

Unix Operating system

Embedded Real Time Systems
Prof. Davide Brugali
Università degli Studi di Bergamo

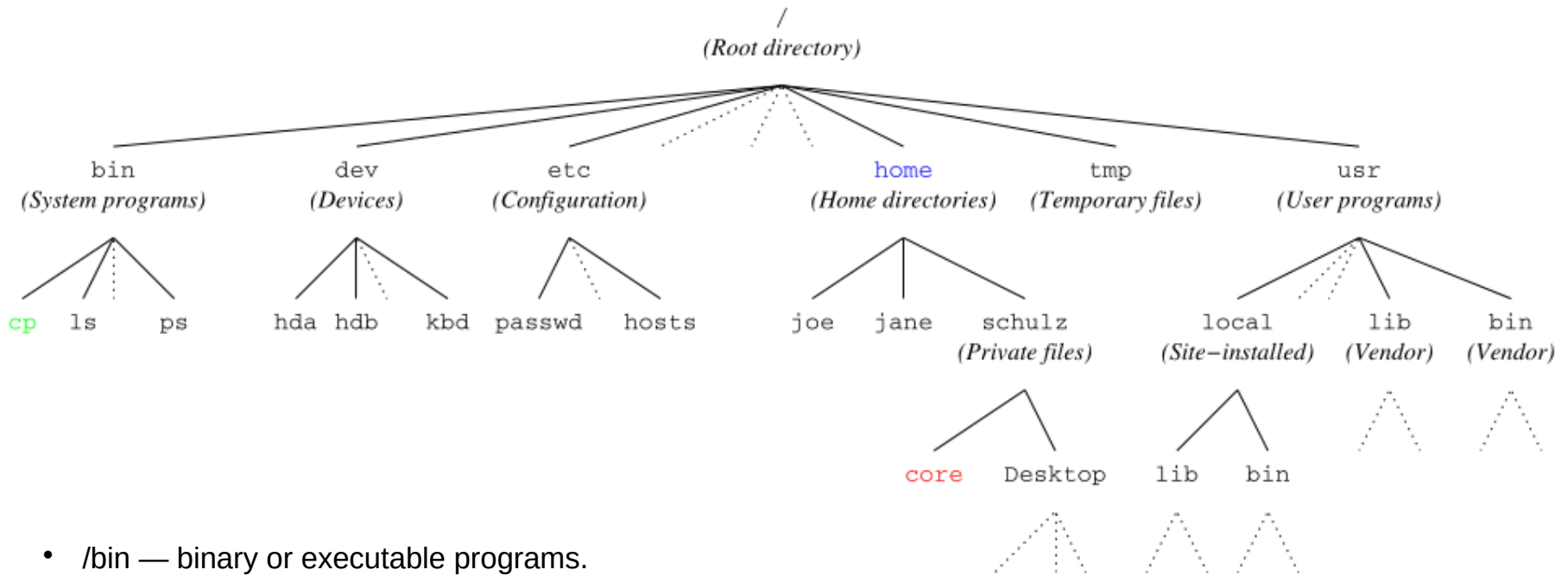
Unix basic concepts

- **UNIX is a multi-user system.** Each user has:
 - User name (mine is schulz on most machines)
 - Numerical user id (e.g. 500)
 - Home directory: A place where (most of) his or her files are stored
- **UNIX is a multi-tasking system**, i.e. it can run multiple programs at once. A running program (with its data) is called a process. Each process has:
 - Owner (a user)
 - Working directory (a place in the file system)
 - Various resources
- **A shell is a command interpreter**, i.e. a process accepting and executing commands from a user.
 - A shell is typically owned by the user using it
 - The initial working directory of a shell is typically the users home directory (but can be changed by commands)

Unix basic concepts

- There are two kinds of users:
 - Super users (“root”)
 - Normal users
- **Super-users:**
 - Have unlimited access to all files and resources
 - Always have numerical user id 0
 - Normally have user name “root” (but there can be more than one user name associated with UID 0)
 - Can seriously damage the system!
- **Normal users**
 - Can only access files if they have the appropriate permissions
 - Can belong to one or more groups. Users within a group can share files
 - Usually cannot damage the system or other users files!

File System Tree



- /bin — binary or executable programs.
- /etc — system configuration files.
- /home — home directory. It is the default current directory.
- /opt — optional or third-party software.
- /tmp — temporary space, typically cleared on reboot.
- /usr — User related programs.
- /var — log files.

Unix basic commands

- **pwd**

shows the directory where you are working

- **ls**

lists the content of a directory

Example: `ls -ltr` (the most recent ones at the end), `ls -a` (for 'invisible' files)

- **cd**

used to move among directories

Example: `cd ..` (one directory up); `cd` (to go to the home directory)

`cd /Users/chris/home` (moves following an absolute path)

- **mkdir**

creates a directory

Example: `mkdir data/`

`mkdir ~/data` (creates a directory `/data`)

Unix basic commands

- **cp**

makes a copy of a file (in another file, in another directory) and of a directory

Example: `cp readme.txt save.dat`

`cp readme.txt data/`

`cp -r data/ new_data` (copies an entire dir)

- **mv**

`mv` files/directories, rename files/dirs (does not make a copy)

Example: `mv readme.txt data.txt`

- **rm**

removes files and dirs

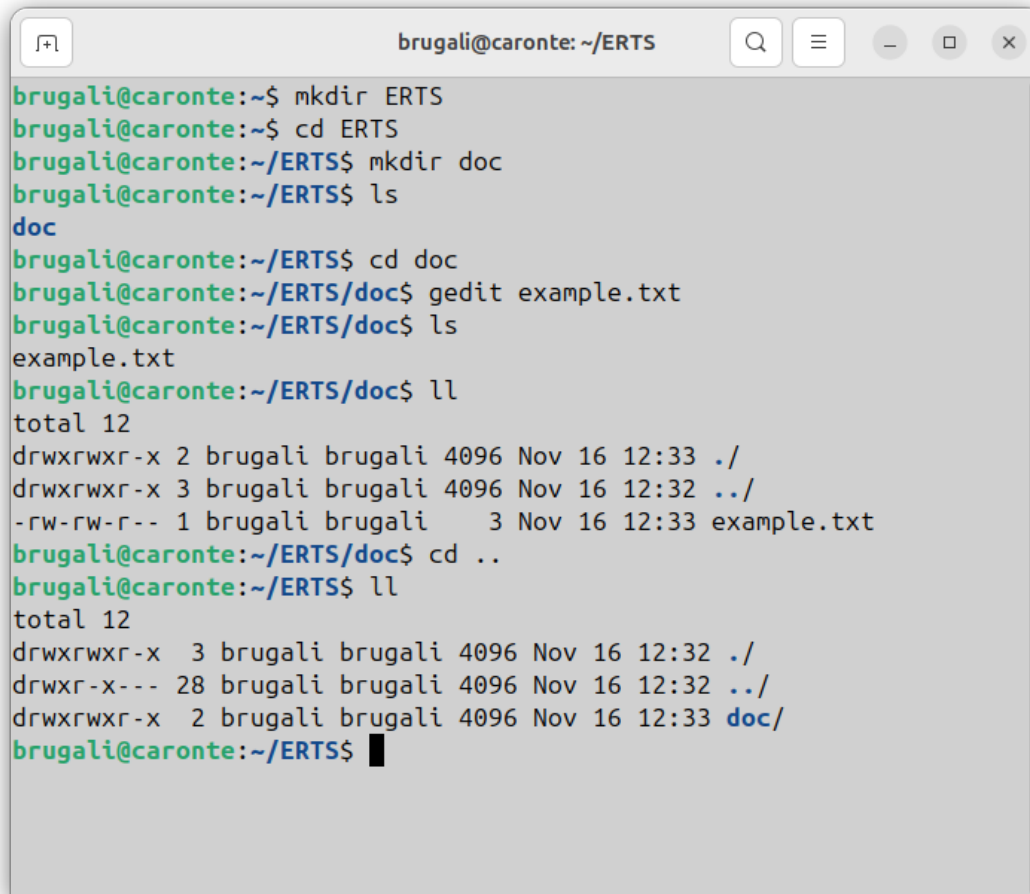
Example: `rm readme.txt`

`rm -r data/` (removes the entire directory)

`rm -i readme.txt` (asks confirmation before deleting)

Unix basic commands

- **gedit** example.txt
 - Creates the example text file and open it in a text editor

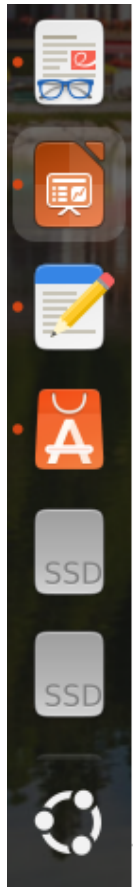


```
brugali@caronte: ~/ERTS
brugali@caronte:~$ mkdir ERTS
brugali@caronte:~$ cd ERTS
brugali@caronte:~/ERTS$ mkdir doc
brugali@caronte:~/ERTS$ ls
doc
brugali@caronte:~/ERTS$ cd doc
brugali@caronte:~/ERTS/doc$ gedit example.txt
brugali@caronte:~/ERTS/doc$ ls
example.txt
brugali@caronte:~/ERTS/doc$ ll
total 12
drwxrwxr-x 2 brugali brugali 4096 Nov 16 12:33 ./
drwxrwxr-x 3 brugali brugali 4096 Nov 16 12:32 ../
-rw-rw-r-- 1 brugali brugali   3 Nov 16 12:33 example.txt
brugali@caronte:~/ERTS/doc$ cd ..
brugali@caronte:~/ERTS$ ll
total 12
drwxrwxr-x  3 brugali brugali 4096 Nov 16 12:32 ./
drwxr-x--- 28 brugali brugali 4096 Nov 16 12:32 ../
drwxrwxr-x  2 brugali brugali 4096 Nov 16 12:33 doc/
brugali@caronte:~/ERTS$
```



```
example.txt
~/ERTS
Save
1
2 Testo di prova
Plain Text Tab Width: 8 Ln 2, Col 15 INS
```

How to install new programs



App Center

- 🔍 Explore
- ★ Featured
- 🕒 Productivity
- 🔧 Development
- 🎮 Games



🔍 Integrate



Notepad++ (WINE)

Taqi Raza
Development



Open

Channel

latest/stable 8.7.1



APT : Advanced Package Tool

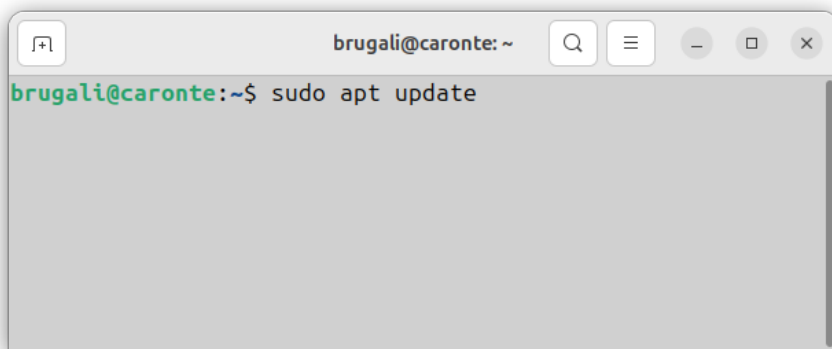
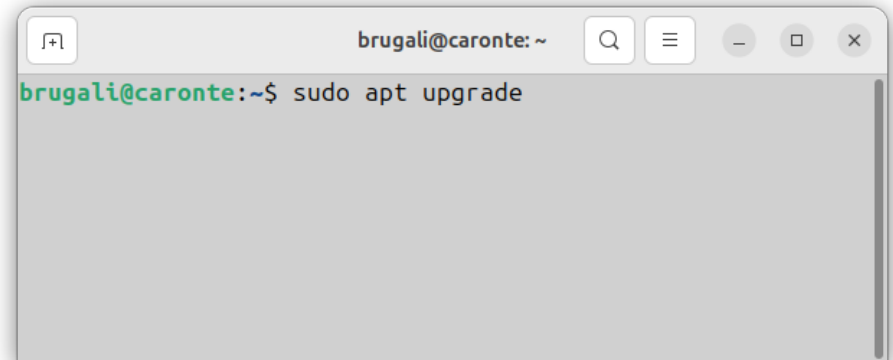
Advanced package tool, or APT, is a free-software user interface that works with core libraries to handle the installation and removal of software on Debian and Debian-based Linux distributions.

APT simplifies the process of managing software on Unix-like computer systems by automating the retrieval, configuration and installation of software packages, either from precompiled files or by compiling source code.

Update package database with apt

apt actually works on a database of available packages. If the database is not updated, the system won't know if there are any newer packages available. This is why updating the repository should be the first thing to do in any Linux system after a fresh install.

Updating the package database requires superuser privileges so you'll need to use sudo.

A terminal window titled 'brugali@caronte: ~' with search, menu, and window control icons. The prompt 'brugali@caronte:~\$' is followed by the command 'sudo apt update'.A terminal window titled 'brugali@caronte: ~' with search, menu, and window control icons. The prompt 'brugali@caronte:~\$' is followed by the command 'sudo apt upgrade'.

APT : Advanced Package Tool

How to install new packages with apt

```
sudo apt install <package_name>
```

Just replace the <package_name> with the desired package. Suppose you want to install mplayer, you can simply use the command below:

```
sudo apt install mplayer
```

How to remove installed packages with apt

```
sudo apt remove <package_name>
```

```
sudo apt purge <package_name>
```

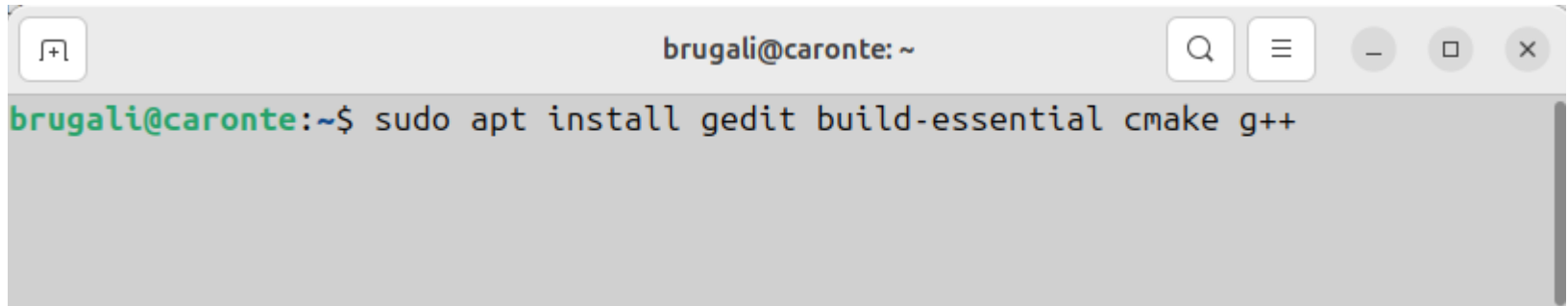
apt remove just removes the binaries of a package. It leaves residue configuration files.

apt purge removes everything related to a package including the configuration files.

APT : Advanced Package Tool

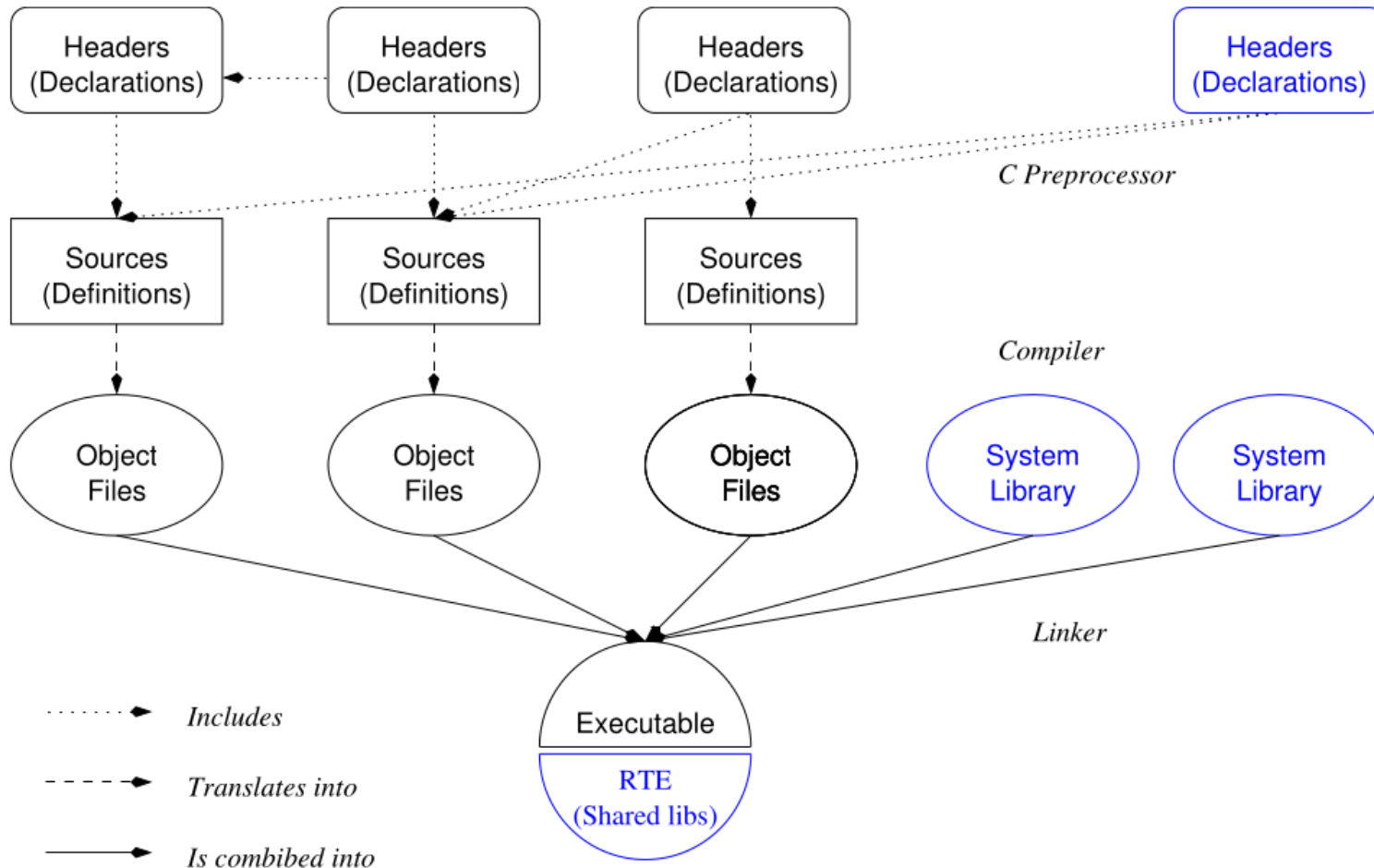
Attenzione!

Pacchetti da installare per le esercitazioni

A terminal window with a title bar that reads 'brugali@caronte: ~'. The title bar includes a search icon, a menu icon, and standard window control buttons (minimize, maximize, close). The terminal content shows the prompt 'brugali@caronte:~\$' followed by the command 'sudo apt install gedit build-essential cmake g++'.

```
brugali@caronte:~$ sudo apt install gedit build-essential cmake g++
```

Structure of a C program



ese01_simple.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char* argv[]) {
5
6      printf("\n\n");
7
8      for(int i=0; i < argc; i++) {
9
10         printf( "argument[%d] : %s\n", i , argv[i] );
11
12     }
13
14     printf("\n\n");
15
16     exit(0);
17 }
```

ese01_simple.c

```
/Lecture_01$ gcc ese01_simple.c -o ese01_simple  
/Lecture_01$
```

Compilazione

```
/Lecture_01$ ./ese01_simple  
  
argument[0] : ./ese01_simple
```

Esecuzione

```
/Lecture_01$ ./ese01_simple a b c  
  
argument[0] : ./ese01_simple  
argument[1] : a  
argument[2] : b  
argument[3] : c
```

Esecuzione

ese02_function.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void printArguments(int size, char* arguments[]);
5
6  int main(int argc, char* argv[]) {
7
8      printArguments(argc, argv);
9
10     exit(0);
11 }
12
13 void printArguments(int size, char* arguments[]) {
14     printf("\n");
15
16     for(int i=0; i < size; i++) {
17
18         printf( "argument[%d] : %s\n", i , arguments[i] );
19
20     }
21
22     printf("\n\n");
23 }
```

ese03_library.h .c

```
1  /*
2  * ese03_library.h
3  */
4
5  void printArguments(int size, char* arguments[]);
```

```
1  /*
2  * ese03_library.c
3  */
4  #include "ese03_library.h"
5  #include <stdio.h>
6
7  void printArguments(int size, char* arguments[]) {
8      printf("\n");
9
10     for(int i=0; i < size; i++) {
11         printf( "argument[%d] : %s\n", i , arguments[i] );
12     }
13
14     printf("\n\n");
15 }
16
17 }
```


ese03_main.c

```
1  /*
2   * Main program
3   */
4  #include "ese03_library.h"
5  #include <stdlib.h>
6
7  int main(int argc, char* argv[]) {
8
9      printArguments(argc, argv);
10
11      exit(0);
12  }
```

```
/Lecture_01$
/Lecture_01$ gcc ese03_main.c ese03_library.c -o ese03_func
/Lecture_01$
```

Compilazione

```
/Lecture_01$ ./ese03_func a b c
```

```
argument[0] : ./ese03_func
argument[1] : a
argument[2] : b
argument[3] : c
```

Esecuzione

Executable files

```
drwxrwxr-x 2 brugali brugali 4096 Nov 16 22:10 ./
drwxrwxr-x 6 brugali brugali 4096 Nov 16 18:15 ../
-rwxrwxr-x 1 brugali brugali 16088 Nov 16 19:04 ese01_simple*
-rw-rw-r-- 1 brugali brugali 268 Nov 16 19:04 ese01_simple.c
-rwxrwxr-x 1 brugali brugali 16128 Nov 16 19:44 ese02_function*
-rw-rw-r-- 1 brugali brugali 371 Nov 16 19:48 ese02_function.c
-rwxrwxr-x 1 brugali brugali 16168 Nov 16 22:05 ese03_func*
-rw-rw-r-- 1 brugali brugali 303 Nov 16 22:05 ese03_library.c
-rw-rw-r-- 1 brugali brugali 76 Nov 16 19:54 ese03_library.h
-rw-rw-r-- 1 brugali brugali 163 Nov 16 21:40 ese03_main.c
```

ese03_library.o

```
/Lecture_01$  
/Lecture_01$ gcc -c ese03_library.c -o ese03_library.o  
/Lecture_01$
```

-c means to create an intermediary object file, rather than an executable.

Compila
la libreria

```
/Lecture_01$  
/Lecture_01$ ar rcs ese03_library.a ese03_library.o  
/Lecture_01$
```

This creates the static library.

r means to insert with replacement, c means to create a new archive, and
s means to write an index.

Genera una
libreria statica

```
/Lecture_01$  
/Lecture_01$ gcc -I. ese03_main.c ese03_library.a -o ese03_main  
/Lecture_01$
```

Link del main
con la libreria

ese03_library.o

```
drwxrwxr-x 2 brugali brugali 4096 Nov 16 22:26 ./
drwxrwxr-x 6 brugali brugali 4096 Nov 16 18:15 ../
-rwxrwxr-x 1 brugali brugali 16088 Nov 16 19:04 ese01_simple*
-rw-rw-r-- 1 brugali brugali 268 Nov 16 19:04 ese01_simple.c
-rwxrwxr-x 1 brugali brugali 16128 Nov 16 19:44 ese02_function*
-rw-rw-r-- 1 brugali brugali 371 Nov 16 19:48 ese02_function.c
-rwxrwxr-x 1 brugali brugali 16168 Nov 16 22:05 ese03_func*
-rw-rw-r-- 1 brugali brugali 1928 Nov 16 22:23 ese03_library.a
-rw-rw-r-- 1 brugali brugali 303 Nov 16 22:05 ese03_library.c
-rw-rw-r-- 1 brugali brugali 76 Nov 16 19:54 ese03_library.h
-rw-rw-r-- 1 brugali brugali 1776 Nov 16 22:23 ese03_library.o
-rwxrwxr-x 1 brugali brugali 16168 Nov 16 22:26 ese03_main*
-rw-rw-r-- 1 brugali brugali 163 Nov 16 21:40 ese03_main.c
```

Make file

```
1 TARGET = ese03
2
3 $(TARGET): ese03_main.o ese03_library.a
4     gcc $^ -o $@
5
6 ese03_main.o: ese03_main.c
7     gcc -c $< -o $@
8
9 ese03_library.a: ese03_library.o
10    ar rcs $@ $^
11
12 ese03_library.o: ese03_library.c ese03_library.h
13    gcc -c -o $@ $<
14
15 clean:
16    rm -f *.o *.a $(TARGET)
```

`$@` the name of the target being generated

`$<` the first prerequisite (usually a source file).

`$^` all the prerequisites

```
~/Threads/Lecture_01$ make
```