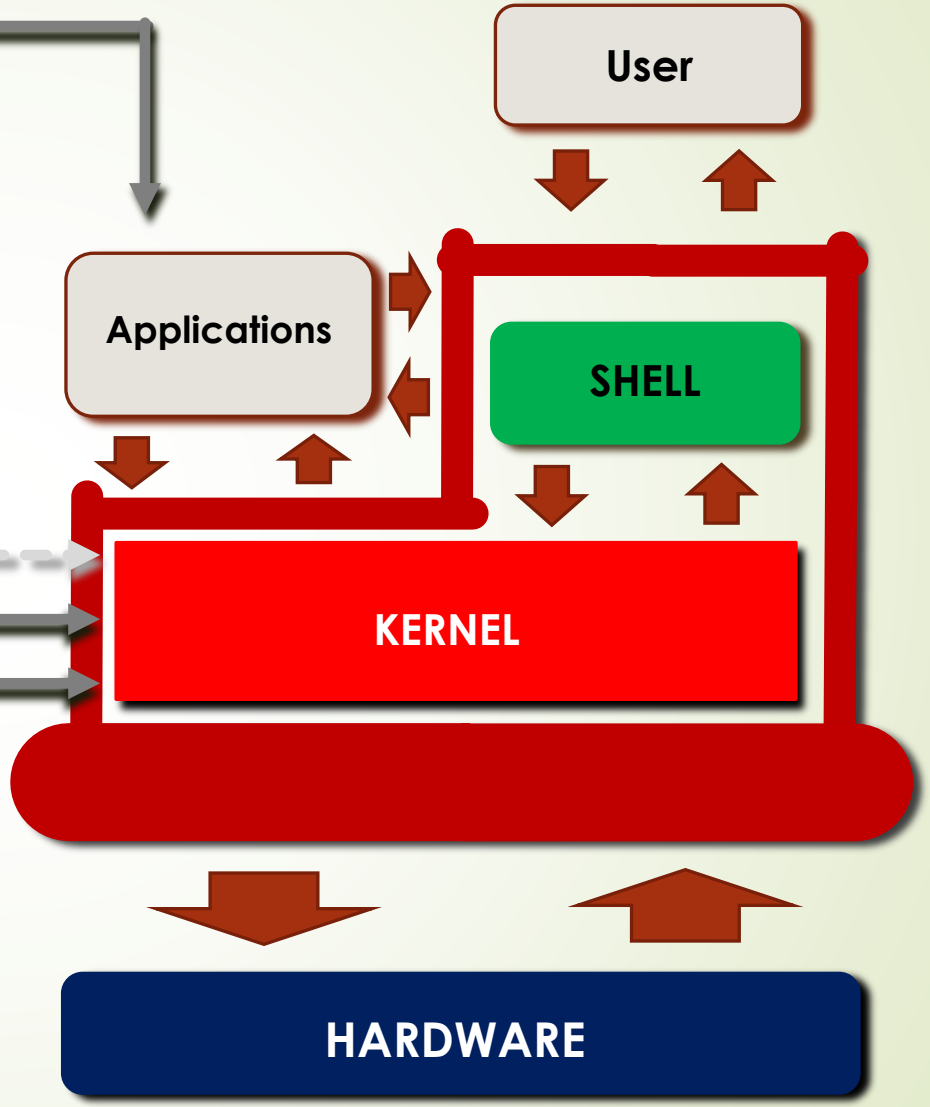
GNU LINUX

A. Tanenmbaum

'90 MINIX

L. Torvalds

1991 LINUX



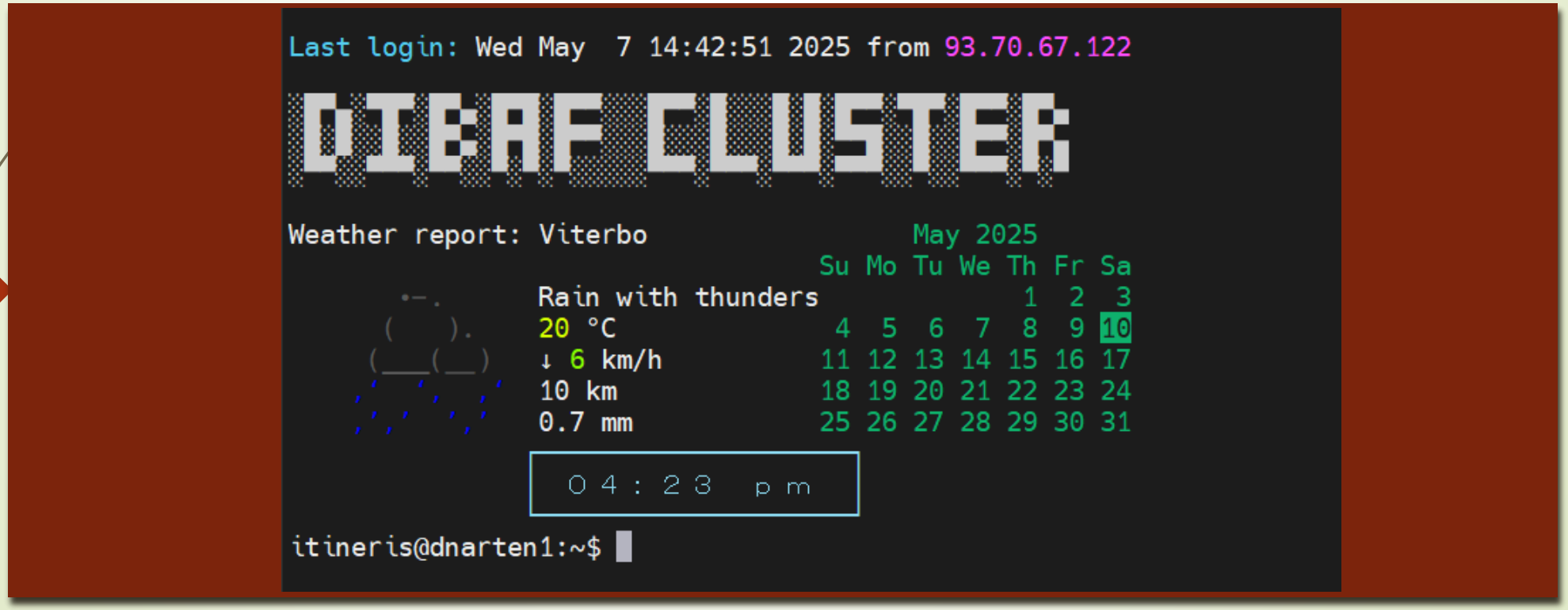
Access to a Linux System

login: itineris
Password:

To enter you are requested a *username* and *password*.
Then, after a welcome message the system shows you the *prompt* of the command interpreter: the SHELL.

% logout

At the of the session you may leave the system with the command *logout* (o *exit*).



Structure of a Unix command

name [-opz1 ... -opzn] [file1 .. filen | dir1 ... dirn]

ls –l (using a long listing format….)

Example:

```
itineris@dnarten1:~$ ls -l
total 36
drwxr-xr-x 2 itineris domain users 4096 May  7 11:23 a.conte
drwxr-xr-x 2 itineris domain users 4096 May  7 11:01 a.marchesini
drwxr-xr-x 2 itineris domain users 4096 May  7 11:00 a.montaghi
-rw-r--r-- 1 itineris domain users   67 May  4 15:43 Commands
drwxr-xr-x 2 itineris domain users 4096 May  5 17:04 c.zazza
drwxr-xr-x 2 itineris domain users 4096 May  7 11:00 g.labbri
drwxr-xr-x 2 itineris domain users 4096 May  7 10:59 l.difiore
drwxr-xr-x 2 itineris domain users 4096 May  7 11:10 p.cozzi
drwx------ 3 itineris domain users 4096 May  7 11:01 snap
itineris@dnarten1:~$
```

snap???

First Unix commands ...

```
%    whoami
c.zazza or itineris
```

Shows your name as recognized by the system (your username, $USER)

```
%    ls
elim.c          elim.dat
Fin.bmp         Host.list
Try.c           makefile
```

Shows the files under current directory

First Unix commands ...

```
%    date
Wed   Jan   24   16:44:32   MET   1996
```

```
%    pwd
/home/david/work
```

Shows the structure of *subdirectories* starting from *root directory (/)*

```
%    uname   (try also uname -a)
Linux
```

Shows the current version of the OS

First Unix commands ...

% cd

Is the command used to change your working directory.

If you use it without options, it changes your current dir to your homedir.

% pwd

```
/home/pippo/programmi
```

% cd /usr/lib

% pwd

```
/usr/lib
```

% cd

% pwd

```
/home/pippo
```

First Unix commands ...

% hostname

```
dnarten1
```

Shows the name of the Linux machine you are logged in.

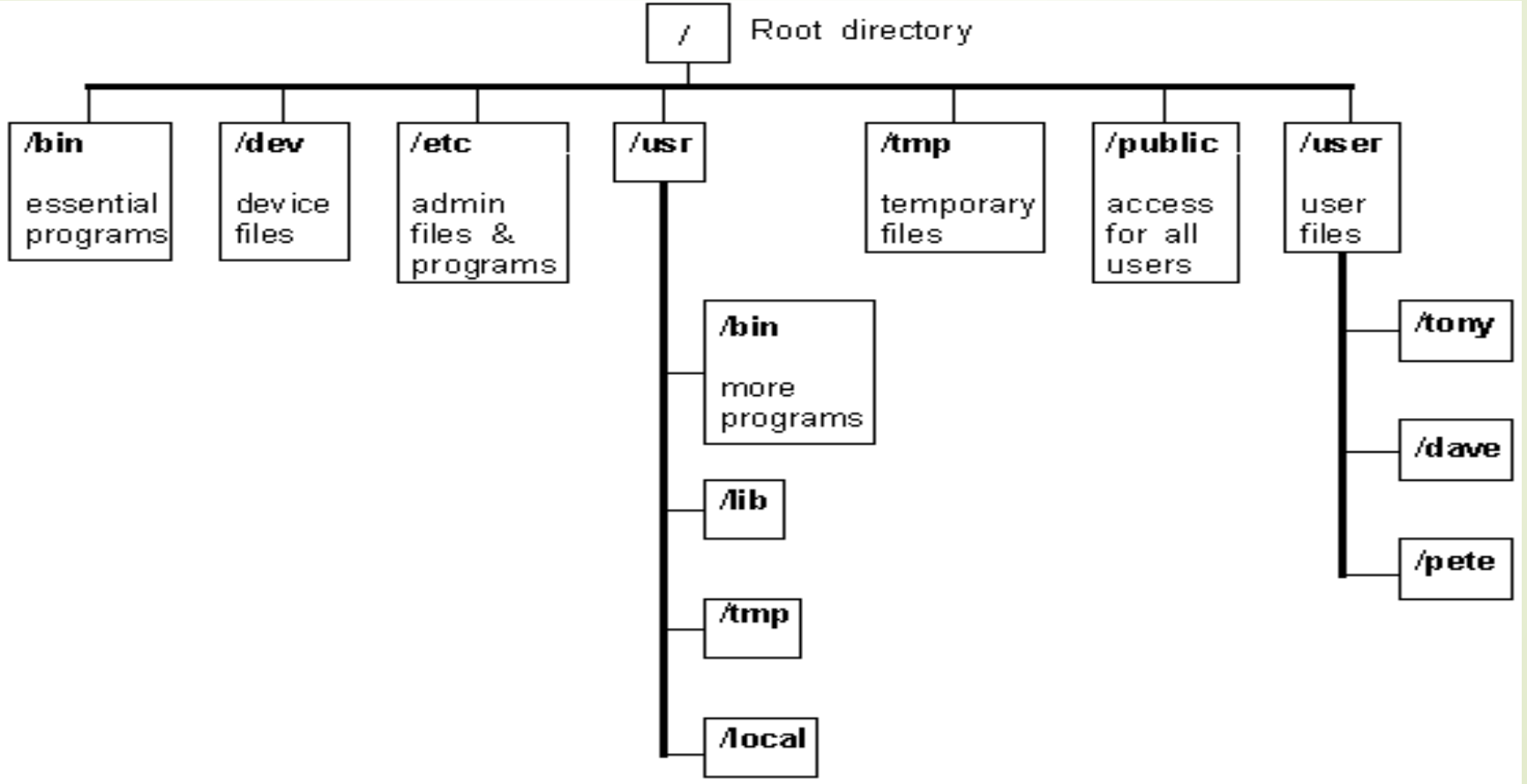
% who

```
itineris@dnarten1:~/c.zazza$ hostname
dnarten1
itineris@dnarten1:~/c.zazza$ who
c.trotta pts/0        2025-05-08 19:30 (10.2.134.30)
n.russo  pts/1        2025-05-10 15:56 (185.174.0.92)
l.gontrani pts/2        2025-05-10 16:17 (93.65.138.152)
itineris pts/3        2025-05-10 16:23 (151.27.119.40)
itineris@dnarten1:~/c.zazza$
```

Show all the users currently logged in.

The Unix *File System (directory tree)* …

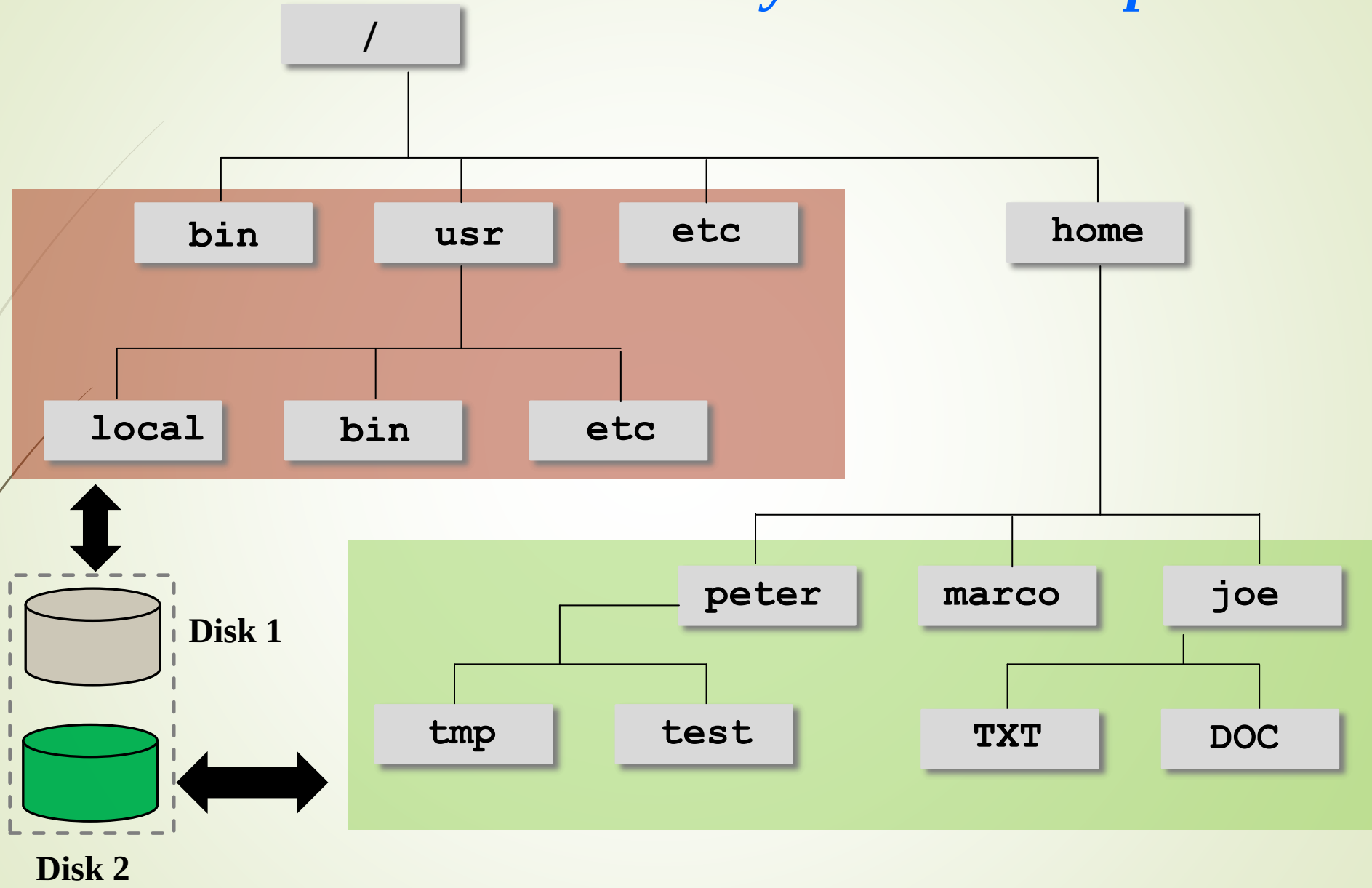
In Unix all data are organized into a hierarchical structure called ***filesystem***



... The *File System* ...

- In Unix there are, in general, many filesystems organized by functionality
- The implementation by *file system* make transparent the 1-1 relation between the physical *device* (disk, usb sticks, etc.) and the *files* they contain.
- The system accesses to a disk content by what is called a «mount point»
- A *mount point* is a directory
- Starting from a *mount point* we can access all *files* on a disk

... File System: mount point ...



... File System: mount point...

% mount

```
/dev/cciss/c0d0p7 on /      type ext3     (rw,acl,user_xattr)
/dev/cciss/c0d0p5 on /boot  type reiserfs (rw,acl,user_xattr)
/dev/cciss/c0d0p8 on /local type ext3     (rw,acl,user_xattr)
proc              on /proc  type proc     (rw)
```

File System

% fdisk -l /dev/cciss/c0d0p8

```
Disk /dev/cciss/c0d0p8: 119.9 GB, 119924550144 bytes
255 heads, 63 sectors/track, 14579 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Description of *file system* ...

% df

```
Filesystem            1K-blocks      Used Available   Use% Mounted on
/dev/md0               19840804  12859748   5956920     69% /
/dev/md1              256038140  24110516   218711780   10% /scratch
tmpfs                   4090468        24   4090444     1% /dev/shm
nfs2-psn:/DB         5859237888 2666161856 3193076032  46% /DBNFS
nfs2-psn:/progetti   5859237888 2666161856 3193076032  46% /progetti
ib-stage:/stage      7322943360 6323748448 999194912   87% /stage
AFS                     9000000           0 9000000     0% /afs
```

% du or du –h ; du -sh ?

```
7658    ./gianni/GianniData/C6H6Tesi/VOLLOC
89      ./gianni/GianniData/C6H6Tesi/CONFRONTI
317     ./gianni/GianniData/C6H6Tesi/SCATLEN
10      ./gianni/GianniData/C6H6Tesi/BASI/griglie
833     ./gianni/GianniData/C6H6Tesi/BASI/Test1
6653    ./gianni/GianniData/C6H6Tesi/BASI
```

... Description of *file system* ...

A Unix *file system* Unix has three basic structures:

*I-Node ** ; *File* ; *Directory*

* Any file or directory name has its own **i-node** which contains the reference needed to reach all the file blocks with data on the physical disk. Furthermore, the inode contains the date of last access and last modification, the creation date, file type, and many other vital infos for the data files on disk. Among those and most important are the numbers defining the User and Group Identification (UID/GID) and the file privileges (in terms of read/write/execute – rwx).

... And three kind of *files*:

- Text (ASCII format)
- Executable (scripts and binaries)
- *Special files* (link, device, ...)

... Description of *file system* ...

A *directory* may contain

- *File*
- *Directory*

- The *file "."* *(current directory)*
- The *file ".."* *(upper directory)*

```
root@bl101i28: ~ >  ls  -a
total 2436
drwx------ 18 root root   4096 Dec 23 14:15 .
drwxr-xr-x 37 root root   4096 Dec 10 15:26 ..
-rw-------  1 root root    299 Oct 20 16:25 .Xauthority
-rw-r--r--  1 root root 104671 Nov  5  2008 IOZONE.tar.gz
drwxr-xr-x  3 root root   4096 Oct 24  2008 Iozone
-rwxr--r--  1 root root    639 Nov  7  2008 multi_run.sh
-rw-r--r--  1 root root     34 Jun 18  2008 niclinkup.log
```

... Description of *file system* ...

- /
- /bin
- /usr/bin
- /dev
- /etc
- /tmp
- /usr/include
- /usr/sbin
- /usr/lib
- /usr/man
- /usr/local

The directory /etc

Contains *file* (ascii) for configuring the system (users, filesystems, network, etc.)

Files in /etc are read by system programs typically at boot time

Some of them are:

- **hosts**
- **passwd**
- **shadow**
- **group**
- **inetd.conf**
- **resolv.conf**
- **syslog.conf**
- **printcap**
- **fstab**
- **exports**

The file passwd

Contains the list of users authorized to login to system

Structure: **uid**, **gid**, **password**, **info**, **home directory**,**shell**

```
nico:x:6506:20053:Nico Sanna:/home/nico:/bin/tcsh
nrega:x:772:287:Nadia Rega:/home/nrega:/bin/tcsh
orlandoc:x:774:287:orlandoc:/home/orlandoc:/bin/tcsh
```

The *file shadow*

```
nico:ByJ6UMP/qFzOM:13931:0:99999:7:::0
nrega:/dhYiXcBcG9ZI:13950:0:99999:7:::0
orlandoc:B4TpJisqYSgyY:13931:0:99999:7:::0
```



```
root@bl101i28: ~ >  ls -lrt /etc/passwd
-rw-r--r-- 1 root root 8421 Jul 23 12:33 /etc/passwd
root@bl101i28: ~ >  ls -lrt /etc/shadow
-rw-r----- 1 root shadow 5859 Jul 23 12:34 /etc/shadow
```

The *file group*

Contains the catalog of all the users' groups

Each user must belong at least to one group

```
ar9:*:287:
ba8:*:294:
be7:*:299:feroce
by6:*:328:
gdv:x:80808:nrega,robertoi
gaussian:*:70707:afilippi,apalma,campete,coletta,dacoco,dangelo,emi
lianos,fgrandi,framondo,gbranca,ghost,girlanda,gmancini,gontrani,go
skab,lben
ci,lbenciv,mancini,nrega,orlandoc,orlcresc,ramondof,robertoi,rutigl
i,saba,tatoli,valemig,vrossi
adt:!:20471:
```

Absolute and relative PATH

Absolute *Path*:

% ls /home/david/work/my_file

```
/home/david/work/my_file
```

This *path* is valid from everywhere position within a given *file system*

Relative *Path*:

Supposing to be in *directory* /home/david

% ls work/my_file

```
work/my_file
```

This *path* is valid starting from directory `/home/david`, that is, from current directory defined as *working dir* (check with command ***pwd***).

Filenames ...

Unix is *case-sensistive*.

File identifies may contain:

‘a’ - ‘z’	**‘0’ - ‘9’**
‘A’ - ‘Z’	**‘+’, ‘-’, ‘:’, ‘_’, ‘.’, ‘ ‘**

One could group together many sets of files and directory by using *wildcard*:

‘~<username>’ – user *home dir*
‘*****’ - a sequence of 0 and more characters
‘**?**’ - a single character
[cset] - a character subset defined into squared parenthesis

... Filenames

The *shell* expands the list of characters in *cset* (one by one) using the identifier **c1**, **c2**, ..., **cn**

[c1c2c3....cn]

[c1-cn]

Examples:

[aeiou]* - All *files* beginning with a vowel

[a-d]*.txt - All *files* with *txt* extensions and beginning with a,b,c,d

file & directory permissions...

The acces to *file* e *directory* is controlled by permission on three possible operations:

- Read (**r**)
- Write (**w**)
- eXecution (**x**)

The set of permissions (**rwx)** may be set independently for

- The owner of the *file* (***user-owner)***
- The users belonging to its group (***group***)
- All the other users (***others***)

file & directory permissions...

Permission may be controlled adding an option to the `ls` command:

% ls -l

total 9

drwxrwxrwx	2	david	512	Aug	2	09:59	dati
-rw-r--rw-	1	david	486	Jan	24	14:00	fin.gif
-rwxrw-rw-	1	root	351	Jan	26	12:31	has.tcl
drwxr--r--	2	david	512	Nov	14	10:46	image
drwxrwxrwx	2	david	512	Aug	2	10:01	lib
-rw-------	1	david	5728	Jan	24	10:55	mbox
-rw-rw-rw-	1	david	565	Dec	18	15:56	pp.c
-rw-rw-rw-	1	david	5122	Dec	18	17:32	prog.tar
-rwx--x--x	1	david	213	Dec	24	23:59	natale

If the user has enough privileges, it is then possible to modify the permissions with the command **chmod**

file & directory permissions...

chmod has this structure:

chmod [-R] mode filename(s)
where **mode** could be:

A comma separated list of permissions
A octal number of 3 digits

Each permission has 3 parts:

[**ugo**|**a**] - **u**ser, **g**roup, **o**ther or **a**ll

[-|+|=]

[**rwx**] - **r**ead, **w**rite, **e**xecute

file & directory permissions...

- Comparison of permissions by letter and octal numbers

Permissions (r=4, w=2, x=1)	Octal number
- - -	0
- - -	1
- w -	2
- w -	3
r - -	4
r - r	5
r w -	6
r w r	7

- Examples:

```
root@bl101i28: ~ >  ls -l multi_run.sh
-rwxr--r-- 1 root root 639 Nov  7  2008 multi_run.sh
root@bl101i28: ~ >  chmod a+x multi_run.sh
root@bl101i28: ~ >  ls -l multi_run.sh
-rwxr-xr-x 1 root root 639 Nov  7  2008 multi_run.sh
```

Now the script *multi_run.sh* is executable by all user (*a+x*, that is, *ugo+x*)

file management ...

- **% ls -l**

```
total ...
drwxrwxrwx    2  david   512   Aug   2  09:59   dati
-rw-rw-rw-    1  david   486   Jan  24  14:00   fin.gif
-rw-rw-rw-    1  root    351   Jan  26  12:31   has.tcl
drwxrwxrwx    2  david   512   Nov  14  10:46   image
drwxrwxrwx    2  david   512   Aug   2  10:01   libraries
-rw-------    1  david  5728   Jan  24  10:55   mbox
-rw-rw-rw-    1  david   565   Dec  18  15:5    pp.c
```

- Show the list of files with some additional infos

- **% cat pp.c**

```
#include <stdio.h>
main ()
{
    printf("Hello, world!\n");
}
```

- Show the content of a single *file*

… *file management* …

- **% more *my_file***

  Give the *output* of *my_file* (a text file) divided in pages

- **% head [-n] *my_file***

  Shows the first n rows in *my_file*

- **% tail [-n] *my_file***

  Shows the last n rows in *my_file*

… *file management* …

- **% cut -f *fields* [-d *char*] *my_file***

Show selected parts of each rows of a text file.
Each row is evaluated and divided in fields separated by the <tab> character or by the character given with the option "**-d**".
Each field is selectable with the option "**-f**".

Below you have an examples where the file *passwd*, is processed using ":" as separator character and then the fields 1,5 are printed:

- **% cut -f 1,5 -d : /etc/passwd**

root:/bin/tcsh
daemon
bin
listen
talevi:/bin/tcsh
vicini:/bin/tcsh
gimenez:/bin/tcsh
panizzi:/bin

… *file management …*

- **% cp *my_file* work/*my_file.1***

  Copy file. This command creates *file my_file.1* in *subdirectory work* staring from current dir and copy the content present in *file my_file*

- **% mv *my_file* work/*your_file***

  Rename *my_file* in directory *work* e lo rename it as *your_file*

- **% rm *my_file***

  Delete the *file my_file*

… *file management* …

- **`% ln name name_of_link`**

  Creates a hard pointer (*hard link*, *i-node pairing*) *name_of_link* to *file name*. After this command, any modification to one of the two reflect automatically on the other.

- **`% ln -s name name_of_link`**

  Do as above but the link is only at the file structure (file name) level. The file *name_of_link* it not a copy of *name* but just a pointer to it. The symbolic *link* (as it is ofter called this kind of link) it is a special file with size equal to the number of character of the original file.

… *file management* …

- Some examples of *links*

```
% ls -l orig
-rw-r--r--  1 john  bb 50560 Jan 26 18:24 orig
% ln orig link
% ls -l orig link
-rw-r--r--  2 john  bb 560 Jan 26 18:24 link
-rw-r--r--  2 john  bb 560 Jan 26 18:24 orig
% ln -s orig link_s
% ls -l orig link*
-rw-r--r--  2 john  bb 560 Jan 26 18:24 link
lrwxr-xr-x  1 john  bb   4 Jan 28 15:25 link_s -> orig
-rw-r--r--  2 john  bb 560 Jan 26 18:24 orig
```

… *file management* …

- **% file fin.***

```
fin.bmp:        data
fin.c:          c program text
```

Give info on the kind of *file* passed as argument

- **% find . -name *my_file* -print**

Search *my_file* starting from current directory (.) and, if succeded, print the relative *path* where *file* was found.

- **% find . -type d -mtime +3 -exec rm -r {} \;**

Search starting from current directory (.) all the subdirs not modified since 3 days and delete them. **BE CAREFUL with rm –r ☺**

… *file management* …

% lpr -P*printer* *my_file*

Print *my_file* on printer called *printer*

% mkdir *my_dir*

Create *directory my_dir* into current *directory*

% rmdir *my_dir*

Delete, if empty, the *directory my_dir.*

If the *directory* is not empty we must use the option -r :

% rm -r *my_dir*

BE CAREFUL with rm –r ☺

File compression ...

One may reduce the space occupated by a file(s) on disk by compressing it.

Compressor and Expander are given in couple. See some of them where each couple refers to a different algorithm of compression:

pack, **unpack**
compress, **uncompress**
gzip, **gunzip**

% pack *file1 ... filen*

Each *file* in the list is compressed and included into file *filex.z*

% compress *file1 ... filen*

Each *file* in the list is compressed and included into file *filex.z*

% gzip *file1 ... filen*

Each *file* in the list is compressed and included into file *filex.gzip*

tar command ...

The Tape Archive (tar) tar command is used to *backup* group of *file* and *directory.*
The *backup* could be saved on file, tape, etc. and the command has many options:

% tar *flags* [*file*] ...

***flags* :**

r : append *file* at the end of archive
x : extract *file* from archive
t : examine the archive content
u : add to archive only modified files
c : create a new archive
f : archive filename (the name should be given soon after the –f)
v : show to *std output* the list of the files in archive

The command by default do not use compression. If one add the flag "**z**" then the files are compressed with gzip before append them to archive.

… tar command …

Examples:

% tar cf /tmp/lib.tar /usr/lib

Generate an archive in /tmp/lib.tar with the content starting from /usr/lib.

% tar cv pippo/images

Create and archive on the default device (typically a tape) of all the file and directories present in pippo/images. Note that *path* is relative to current *directory*.

% tar zcvf my_archive.tgz /home/nico

Create a gzip compressed tar archive `my_archive.tgz` of `/home/nico` content.

% tar zxvf my_archive.tgz

Extract the content of `my_archive.tgz` into current directory.
Note that, the complete path recorder in archive will be rebuild starting from current dir.

% tar ztvf my_archive.tgz

Show at terminal the content of `my_archive.tgz`.

Redirections ...

By definition Unix programs/commands open 3 *file* .

standard (std) *input* (keyboard)
standard (std) *output* (video)
standard (std) *error* (video)

It is always possible to redirect the *input* and *output* streams on any other (file) device using the characters

<	The *std input*
>	The *std output*
>>	Append the *std output*
>&	*std output and std error*

... redirections

Examples:

```
mail   stefano   <   mail.stefano
ls   -l   >   elenco_files
myprog   >&   out_err.log

who   >   utenti
myprog   <   utenti   >>   myout
```

On can make a pipeline of two (or more) commands by connecting the *output* del primo con *input* del secondo, connecting the two commands with a *pipe* (|):

```
who   |   myout
grep   stefano   list   |   sort   |   lpr
find   /   -name   "*.conf"   -print   | sort   >   conf.txt
```

On line help of Unix commands ...

% man who

WHO(1) USER COMMANDS WHO(1)

NAME
who - who is logged in on the system
SYNOPSIS
who [who-file] [am i]
DESCRIPTION
Used without arguments, who lists the login name,terminal name, and login time for each current user. who gets this information from the /etc/utmp file.
.....

EXAMPLES
example% who am i
.....

FILES
/etc/utmp
/var/adm/wtmp

SEE ALSO
login(1),w(1),whoami(1),utmp(5V),locale(5)

… On line help of Unix commands ...

- Pros:
  - All Unix command are documented
  - It is quite easy to write *man page* of applications (nroff)
  - Any *man page* has many references to other commands.
  - The docs include also libraries, compilers, tools, etc.
- Cons:
  - You must know in advance the name of the command
  - Could be cryptic
  - Could be slow

... On line help of Unix commands

% apropos user

```
.....
dirt  (1l)          -  The Dirt User Interface Builder
groups  (1L)        -  print the groups a user is in
listalias  (1L)     -  list user and system aliases
logname  (1L)       -  print user's login name
.....
whoami  (1L)        -  print effective userid
```


The command *apropos* gives a short description of all commands referring to a string given as argument.

The command interpreter (*Shell*) ...

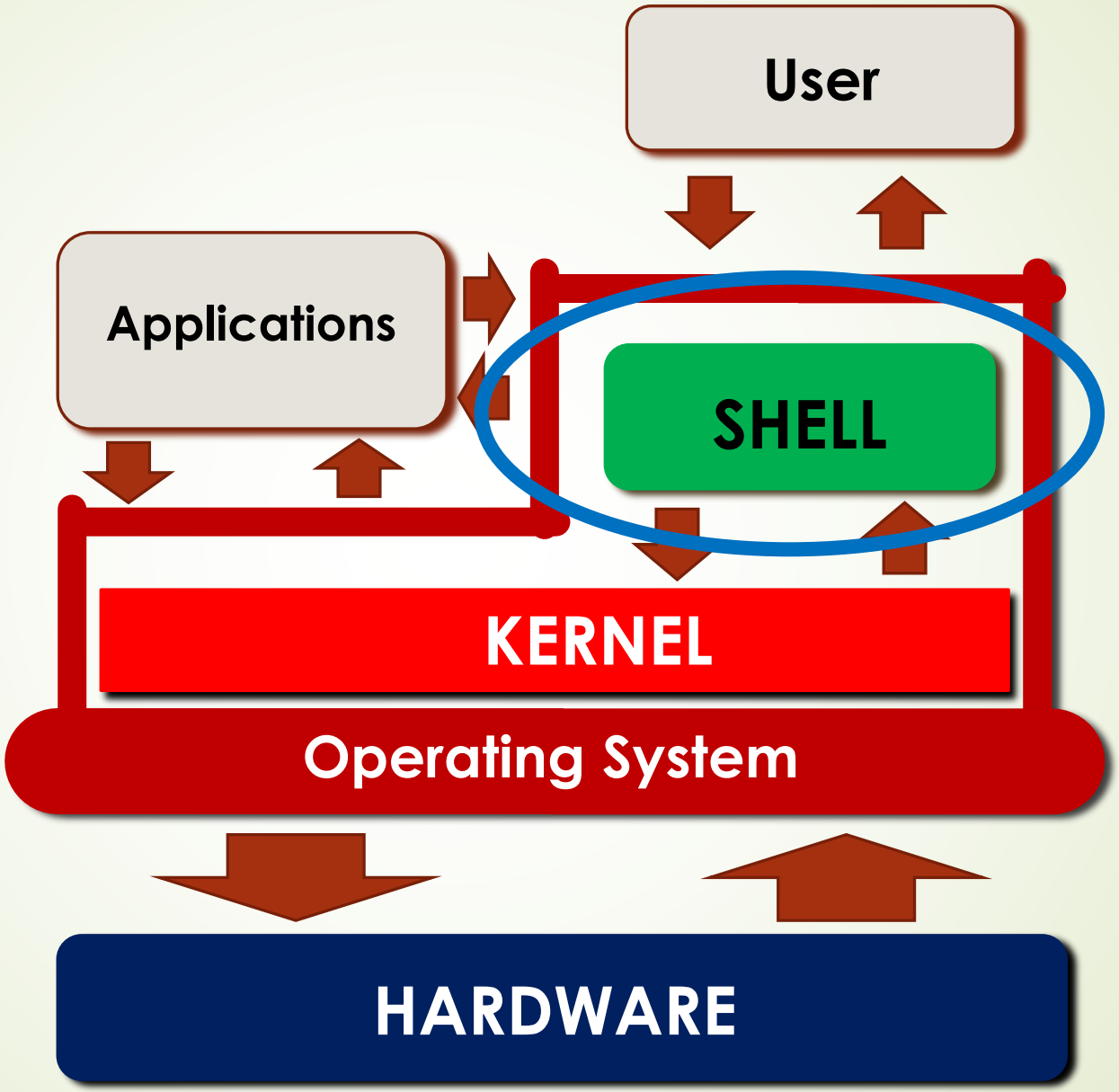
The SHELL is the interface between the Operating System and USER.

The shell decodes the characters given in input and run the system executable(s) corresponding to the command.

Historically, there were many *shells* with different features:

- *Bourne shell* *sh*
- *C shell* *csh*
- *Tcsh shell* *tcsh*
- *Korn shell* *ksh*
- ***Bash shell*** ***bash***
- *Z shell* *zsh*

Unix structure



... The command interpreter (*Shell*) ...

- The login *shell* is assigned by the administrator, but user can change it.
- The usage of one or different shell strongly depend on the users' habits and on the kind of processing she/he carry on every day.
- There are a plethora of *shells* best suited for a specific activities (i.e., tcl/tk for X11 graphic interface)

... The command interpreter (*Shell*) ...

- Long Unix commands may be shortened with the *aliasing*:

% alias dir ls -aF

- You may digit just the first character of a command and using <*Tab*> to complete it.

- Commands may be executed on multiple lines:

% ls
% cd ~

- ... Or on just one line separated by ";"

% ls; cd

Environment Variables

Envars (as often are called environment variable) are predefined words any user will have configured by default.
The value/content of these variables are predefined by shell in two steps:

1. At login time *(.profile* in your homedir using bash)
2. Any time you start a new shell *(.bashrc* in your homedir using bash)

HOME	LD_LIBRARY_PATH
SHELL	PROMPT
PATH	HOSTTYPE
PWD	MANPATH
HISTORY	

The vim Editor

The standard in *nix systems

In this chapter …

- `ed`, `ex`, `vi`, `vim`
- `vim` basics
- Command Mode
- Input Mode
- Last Line Mode
- Buffers
- Yanking

In the beginning …

- There was `ed` … single line editor
- Then came `ex` … had a nifty visual mode
- Visual mode leads to `vi`
- Written by Bill Joy (BSD, csh, Sun) in 1976
- `vi` a Unix utility – so we need a free clone
  - `elvis`, `nvi`, `vile`, and `vim`

vim

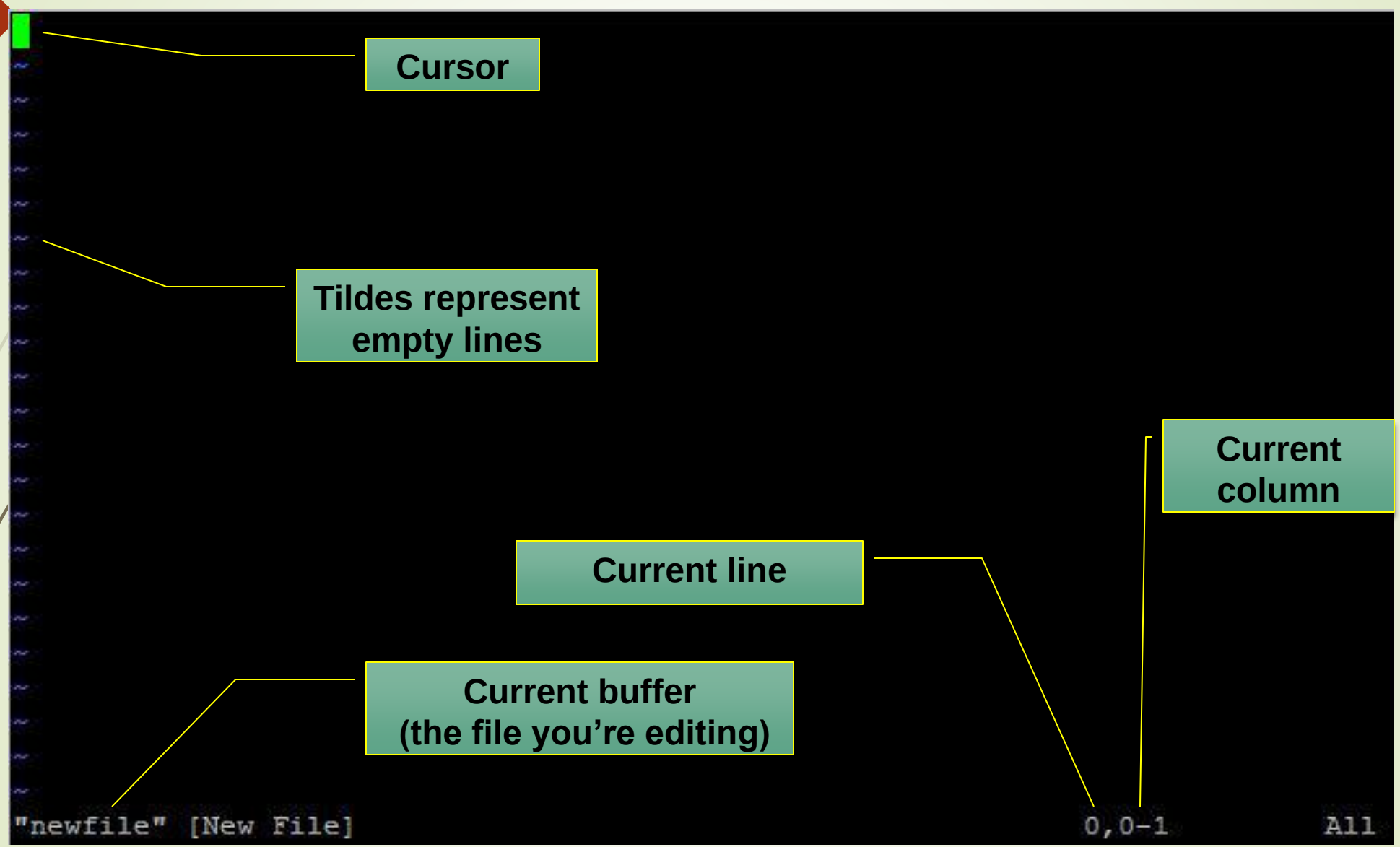
- We'll be using `vim` - vi improved
- Written by Bram Moolenaar
- In our Ubuntu distro, we have `/usr/bin/vim`
- `vi` is just an alias to `vim`
- Standard in just about all Linux distros
- Available from www.vim.org
  - gVim, multi-platform

vim con't

- Powerful, quick text editor
- Excellent for programming due to intelligent language detection
- NOT a formatting tool … plain text only
- Nearly limitless options and commands
- Excellent tutorial - `vimtutor`

Starting vim

- Syntax: **`vim [options] [filename]`**
- Filename is optional
  - With it, it opens that file for editing
  - Without, it opens a default screen
- Many options available, most commonly used ones are for file recovery


Cursor
Tildes represent empty lines
Current column
Current line
Current buffer (the file you're editing)
"newfile" [New File]
0,0-1
All

How it works

- vim copies the contents of the file you want to edit into memory
- This memory is referred to as the Work Buffer
- Changes are made to the buffer, not the file
- You must write changes to file when done editing the buffer

Modes

- `vim` has three modes
  - Command
  - Input
  - Last Line
- When you start `vim`, you begin in Command Mode by default
- Hitting `ESCAPE` will get you back to Command Mode from other modes

Command Mode

- Default mode
- Used to enter commands
  - Text manipulation
  - Change modes
  - Save/exit
- Most commands are just alpha characters, not control sequences
- Case sensitive!

Insert Mode

- The mode that lets you edit and enter text
- Several sub-modes
  - Insert
  - Append
  - Open
  - Replace
  - Change
- You'll spend most of your time here

Last Line Mode

- From command mode press :
- Cursor jumps to the last line on the screen
- Here you can manage files, issue shell commands, change editor settings
- Also where you go to exit

Getting into Input Mode

- `i` nsert before cursor
- `I` nsert before first nonblank character on line
- `a` fter cursor
- `A` t end of line
- `o` pen line below
- `O` pen line above
- `r` eplace current character
- `R` eplace characters

Command Mode - Essentials

- `h` move cursor left
- `j` move cursor down
- `k` move cursor up
- `l` move cursor right
- `x` delete character
- `dw` delete word
- `dd` delete line
- `ZZ` write and quit

Command Mode con't

- `/`*`regexpr`* search forward
- `?`*`regexpr`* search backwards
- `n` repeat last search (ie, find next result)
- `N` repeat last search, in opposite direction
- *`n`*`G` Jump to line *n* (omit *n* to go to last line)

Last Line Mode Essentials

- `w` write file
- `q` quit
- `w!` write read-only file
- `q!` quit without saving changes
- `e` *`filename`* opens a file for editing

Last Line Mode con't

- `sh` open a shell
- `!`*`command`* open a shell, run a command, then exit the shell
- `.!`*`command`* open a shell, run a command, exit the shell, placing the standard output into the work buffer
  - Can also do `!!`*`command`* from Command Mode

Buffers

- Work buffer
- General Purpose Buffer – kind of like the clipboard in Windows
- Named buffers
- Numbered buffers

General Purpose Buffer

- Contains recently edited or deleted text
- It's where undo information is stored
- You can copy (yank) text to this buffer and then paste (put) it elsewhere

Named Buffers

- Similar to the General Purpose Buffer
- Does not contain undo info – only contains text if you put it there
- Each of the 26 buffers is referenced by letter a-z

Numbered Buffers

- Numbered 1-9
- Read only
- Contain most recently deleted chunks of data greater than one line long
- You can paste (put) from these buffers and use them for undoing deletes

yank

- Copies lines of text
- To yank a line, use `yy`
- Or use `Y` – it's shorter
- To yank multiple lines, place cursor on the first line and use *n*`Y`, where *n* is the number of lines to yank

yank con't

- By default it yanks text to the General Purpose Buffer
- To place in a named buffer, precede the yank command with double quotes and the letter of the buffer you wish to use
- Use lowercase letter to overwrite, upper case to append
- Ex: `"c5Y` would yank 5 lines to buffer c

put

- Pastes text from a buffer into the Work Buffer
- Use `p` to put below current line
- Use `P` to put above current line
- Again, if using a named buffer, precede with double quotes and the letter

vim

- Just barely scratching the surface
- Hundreds of commands
- Command modifications
- We'll cover searching and substituting in Appendix A

Unix processes ...

Program: a set of instructions the computer decode and execute

Process: is a program in execution

Each process has its own number identifying it:
Process Identifier (pid)

Since Unix is a multi-tasking OS it execute many processes concurrently

Any running process has at least three open files:
stdin, stdout e **stderr**

… Unix processes …

A single user may have many process running at the same time

The execution of a process could happen in two ways:

foreground Any character striked on keyboard is read as stdin of the process

background The process is NOT controlled by stdin activity

It is permitted to run just ONE *foreground* process but many *background* processes.

… Unix processes …

The main command to check process status is **ps**:

% ps

```
  PID TTY          TIME CMD
17059 pts/0    00:00:00 bash
17083 pts/0    00:00:00 ps
```

% ps -ef

```
root      3387  3386  0  2010 ?        00:00:00 /usr/libexec/courier-authlib/aut
root      3673     1  0  2010 ?        00:00:12 /sbin/syslogd -a /var/lib/ntp/de
root      3676     1  0  2010 ?        00:00:00 /sbin/klogd -c 1 -2 -x
nobody    3703     1  0  2010 ?        00:00:00 /sbin/portmap
root      3706     1  0  2010 ?        00:00:00 /sbin/resmgrd
```


The following built-in command of bash is used to check the status of your backgroud processes:

% jobs

```
[1]  +  Suspended        vi  pp.c
[2]     Done             find  .  -name  xyz  -print
```

… Unix processes …

All Unix processes are able to receive *signals* at runtime

A *signal* is a construct to permit inter-process communication

When a process receive a signal:

1) The OS start a default action for a given signal

2) The process *signal handler* manage the signal received

… Unix processes …

Name	#	Action
HUP	1	Hangup - Termina l'esecuzione
INT	2	Interrupt - Termina l'esecuzione (Ctrl-c)
QUIT	3	Quit - Termina l'esecuzione (Ctrl-c)
KILL	9	Kill - Termina l'esecuzione in modo incondizionato
SEGV	11	Segmentation Violation
TERM	15	Terminate - Termina l'esecuzione in modo...
STOP	17	Stop - Arresta l'esecuzione in modo incondizionato. Si prosegue con CONT
TSTP	18	Stop - Arresta l'esecuzione immediatamente Si prosegue con CONT (Ctrl-z)
CONT	19	Continue - Riprende l'esecuzione dopo STOP o TSTP
CHLD	20	Child - Lo stato di un processo "figlio" e' mutato
USR1	30	User-defined Signal

Daemons ...

Are processes running in *background*

used by OS to manage all system services

They may be temporary or permanent in the OS but generally

Start at boot time and remain active until next *shutdown*

Start when the service they refer to is requested and stop when the service is completed

Daemons are programs defined as standalone

... daemons ...

A Unix system service is managed by one or more daemons

Examples:

Printer : **lpd**

email: **sendmail**

Remote login: **sshd**

File transfer: **ftpd**

Authentications : **ypserv, ypbind, yppasswdd**

File share, NFS : **nfsd, rpc.mountd, rpc.statd, ...**

... daemons ...

init

Is the “father” of all processes: PID=1

Start at machine boot and read /etc/inittab

Kills all *zombie* processes

cron

Execution of commands periodically

inetd

Manager of the network services

Start and terminate (on request the services (daemons) given in /etc(inetd.conf

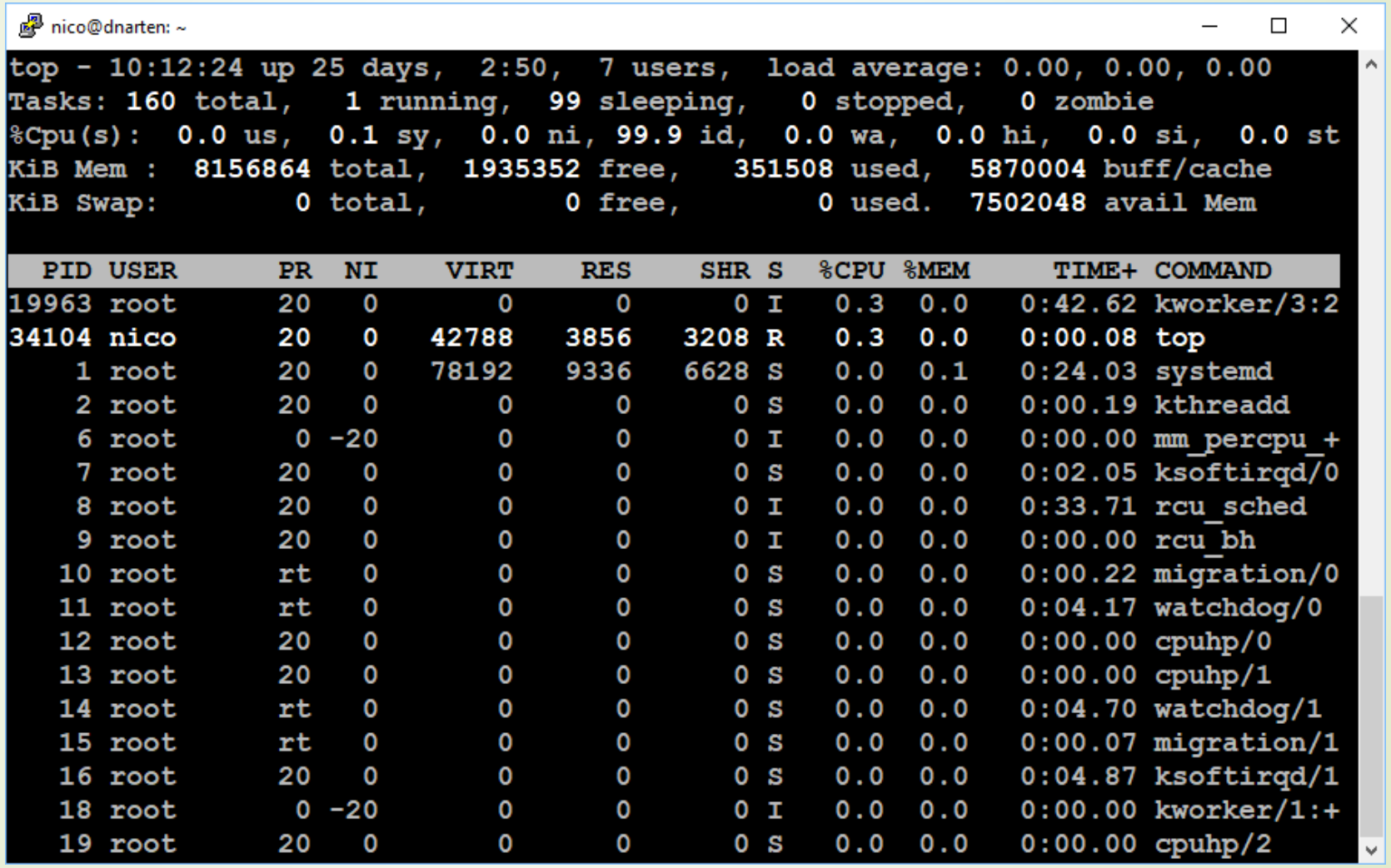
... daemons ...

% ps auxw

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0    596   236 ?        S     2010   0:04 init [3]
root          2  0.0  0.0      0     0 ?        S     2010   0:02 [migration/0]
root          3  0.0  0.0      0     0 ?        SN    2010   0:00 [ksoftirqd/0]
root          4  0.0  0.0      0     0 ?        S     2010   0:00 [migration/1]
……..
……
nobody    16289  0.0  1.9 25124 19744 ?        S    20:50   0:08 httpd -k start -DSSL
nobody    16549  0.0  1.9 25164 19768 ?        S    21:44   0:04 httpd -k start -DSSL
nobody    16648  0.0  1.8 24888 19496 ?        S    21:58   0:04 httpd -k start -DSSL
nobody    16683  0.1  1.8 24892 19548 ?        S    22:01   0:05 httpd -k start -DSSL
nobody    16720  0.0  1.9 25460 19996 ?        S    22:06   0:04 httpd -k start -DSSL
postfix   17084  0.0  0.1  4696 1476 ?         S    23:12   0:00 pickup -l -t fifo -u
nobody    17139  0.1  1.8 24524 19096 ?        S    23:18   0:00 httpd -k start -DSSL
root      17158  0.0  0.0  2372  704 pts/0     R+   23:20   0:00 ps auxw
```

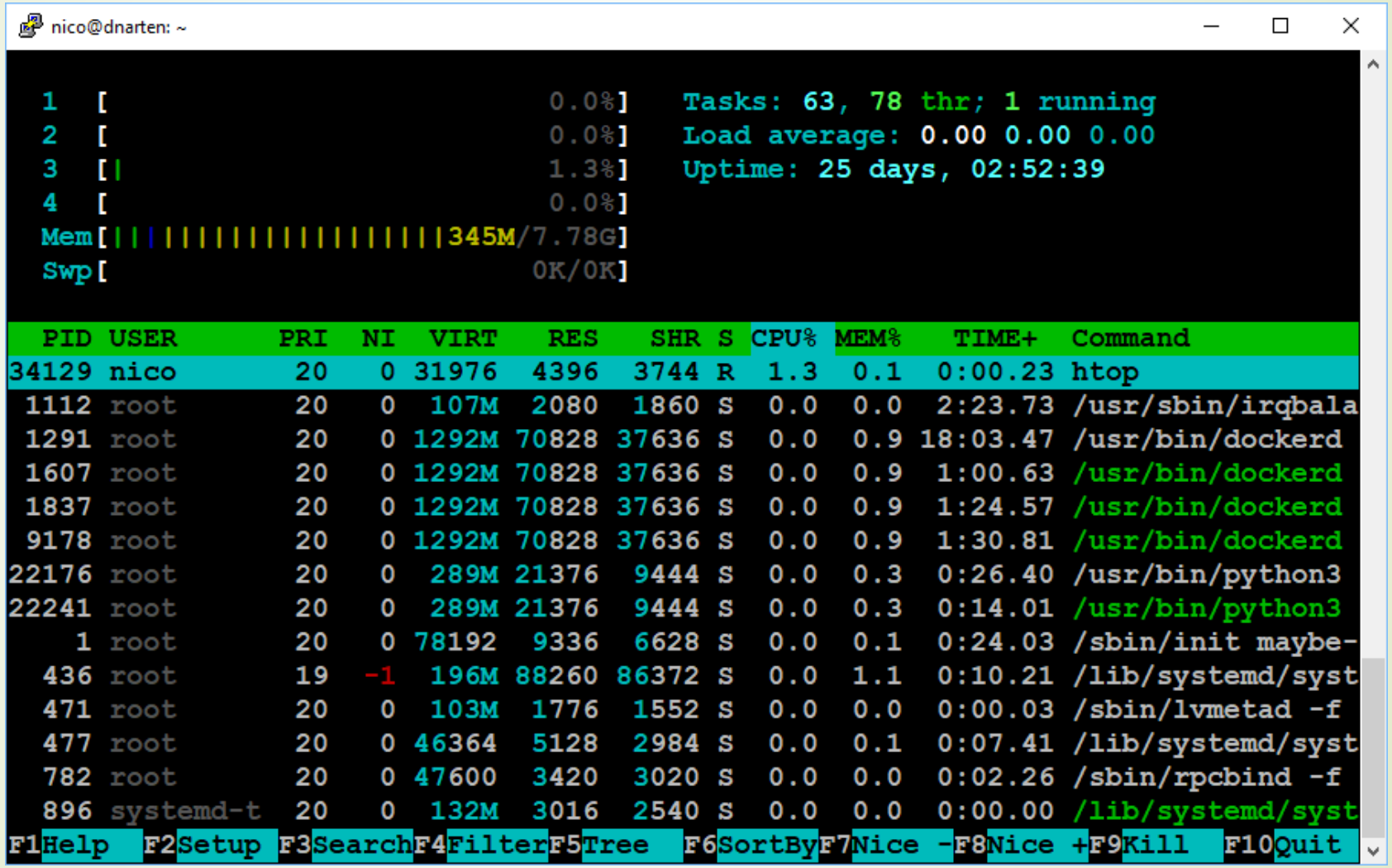
… top command …

- **TOP** show all active processes on system you are logged in.



... htop command …

- **HTOP** show all active processes on system you are logged in.



Virtual Memory ...

- The applications (embedded in processes) do not «see» the physical RAM memory but they work on «Virtual Memory» (VM)
- VM > RAM
- Unix will manage the swap between virtual and real memory addresses
- When RAM is not enough, Unix will use disk space (*swap space*)
- swap : is a memory much slower than RAM
- Swap space has a less complex file structure with respect to a Unix file system, thus resulting on a lower i/O efficiency.

… Virtual Memory …

- VM is divided into pages of fixed size (typically 4/8 *Kb* per page)
- Page size depends on the OS architecture (32/64 bits)
- The core of Unix, the *Kernel*, keep track on a dynamic table (*Free list*) of the pages in RAM memory eligible to be released

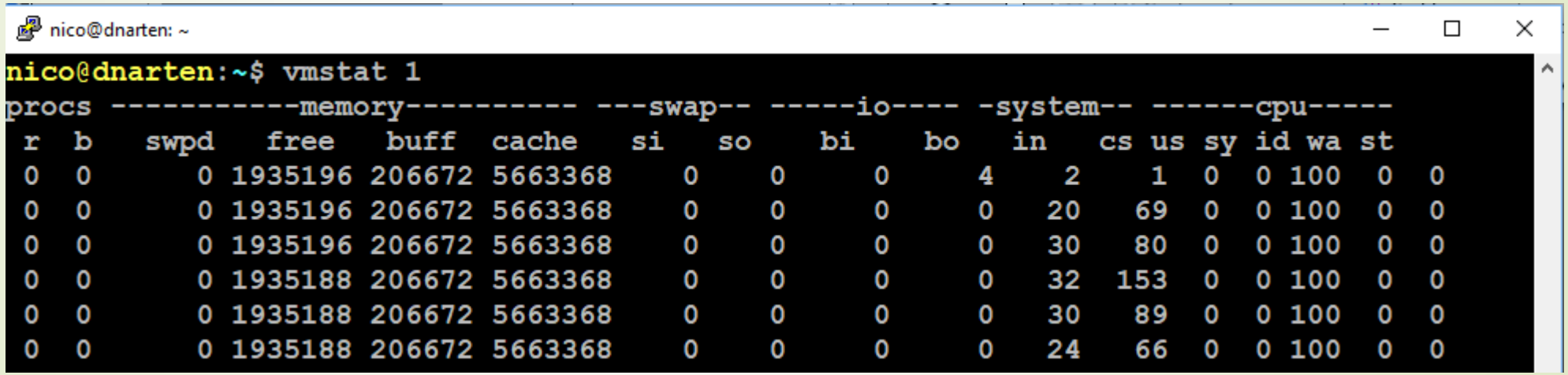
… Virtual Memory …

The *vmstat* command give some useful info on the VM usage

vmstat <intervallo temporale>

Example

% **vmstat 5**



Networking…

- Together with disk and memory, the **network** is a device of vital importance in modern OS and we are approaching the point when ALL applications (and OS) will not work without it.
- UNIX machines have built-in the support for many network protocols the most important of which is TCP/IP.
- At present TCP/IP is implemented on all the OSes used everyday.

Networking...

The protocols TCP/IP (***T**ranfer **C**ontrol **P**rotocol / **I**nternet **P**rotocol)* makes viable the exchange of information and data independently from the computer hardware and the network media used (optic fibre, satellite, etc.).

Networking...

- Data send through network are divided into messages an those into packets of small size, of the order 1 KB (1536 B for ethernet).
- Network protocols are made out of *layers*
- Each layer is responsible of one aspect of the communication

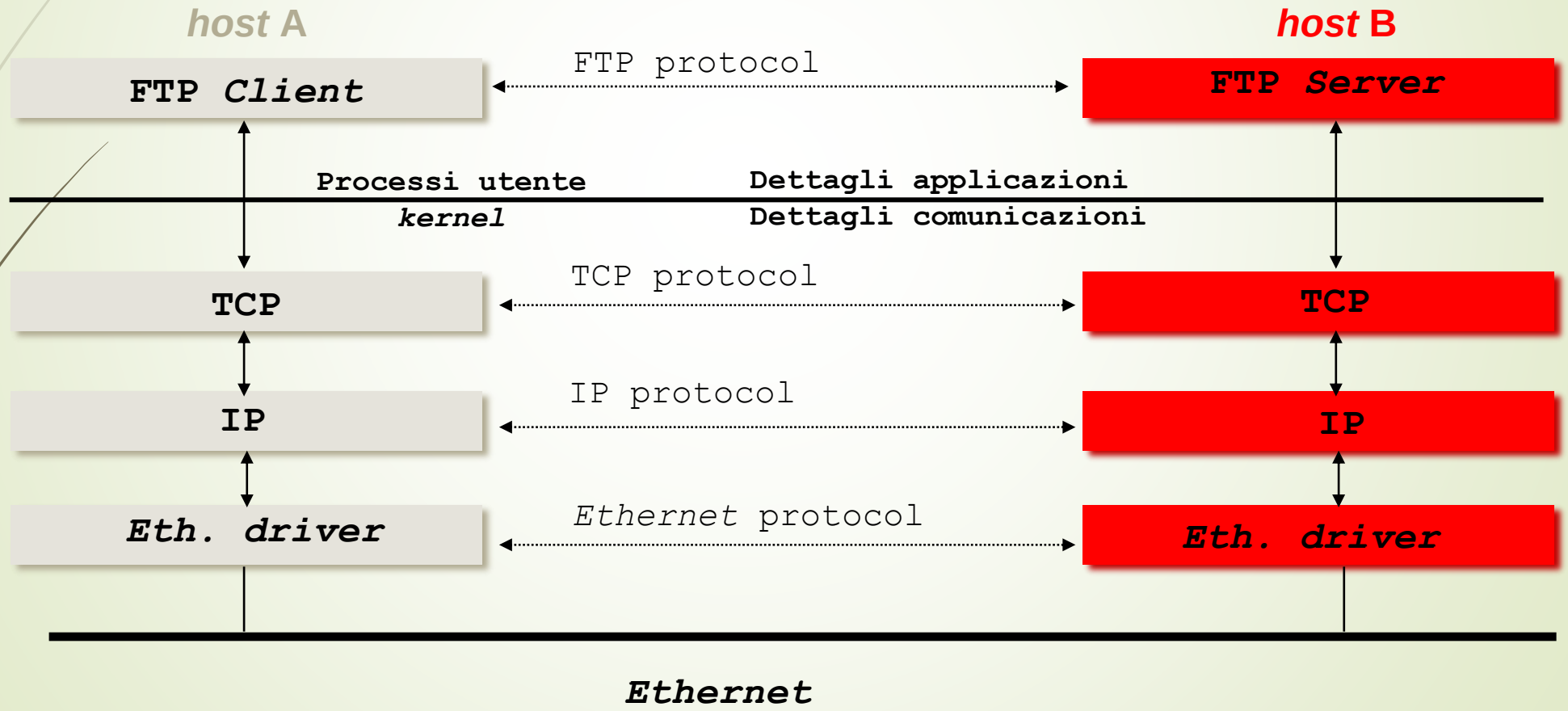
The TCP/IP refers to the lower 4 layers of the ISO/OSI model:

- *Hardware*
- *Network* - *instradamento pacchetti*
- *Transport* - servizi di trasferimento dati tra due *host* a beneficio delle applicazioni
- *Application*

Application	Telnet, FTP, e-mail,etc
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Hardware	Device driver e schede rete ARP - RARP

Networking...

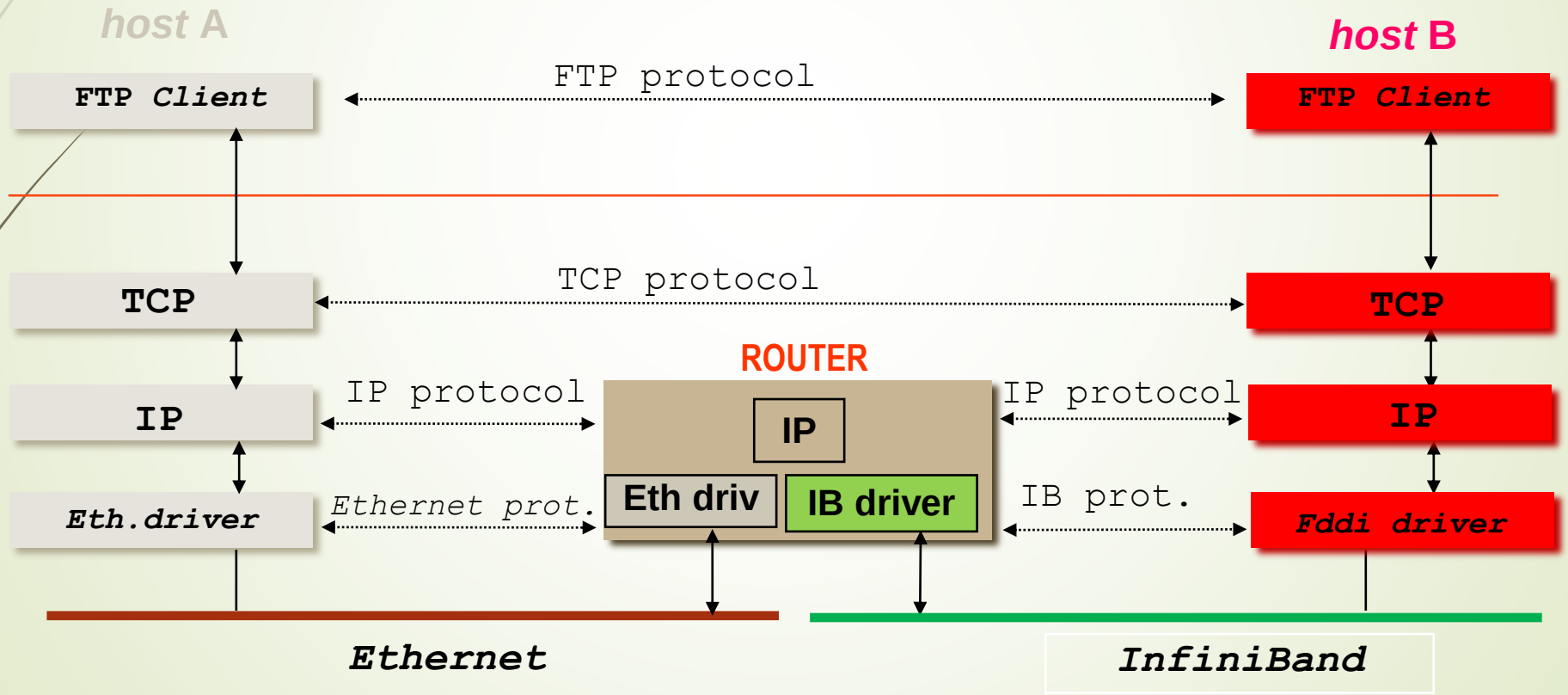
- (*Local Area Network*) *LAN connection . I.e., ethernet*
- Direct connection



Networking...

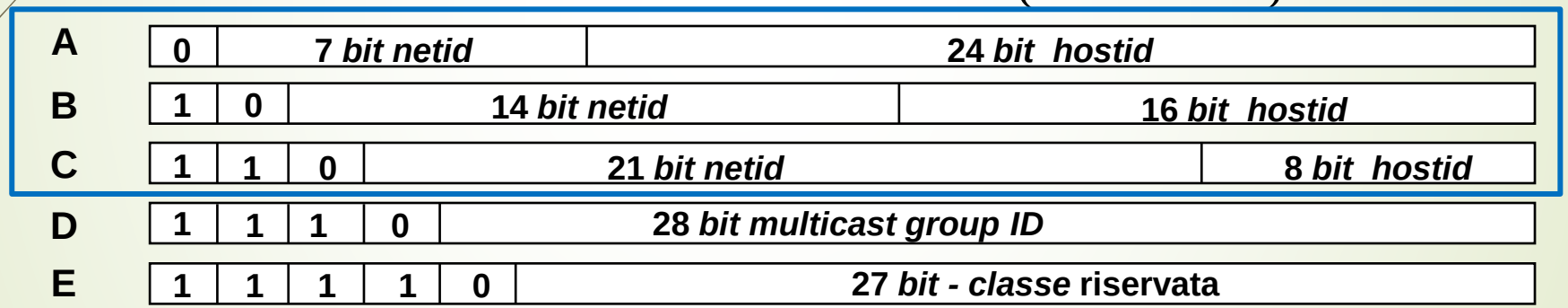
- Connection between two networks (***InterNetwork***)
- We need a bridge a L1 and a management device at L2 called ***router***.

(Modern router also act as bridge among many different network media).



Networking...

- Any *computer* when connected, is assigned a unique numeric number made out of 4 bytes, the IP address, i.e., 193.205.145.87
- A part of the address identify the network (***netid***), the rest of the numeric address the *computer* (***hostid***)
- There are 5 classes of different *internet* address (TCP/IP V4)



Class	Range
A	0.0.0.0 - 127.255.255.255
B	128.0.0.0 - 191.255.255.255
C	192.0.0.0 - 223.255.255.255
D	224.0.0.0 - 239.255.255.255
E	240.0.0.0 - 247.255.255.255

Networking...

- For practical reasons, the numeric addresses are organized into tables with a 1-1 correlation with a named address divided into *hostname*.*domainname*:

  193.205.145.87 ⟷ narten.unitus.it

- The TCP/IP ***D**omain **N**ame **S**ystem* (DNS) it's the *database* (distributed worldwide) giving anywhere/anytime the translation from numeric to domain address.

- You may contact your **DNS** via the command **nslookup**

```
nico@dn04: ~
nico@dn04:~$ nslookup narten.unitus.it
Server:         192.168.100.102
Address:        192.168.100.102#53

Non-authoritative answer:
Name:   narten.unitus.it
Address: 193.205.145.87
```

UTILITIES

grep make a search of string in *file*

grep mystring myfile

grep mystring *.c

wc counts row, words and characters in a *file*; with "**-**l" counts just the lines

wc -l myfile

sort sort out rows in your file in order. By default it uses alphabetic sorting but also numeric sort is available. Many useful options to sort with respect to many columns present in your file.

diff shows the differences between two files

bc is a simple calculator with many built-in mathematical functions

Secure Shell (SSH)

https://www.cs.clemson.edu/course/cpsc420/presentations/Spring2007/ssh.ppt

What is SSH?

- "SSH is a protocol for secure remote login and other secure network services over an insecure network." – RFC 4251
- Secure channel between two computers
  - Provides data confidentiality and integrity
- Many uses other than remote shell

Implementations

- OpenSSH – common on UNIX systems
- SSH Tectia – commercial implementation
- PuTTY – client only, Windows
- MobaXterm
- MindTerm – client only, Java applet
  - And many, many other!!!

Layering of SSH Protocols

- Transport Layer Protocol
  - Provides server authentication, confidentiality, and integrity
- User Authentication Protocol
  - Authenticates the client-side user to the server
- Connection Protocol
  - Multiplexes the tunnel into logical channels
- New protocols can coexist with the existing ones

Transport Layer Protocol

- Public-key host authentication
  - Lets the client know the correct server is on the other end
  - DSS or RSA, raw or through OpenPGP
- Strong symmetric encryption
  - Uses Diffie-Hellman algorithm for secure key exchange
  - Many ciphers are supported: 3des, blowfish, twofish, aes, etc., most with multiple key sizes
  - New keys generated every 1 GB or 1 hour
- Data integrity via MACs (message authentication codes)
  - SHA-1 and MD5 are supported

User Authentication Protocol

- Multiple authentication methods
  - public-key, password, host-based
  - Extensible
- Server tells client which methods can be used, client picks the most convenient
- Provides a single authenticated channel to the connection protocol

Connection Protocol

- Provides multiple channels:
  - interactive login sessions
  - remote execution of commands
  - forwarded X11 connections
  - forwarded TCP/IP connections
- All channels are multiplexed into a single encryption tunnel

User Configuration Files (OpenSSH)

- ~/.ssh/
  - id_* - private authentication keys
  - id_*.pub – public authentication keys
  - known_hosts – list of known public host keys
  - authorized_keys – list of allowed public authentication keys

Public-Key Authentication Howto

```
$ ssh-keygen -t rsa
...
$ cat ~/.ssh/id_rsa.pub | ssh <remote-host> 'cat - >> ~/.ssh/authorized_keys'
...
$ ssh <remote-host>
...
```

Accept the defaults and leave the passphrase blank

Enter your password one last time

Enjoy not having to enter a password

References and Resources

- RFC 4250-4254
- SSH: The Secure Shell – *The Definitive Guide*
  - http://www.snailbook.com/index.html
- http://en.wikipedia.org/wiki/Secure_Shell
- http://www.cs.clemson.edu/~duckwos/ssh_lab/