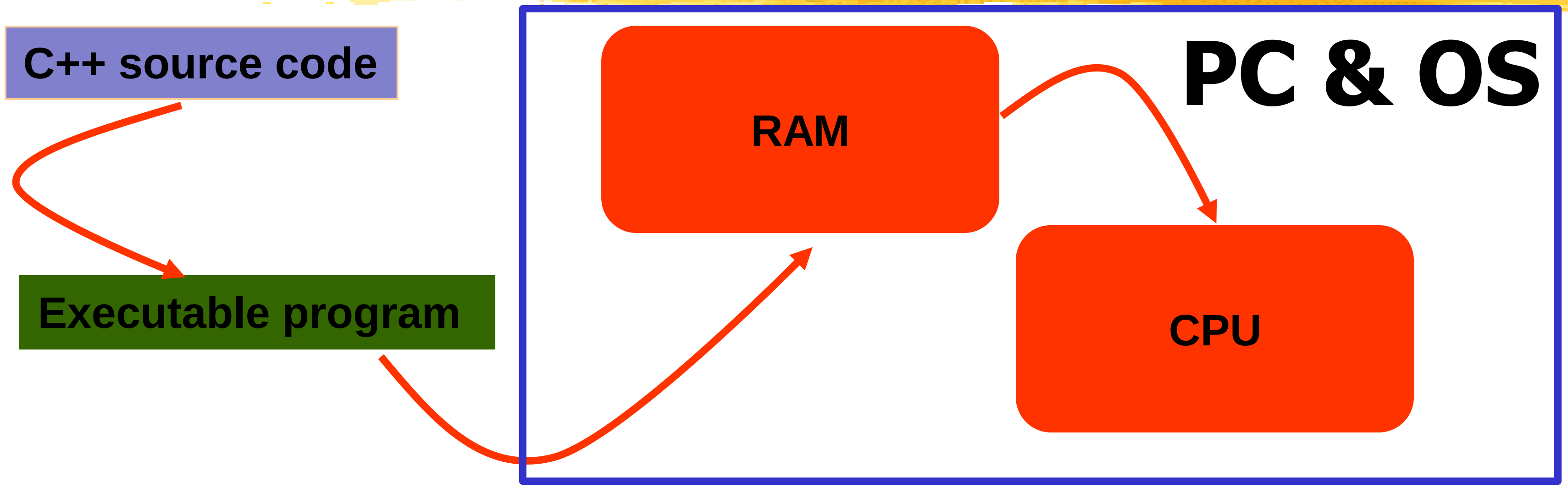
Foundations and advanced C++ programming language

F.S. Cafagna

INFN
Istituto Nazionale di Fisica Nucleare
Sezione di Bari

Intro



- We would like to write, using a given programming grammar, *i.e.* language, a code, in an human readable …
- … and transform it into something that can be executed by a given processor.
- The latest being named an executable program that can be load by the operating system (**OS**) into a device, *i.e.* **PC** (Personal Computer), memory, *i.e.* **RAM**, and executed by the processor, *i.e.* **CPU**.

Personal Computer

- The Random access memory (RAM) is a group of integrating circuits (board) that implements the storage of data in a random order.
- RAM is volatile. Its content is erased upon PC power down.
- What "random" means? Every data is extracted in a fixed time, independent of memory address or of any relationship with the previously written or read data.

Personal Computer

- Data (datum) ?
  - A computer stores any information binary coding it. Any information: text, digit, immagine, audio, etc. etc. it is converted into an ordered (often coded) bit stream or block. The smaller bit block a PC can handle is a byte.
- Bit ?
  - A digit in binary format. It can only be assigned: 0,1
- Byte ?
  - A block of 8 bit. It can span the range from 0 to (2^8-1) (255)
  - 255_{10}=FF_{16}=11111111_2

Personal Computer

- The Random access memory (RAM) is a group of integrating circuits (board) that implements the storage of data in a random order.
- RAM is volatile. Its content is erased upon PC power down.
- What "random" means? Every data is extracted in a fixed time, independent of memory address or of any relationship with the previously written or read data.
- Access to RAM is usually slow, compared to a CPU speed. For this reason usually a CPU has its own on board “little” RAM: a cache, to speed-up data accesses during calculations.

Personal Computer

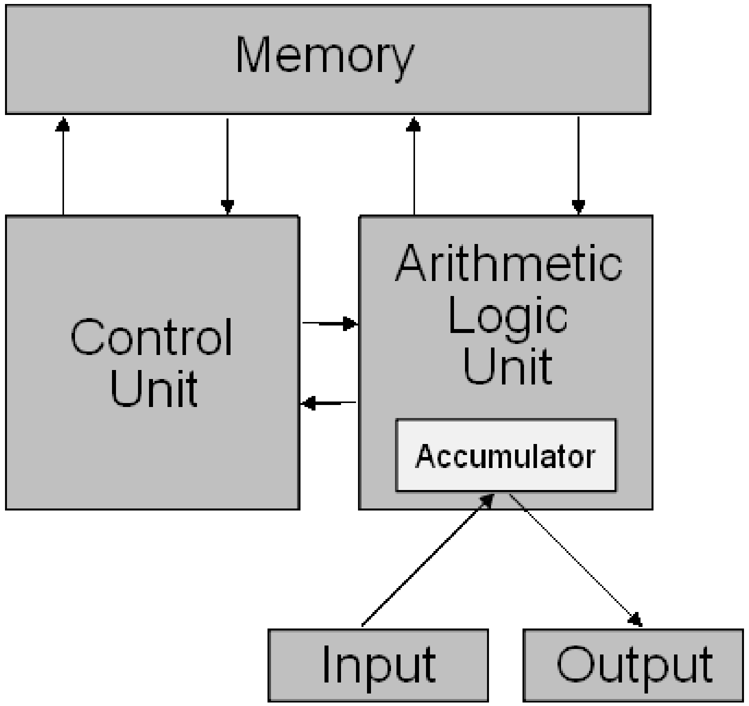
- CPU?
  - Central Processing Unit (CPU).
  - CPU operates on data.
  - It is a logic machine that can execute a finite set of instructions (instruction set architecture - ISA).
  - An ISA is strongly related to programming, and includes the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O.

Personal Computer

- CPU?
  - An ISA is not the microarchitecture of the CPU. Different microarchitecture can share (mostly) the same ISA. Intel and AMD implement (almost) the same x86 ISA but have completely different architectures.
  - An ISA define the machine language instruction set, including the opcodes (operation code) and operands.
    - Complex Instruction Set Computers (CISC).
    - Reduced Instruction Set Computers (RISC)
    - Minimal Instruction Set Computers (MISC)

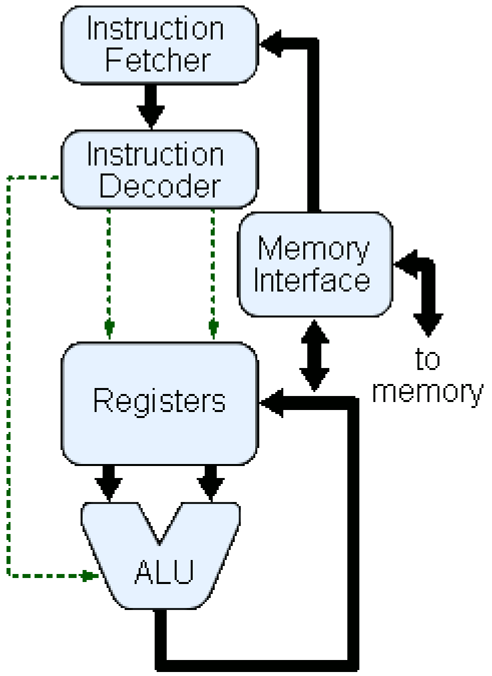
Personal Computer

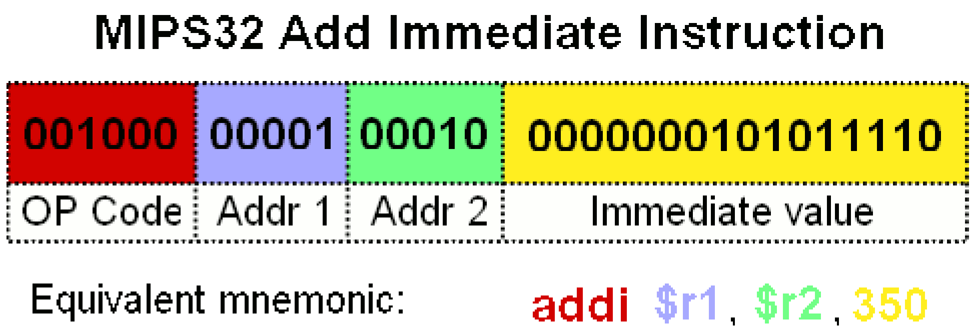
- **fetch**. Instruction is fetch from memory.
- **decode**. Instruction is decoded and operand specified.
- **execute**. Computation is executed
- **writeback**. Computation results are wrote back into output registers.



Personal Computer

- The possibility of fetch and store instructions and data, increase versatility of a computing unit and lead to a natural need for programming.
- A list of actions, the program code, is translated and coded into a list of low level instructions according to the CPU microcode and registered in memory.





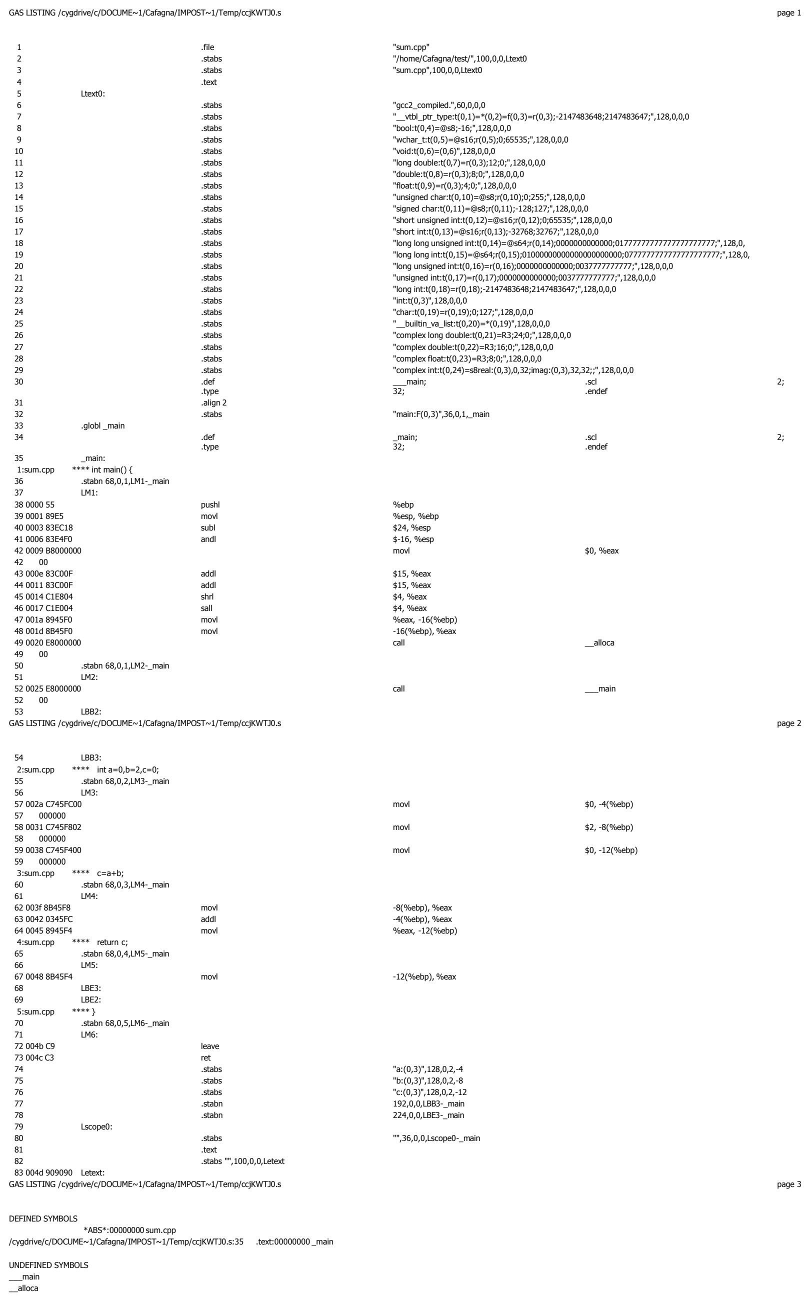
MIPS32 Add Immediate Instruction

001000	00001	00010	0000000101011110
OP Code	Addr 1	Addr 2	Immediate value

Equivalent mnemonic: addi $r1, $r2, 350

Personal Comput

```
int main() {
  int a=0,b=2,c=0;
  c= a+b;
  return c;
}
```



```
GAS LISTING /cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s                                        page 1


  1                          .file                "sum.cpp"
  2                          .stabs               "/home/Cafagna/test/",100,0,0,Ltext0
  3                          .stabs               "sum.cpp",100,0,0,Ltext0
  4                          .text
  5            Ltext0:
  6                          .stabs               "gcc2_compiled.",60,0,0,0
  7                          .stabs               "__vtbl_ptr_type:t(0,1)=*(0,2)=f(0,3)=r(0,3);-2147483648;2147483647;",128,0,0,0
  8                          .stabs               "bool:t(0,4)=@s8;-16;",128,0,0,0
  9                          .stabs               "wchar_t:t(0,5)=@s16;r(0,5);0;65535;",128,0,0,0
 10                          .stabs               "void:t(0,6)=(0,6)",128,0,0,0
 11                          .stabs               "long double:t(0,7)=r(0,3);12;0;",128,0,0,0
 12                          .stabs               "double:t(0,8)=r(0,3);8;0;",128,0,0,0
 13                          .stabs               "float:t(0,9)=r(0,3);4;0;",128,0,0,0
 14                          .stabs               "unsigned char:t(0,10)=@s8;r(0,10);0;255;",128,0,0,0
 15                          .stabs               "signed char:t(0,11)=@s8;r(0,11);-128;127;",128,0,0,0
 16                          .stabs               "short unsigned int:t(0,12)=@s16;r(0,12);0;65535;",128,0,0,0
 17                          .stabs               "short int:t(0,13)=@s16;r(0,13);-32768;32767;",128,0,0,0
 18                          .stabs               "long long unsigned int:t(0,14)=@s64;r(0,14);0000000000000;01777777777777777777777;",128,0,
 19                          .stabs               "long long int:t(0,15)=@s64;r(0,15);01000000000000000000000;0777777777777777777777;",128,0,
 20                          .stabs               "long unsigned int:t(0,16)=r(0,16);0000000000000;0037777777777;",128,0,0,0
 21                          .stabs               "unsigned int:t(0,17)=r(0,17);0000000000000;0037777777777;",128,0,0,0
 22                          .stabs               "long int:t(0,18)=r(0,18);-2147483648;2147483647;",128,0,0,0
 23                          .stabs               "int:t(0,3)",128,0,0,0
 24                          .stabs               "char:t(0,19)=r(0,19);0;127;",128,0,0,0
 25                          .stabs               "__builtin_va_list:t(0,20)=*(0,19)",128,0,0,0
 26                          .stabs               "complex long double:t(0,21)=R3;24;0;",128,0,0,0
 27                          .stabs               "complex double:t(0,22)=R3;16;0;",128,0,0,0
 28                          .stabs               "complex float:t(0,23)=R3;8;0;",128,0,0,0
 29                          .stabs               "complex int:t(0,24)=s8real:(0,3),0,32;imag:(0,3),32,32;;",128,0,0,0
 30                          .def                 ___main;             .scl              2;
                             .type                32;                  .endef
 31                          .align 2
 32                          .stabs               "main:F(0,3)",36,0,1,_main
 33            .globl _main
 34                          .def                 _main;               .scl              2;
                             .type                32;                  .endef
 35            _main:
  1:sum.cpp   **** int main() {
 36            .stabn 68,0,1,LM1-_main
 37            LM1:
 38 0000 55                  pushl                %ebp
 39 0001 89E5                movl                 %esp, %ebp
 40 0003 83EC18              subl                 $24, %esp
 41 0006 83E4F0              andl                 $-16, %esp
 42 0009 B8000000                                 movl                 $0, %eax
 42      00
 43 000e 83C00F              addl                 $15, %eax
 44 0011 83C00F              addl                 $15, %eax
 45 0014 C1E804              shrl                 $4, %eax
 46 0017 C1E004              sall                 $4, %eax
 47 001a 8945F0              movl                 %eax, -16(%ebp)
 48 001d 8B45F0              movl                 -16(%ebp), %eax
 49 0020 E8000000                                 call                 __alloca
 49      00
 50            .stabn 68,0,1,LM2-_main
 51            LM2:
 52 0025 E8000000                                 call                 ___main
 52      00
 53            LBB2:
GAS LISTING /cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s                                        page 2


 54            LBB3:
  2:sum.cpp   ****   int a=0,b=2,c=0;
 55            .stabn 68,0,2,LM3-_main
 56            LM3:
 57 002a C745FC00                                 movl                 $0, -4(%ebp)
 57      000000
 58 0031 C745F802                                 movl                 $2, -8(%ebp)
 58      000000
 59 0038 C745F400                                 movl                 $0, -12(%ebp)
 59      000000
  3:sum.cpp   ****   c=a+b;
 60            .stabn 68,0,3,LM4-_main
 61            LM4:
 62 003f 8B45F8              movl                 -8(%ebp), %eax
 63 0042 0345FC              addl                 -4(%ebp), %eax
 64 0045 8945F4              movl                 %eax, -12(%ebp)
  4:sum.cpp   ****   return c;
 65            .stabn 68,0,4,LM5-_main
 66            LM5:
 67 0048 8B45F4              movl                 -12(%ebp), %eax
 68            LBE3:
 69            LBE2:
  5:sum.cpp   **** }
 70            .stabn 68,0,5,LM6-_main
 71            LM6:
 72 004b C9                  leave
 73 004c C3                  ret
 74                          .stabs               "a:(0,3)",128,0,2,-4
 75                          .stabs               "b:(0,3)",128,0,2,-8
 76                          .stabs               "c:(0,3)",128,0,2,-12
 77                          .stabn               192,0,0,LBB3-_main
 78                          .stabn               224,0,0,LBE3-_main
 79            Lscope0:
 80                          .stabs               "",36,0,0,Lscope0-_main
 81                          .text
 82                          .stabs "",100,0,0,Letext
 83 004d 909090  Letext:
GAS LISTING /cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s                                        page 3


DEFINED SYMBOLS
                 *ABS*:00000000 sum.cpp
/cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s:35     .text:00000000 _main

UNDEFINED SYMBOLS
___main
__alloca
```

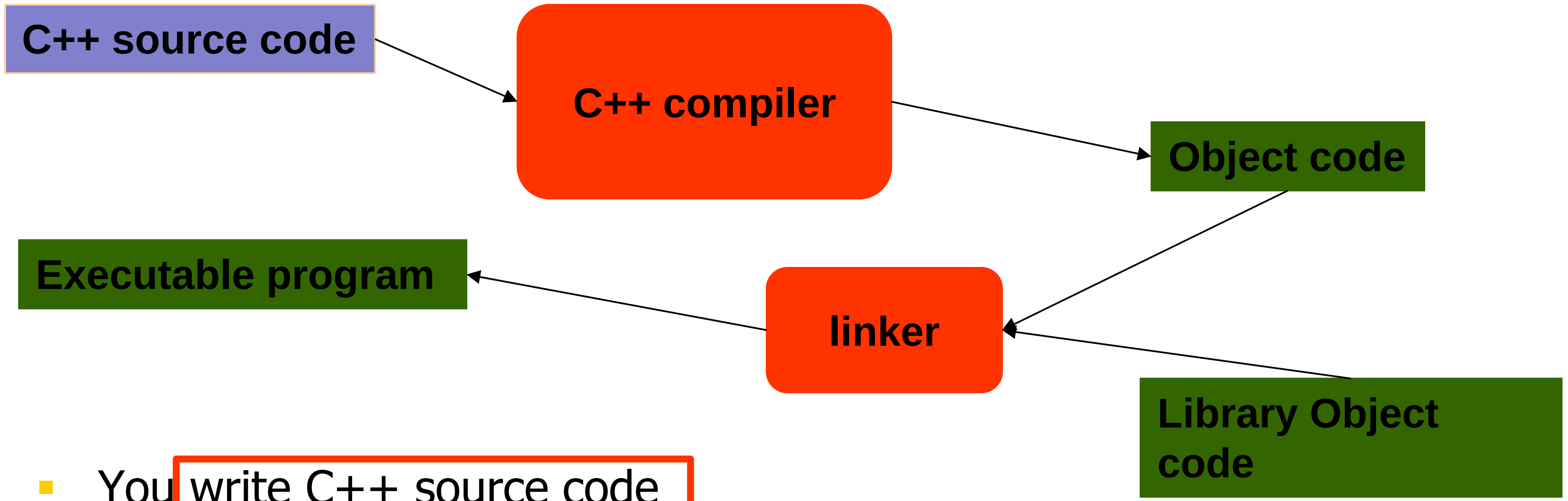
Personal Computer

```
1:sum.cpp        **** int main() {
 36                           .stabn 68,0,1,LM1-_main
 37                           LM1:
 38 0000 55                               pushl        %ebp
 39 0001 89E5                             movl         %esp, %ebp
 40 0003 83EC18                           subl         $24, %esp
 41 0006 83E4F0                           andl         $-16, %esp
 42 0009 B8000000                         movl         $0, %eax
 42      00
 43 000e 83C00F                           addl         $15, %eax
 44 0011 83C00F                           addl         $15, %eax
 45 0014 C1E804                           shrl         $4, %eax
 46 0017 C1E004                           sall         $4, %eax
 47 001a 8945F0                           movl         %eax, -16(%ebp)
 48 001d 8B45F0                           movl         -16(%ebp), %eax
 49 0020 E8000000                         call         __alloca
 49      00
 50                           .stabn 68,0,1,LM2-_main
 51                           LM2:
 52 0025 E8000000                         call         ___main
 52      00
 53                           LBB2:
 54                           LBB3:
 2:sum.cpp        ****   int a=0,b=2,c=0;
 55                           .stabn 68,0,2,LM3-_main
 56                           LM3:
 57 002a C745FC00                         movl         $0, -4(%ebp)
 57      000000
 58 0031 C745F802                         movl         $2, -8(%ebp)
 58      000000
 59 0038 C745F400                         movl         $0, -12(%ebp)
 59      000000
```

Personal Computer

```
3:sum.cpp      ****  c=a+b;
 60                    .stabn 68,0,3,LM4-_main
 61                    LM4:
 62 003f 8B45F8                   movl        -8(%ebp), %eax
 63 0042 0345FC                   addl        -4(%ebp), %eax
 64 0045 8945F4                   movl        %eax, -12(%ebp)
  4:sum.cpp      ****  return c;
 65                    .stabn 68,0,4,LM5-_main
 66                    LM5:
 67 0048 8B45F4                   movl        -12(%ebp), %eax
 68                    LBE3:
 69                    LBE2:
  5:sum.cpp      **** }
 70                    .stabn 68,0,5,LM6-_main
 71                    LM6:
 72 004b C9                       leave
 73 004c C3                       ret
 74                               .stabs      "a:(0,3)",128,0,2,-4
 75                               .stabs      "b:(0,3)",128,0,2,-8
 76                               .stabs      "c:(0,3)",128,0,2,-12
 77                               .stabn      192,0,0,LBB3-_main
 78                               .stabn      224,0,0,LBE3-_main
 79                    Lscope0:
 80                               .stabs      "",36,0,0,Lscope0-_main
 81                               .text
 82                               .stabs "",100,0,0,Letext
 83 004d 909090        Letext:
```

Compilation and linking



- You write C++ source code
  - Source code is (in principle) human readable
- The compiler translates what you wrote into object code (sometimes called machine code)
  - Object code is simple enough for a computer to "understand"
- The linker links your code to system code needed to execute
  - E.g. input/output libraries, operating system code, and windowing code
- The result is an executable program
  - E.g. a **.exe** file on windows or an **a.out** file on Unix

Introduction

- We already reviewed a bit the basics of a personal computer…
  - We need to know about the hardware if we want to write a good software!
- … and programming, …
  - What does programming means?
  - Do we know how many programmable device we have around us?
- … along with the minimal set of programming tools needed to start up.
- Now we will try to focus on the different programming approaches: from procedures to objects.

So what is programming?

- Conventional definitions
  - Telling a **very** fast moron ***exactly*** what to do
  - A plan for solving a problem on a computer
  - Specifying the order of a program execution
    - But modern programs often involve millions of lines of code
    - And manipulation of data is central
- Definition from another domain (academia)
  - A … program is an organized and directed accumulation of resources to accomplish specific … objectives …
    - Good, but no mention of actually doing anything
- The definition we'll use
  - Specifying the structure and behavior of a program, and testing that the program performs its task correctly and with acceptable performance
    - Never forget to check that "it" works
- Software == one or more programs

Stroustrup/Programming

Programming

- Programming, that is, the ideals, techniques, and tools of expressing ideas in code.
- Programming is fundamentally simple
  - Just state what the machine is to do
- So why is programming hard?
  - We want "the machine" to do complex things
    - And computers are nitpicking, unforgiving, dumb beasts
  - The world is more complex than we'd like to believe
    - So we don't always know the implications of what we want
  - "Programming is understanding"
    - When you can program a task, you understand it
    - When you program, you spend significant time trying to understand the task you want to automate
  - Programming is part practical, part theory
    - If you are just practical, you produce non-scalable unmaintainable hacks
    - If you are just theoretical, you produce toys

Stroustrup/Programming

Linux: a brief introduction

Operating System

- What is an operating system?

  Is a software layer whose job is to provide user programs with a better, simpler, cleaner model of the computer and to handle it managing all the computer resources.
- It is on top of the hardware layer.
- It is the most (and low level) piece of software and runs in kernel mode (supervisor mode), *i.e.* has complete access to all the hardware and can execute any instruction the machine is capable of executing.

Operating System

- The rest of the software runs in user mode, *i.e.* only a subset of the ISA is available the rest is forbidden.
- A modern O.S. manages and controls the resource load and supervise processes data and user's input/output (I/O) requests allocating tasks and internal resources as user services.

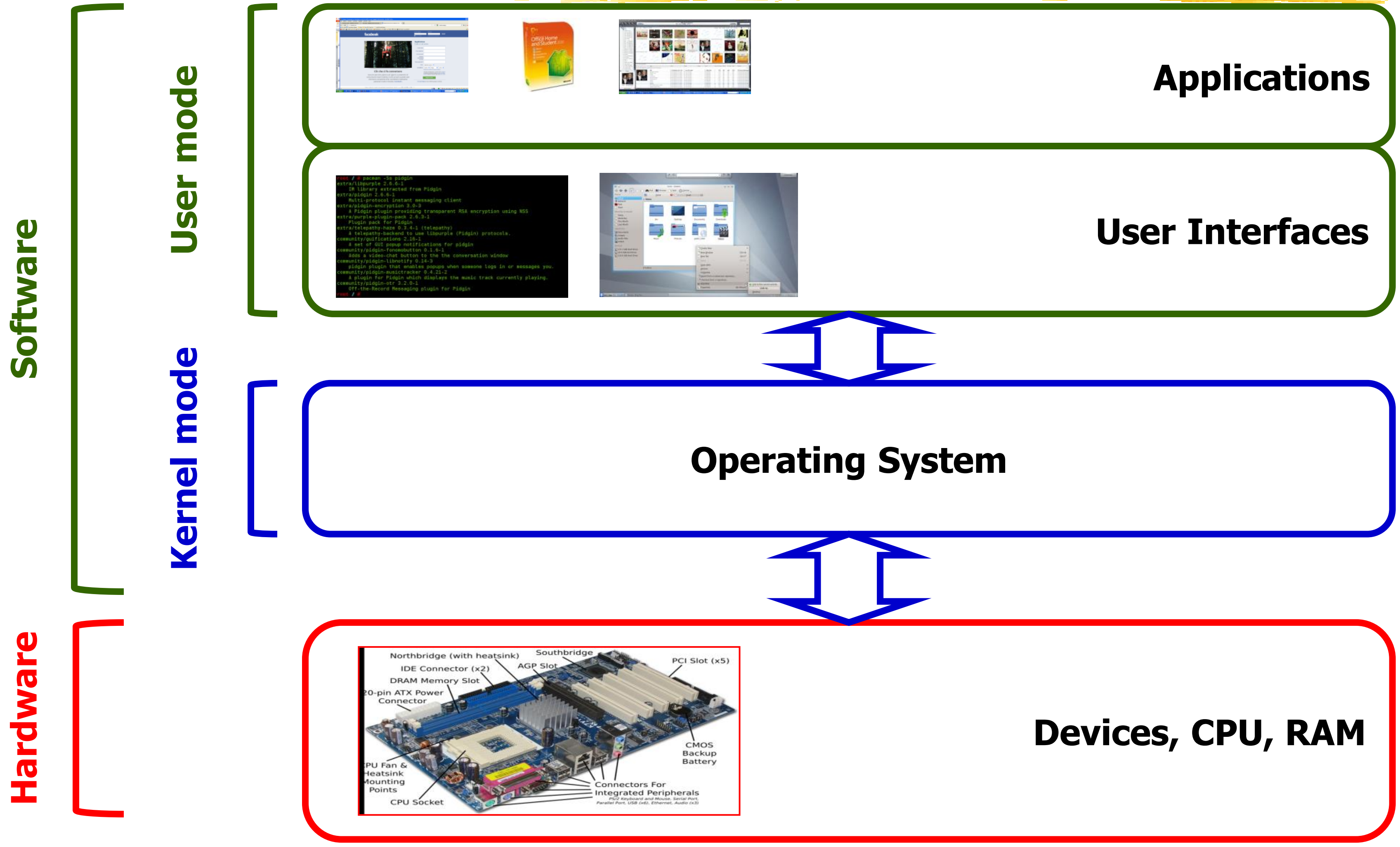
Operating System

- Supervise and distributes the memory load, chooses the system requests priorities, supervise I/O peripherals, network and filesystem.

Operating System

- To interact with the O.S. the user needs a program that is usually called shell.
- The shell is usually text based.
- There are also graphic user interfaces (GUI), that are no more built inside the kernel.
- More often a GUI is built on top of a windowing system (X11) handling the basic window management leaving to the user the choose of the look and feel of the GUI (GNOME, KDE, etc. etc.).

Operating system



```
root / # pacman -Ss pidgin
extra/libpurple 2.6.6-1
    IM library extracted from Pidgin
extra/pidgin 2.6.6-1
    Multi-protocol instant messaging client
extra/pidgin-encryption 3.0-3
    A Pidgin plugin providing transparent RSA encryption using NSS
extra/purple-plugin-pack 2.6.3-1
    Plugin pack for Pidgin
extra/telepathy-haze 0.3.4-1 (telepathy)
    A telepathy-backend to use libpurple (Pidgin) protocols.
community/guifications 2.16-1
    A set of GUI popup notifications for pidgin
community/pidgin-fonomobutton 0.1.6-1
    Adds a video-chat button to the the conversation window
community/pidgin-libnotify 0.14-3
    pidgin plugin that enables popups when someone logs in or messages you.
community/pidgin-musictracker 0.4.21-2
    A plugin for Pidgin which displays the music track currently playing.
community/pidgin-otr 3.2.0-1
    Off-the-Record Messaging plugin for Pidgin
root / #
```

How can I use it ?

- Follow distribution instructions
  - Just username and password

    …
- Different user interfaces. Essentially:
  - X-windows
  - ANSI terminal

How can I use it ?

- X-windows
  - XFree86. Open source implementation of the X11 protocol
- A “windows manager” is needed:
  - Twm
  - Fwm
  - Enlightenment
  - AfterStep
  - ………
- … too many windows … a “desktop manager” is needed:
  - KDE
  - GNOME
  - ….

How can I use it ?

- You badly need a “shell” !
  - It is a “command line interpreter” parsing the”user input” and execute the requested task.
  - “user input” ??? It can be a keyboard or a file: script
- It is our basic interface to the operating system.
- The most popular shells:
  - sh: Bourne shell
  - csh: C shell
  - ksh: Korn shell
  - tcsh: Enanched C shell
  - bash: Bourne Again shell

How can I use it ?

- **Where the damned files are?**
  - Into a hierarchical filesystem, organized in directories and files.
  - The hierarchy can be represented as a "tree" generated by a "root". The name used for the "root" is a single character: /
  - *Directories* are files containing other directories or files. They are files containing information on their contents: filenames.
  - *Filenames*. Are names of files contained into a directory. Names can contain all characters. The best will be to avoid the usage of "slash": /, and "blanck":

    /usr/local/bin/mio_eseguibile_.exe
    /home/corsovme/.filenascosto
    /home/corsovme/prova.exe.old_version
  - If a name starts with a dot the file is "invisible".

How can I use it ?

- **A tipical filesystem tree:**

```
/ -
    | - /bin                -- Most of the executables here
    | - /boot               -- What is needed to boot the OS
    | - /home               -- User's directories
  | - /usr                  -- All is needed by a standard user
       | - /usr/local          binaries, libraries, include files etc. etc. etc.
       | - /usr/bin
       | - /usr/lib
       | - /usr/include
  | - /include              -- System include files here
  | - /lib                  -- System libraries here, driver modules
                               included as well
  | - /etc                  -- OS configuration files here
  | -..........
```

How can I use it ?

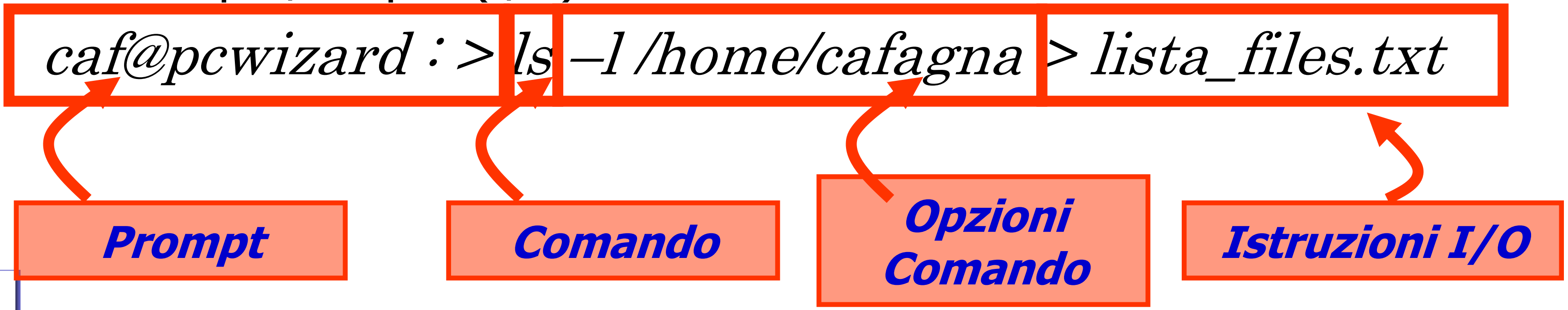
- **How can I navigate the filesystem ?**
  - cd : change directory
  - ls : list directory
- **How can I see what is stored into a file?**
  - cat : dumps file contents to the standard output
  - more, less: formatted prints file contents on standard ouput
  - edit it !!!

How can I use it ?

- Editors:
  - nano, pico : alphanumeric editors
  - emacs, xemacs: “more than editors: a way of life”
  - vi: no comment !
- Manual
  - Is online, distributed with the system software:
    - man : the unix manual, subdivided into 8 chapters
    - info : the GNU hypertext documentation system

Shell usage

- How can I use the shell ?
  - Once logged into the operating system, a shell process is created to manage our account and a *prompt* is issued, i.e. a string of character delimiting the start of the line we can use to enter our commands.
  - A command is a list of strings stating:
    - The command name.
    - The command options I would like to change.
    - Input/Output (I/O) instructions.



```
caf@pcwizard : > ls –l /home/cafagna > lista_files.txt
```

Shell usage

- How can I use the shell?
  - An application, if executable, can be run just calling it by name:

    /home/cafagna/prove_corso/my_first

    /usr/local/bin/mozilla

    /home/billy/test.exe

    ./prova_classi
  - *executable?*
    - *Unix attributes access mode to any file. A file can be:*
      - *read: r*
      - *write: w*
      - *executed: x*

Shell usage

- How can I use the shell?
  - Every one of this attribute can be set for three user categories:
    - *The rest of the world (others): o*
    - *The group the user account is registered to (group): g*
    - *The user account which own the file (user): u*
  - A file attribute is a 16 bit number. Less significant three bit are assigned to the *others*, the next three to the *group*, the next three to the *user*.

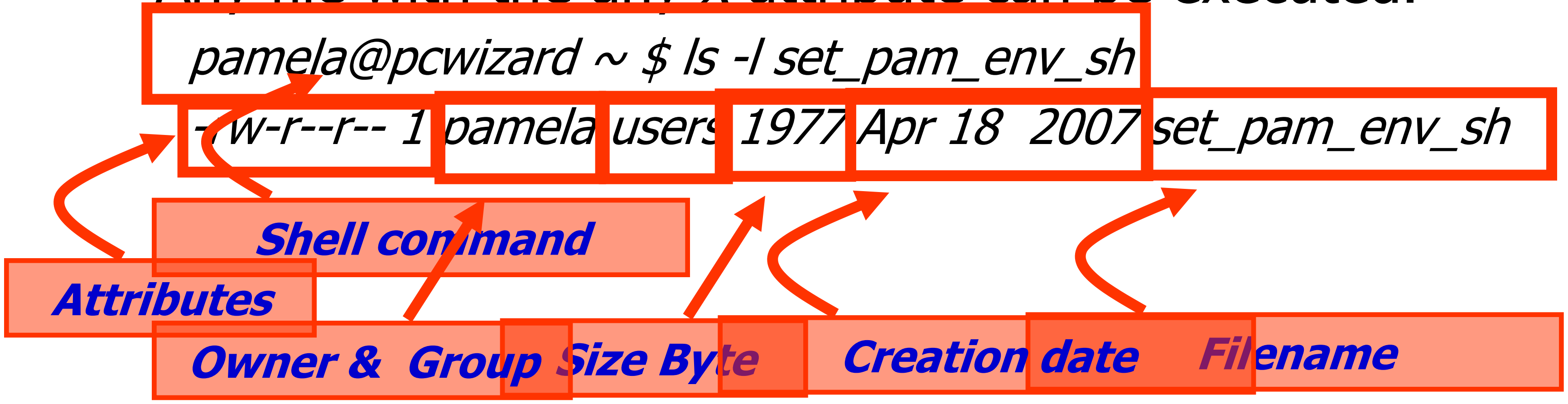
Shell usage

- How can I use the shell?
  - File attributes can be modified using the command: *chmod*

    chmod 755 nomefile

    chmod u+x,o=r,g=r nomefile

    chmod a=rwx nomefile
  - Any file with the any *x* attribute can be executed.



```
pamela@pcwizard ~ $ ls -l set_pam_env_sh
-rw-r--r-- 1 pamela users 1977 Apr 18  2007 set_pam_env_sh
```

Shell usage

- How can I use the shell ?
  - Don’t we state that a program can be executed just typing its name?
  - What about *ls* ? Isn’t stored in: /bin/ls ?
  - The shell rely on some environment variables to customize to my preferences. In particular on the variable named: PATH. The value of PATH is parsed and the executable searched for in any directory specified.

    .:/usr/bin:/bin:/usr/local/bin:/home/bin
  - Several directories can be set, separating their names using *:*

Shell usage

- How can I use the shell?
  - Ho can the enviroment variables be modified?
  - The command used depends on the shell we are using.
  - bash and all the Bourne shell based use the command: *export*

    export PATH=/usr:/usr/local/bin
  - csh and all the C-shell based use the command: *setenv*

    setenv PATH /usr:/usr/local/bin
  - To print the string stored in any variable you can use the command: *printenv*
  - Variable names must be prefixed by the charater *'$'* to be recognized as variable names:

    echo $PATH PATH ***String "PATH"*** ***Variable PATH***

Shell usage

- How can I use the shell?
  - How can the shell know my preferences?
  - It parses the user login directory, reads and executes commands listed in some files. If none of the following files is found, the default common files stored in the */etc* directory are executed. The parsed files are:
    - *.bash_profile,* executed anytime the user logins;
    - *.bash_login,* executed if no *.bash_profile* is found;
    - *.profile,* executed if no *.bash_login* is found;
    - *.bashrc,* executed every time a new *shell* is run*;*
    - *.bash_logout*, executed any time a *shell* is closed;

Shell usage

- How can I use the shell?
  - The above mentioned files contain a list of shell command…
  - … so it is possible to store these lists into file usually called *script*.
  - Usually *bash script*s are usually named with the *sh* extension.
  - These scripts can be executed by the shell usign the command: *source* (or the shortcut '.')
  - If executable the can be executed as the other file, i.e. typing their names.

Shell usage

- How can I use the shell?
  - Besides environment variables, shell can be customized defining *alias*, that is a shortcut for long or complex commands:

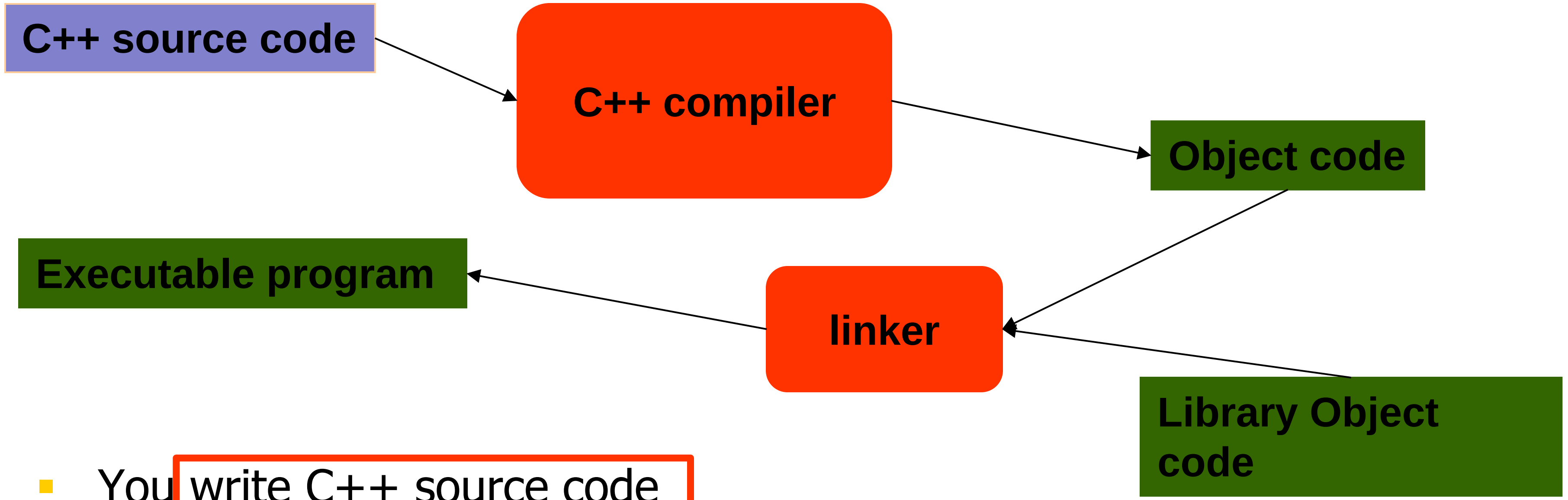
    alias dir='ls –l |grep ^d'
  - Usually *alias*es are defined in the *.bashrc.*
  - There are special shell variables storing all words typed in the command line:
    - 0, the first word;
    - n, the n-th+1 word.
  - Wildcards can be used (regular expression). The most used:
    - *, stands for "any string";
    - ?, stands for "any single character".

Shell usage

- How can I use the shell?
  - What do these simbols mean: '>', ">>", '<', '|' ?
  - First three are i/o modifier to redirect the *standard output* and the *standard input*.
  - The last is a symbol used to pipeline the output of a program into the input of another one.
  - What does redirection means?
    - What about if I want my program output not on the screen?
    - What about if I want to store a program list of options into a file and pass them to a program as comman line options?
      - cat < to_be_copied > copied
    - If I don't want to delete the redirected file I can concatenate the redirection stream using ">>"
      - cat < another_to_be_copied >> copied
  - If a program accepts input from the standard input then I can connect it to the standard output of another program.
    - That is I can "pipe" them
      - cat file_da_copiare | grep –c data_april

Brief introduction to programming tools (for Linux)

Compilation and linking



- You write C++ source code
  - Source code is (in principle) human readable
- The compiler translates what you wrote into object code (sometimes called machine code)
  - Object code is simple enough for a computer to "understand"
- The linker links your code to system code needed to execute
  - E.g. input/output libraries, operating system code, and windowing code
- The result is an executable program
  - E.g. a **.exe** file on windows or an **a.out** file on Unix

Editors

- Editors:
  - nano, pico : alphanumeric editors
  - emacs, xemacs: “more than editors: a way of life”
  - vi: no comment !
- Manual
  - Is online, distributed with the system software:
    - man : the unix manual, subdivided into 8 chapters
    - info : the GNU hypertext documentation system

Programming tools

- What are the compiling tools ?
  - gcc. The GNU suite: GNU Compiler Collection
    - `cpp`: prepocessor & compiler
    - `as` : assembler
    - `ld` : linker
    - …….

Programming tools

- Tools for debugging ?
  - *gdb*: The GNU debugger
  - *ddd, Kdbg*: Graphic front end for gdb
  - …
  - *gprof*: GNU profiler
  - *Valgrid, cachegrid, electricfence*: memory debuggers

Programming tools

- Let’s install also:
  - *gprof*
  - *Kdgb*
  - *Ddd (for ddd installation follow instructions on http://www.gnu.org/software/ddd/, download the rpm, install packages it depends from: gnuplot, Motif (openmotif or equivalent)*

Programming tools

- How can I use the tools ?
  - Just type their name and use the (right) options:
    - g++ [OPTIONS] filename_to_be_compiled
  - Usually there are two way to specify options:
    - Short form: -'a single char'
      - g++ –g –c my_first.cpp
    - Long form: --'string'
      - g++ --help

Programming tools

- How can I use the tools?
  - The options we will use the most:

    -v : be verbose

    -c : just compile, don't create executables

    -o filename : specify an output file name

    -*Ldirname* : specify the name of the directory to be searched for libraries (NOTE! No blanks between option and name)

    -llibname : nome of the library to be linked (NOTE! No blanks between option and name).

    -Idirname : nome del direttorio dove cercare gli "include" files (notare la mancanza di spazi)
  - NOTE! Only the name without prefix and extensions must be used with the *–l* option. So to use libMath.so or libMyAnalysis.so, you must use: *–lMath -lMyAnalysis*

Programming tools

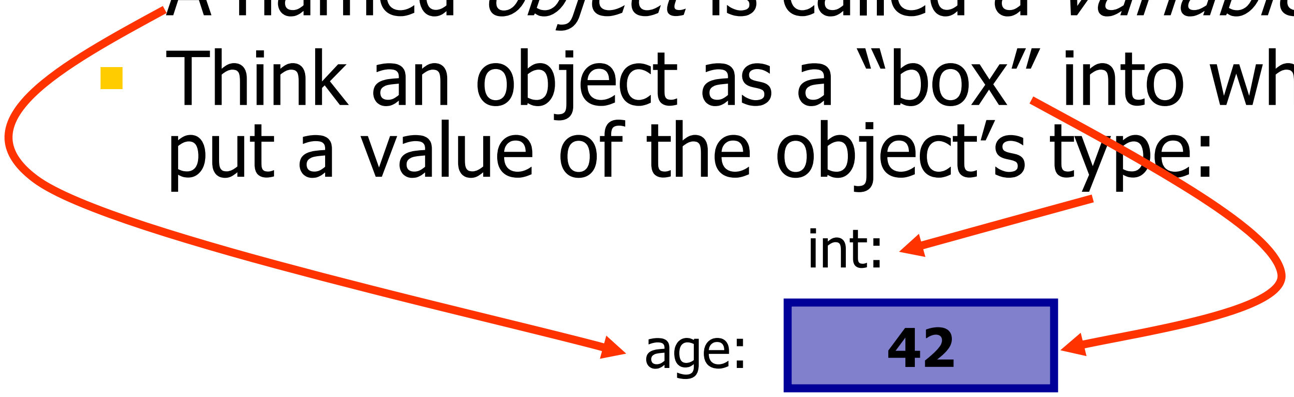
- Where are the libraries?
  - Usually in: /lib, /usr/lib, /usr/local/lib
  - They can be any place. (Use -L –l to specify where)
- …. And the include?
  - Like above, in the corresponding directory under */,/usr,/usr/local*: include
- NOTE! Library names must be prefixed by *lib* and use extensions *.a* or *.so*: liblibname.a or liblibname.so.

Breve intro. Alternative

- For Windows lovers ?????
  - Developing "free software" (means no limitation to the user) GNU ports the tools also under Win
  - Get a look to cygwin. A full Unix-like shell and tools: www.cygwin.com.
  - Xemacs: www.xemacs.org
  - Dev-c++ (old, www.bloodshed.net) or other IDE (Integrate Development Enviromnent) per C++ in Win or Linux (http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments).

Programming jargon

- To calculate something, we need somewhere to read and write into; i.e. we need a “place” in PC memory to read from or write to. We call such a “place” an *object*.
- An *object* is a region of memory with a *type* that specified what kind of information can be placed in it.
- A named *object* is called a *variable.*
- Think an object as a “box” into which you can put a value of the object’s type:



Programming jargon

- The most basic building block of a program is an *expression*.
- An *expression* computes a value from a number of operands.
- A part of a code that specifies an action is called a *statement*.

Programming example

- Let’s write a program to solve the quadratic equation:

$$ax^2 + bx + c = 0, a \neq 0.$$

- We already know the solutions:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- How many *objects* are present in each of the above equations?
  - 4?
  - 13?
  - 0?
- How many *expressions*?
  - 1?
  - 10?
  - 14?

Programming example

- We need objects for:
  1. a;
  2. b;
  3. c;
  4. x_1;
  5. $-b$;
  6. $2a$;
  7. ac;
  8. $4ac$;
  9. b^2;
  10. $b^2 - 4ac$;
  11. $\sqrt{b^2 - 4ac}$;
  12. $-b + \sqrt{b^2 - 4ac}$;
  13. $\frac{-b+\sqrt{b^2-4ac}}{2a}$.

- We need expressions for:
  1. Introducing the variable a;
  2. Introducing the variable b;
  3. Introducing the variable c;
  4. Introducing the variable x_1;
  5. $-b$;
  6. $2a$;
  7. ac;
  8. $4ac$;
  9. b^2;
  10. $b^2 - 4ac$;
  11. $\sqrt{b^2 - 4ac}$;
  12. $-b + \sqrt{b^2 - 4ac}$;
  13. $\frac{-b+\sqrt{b^2-4ac}}{2a}$;
  14. $x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}$.

Programming example

- So we can write our first procedure (assuming grouping of expression into statements:
  1. Introduce the variable a;
  2. Introduce the variable b;
  3. Introduce the variable c;
  4. Introduce the variable x_1;
  5. $\boxed{\text{Assign a value to } a;}$
  6. Assign a value to b;
  7. Assign a value to c;
  8. Evaluate b^2 and subtract $4ac$;
  9. $\boxed{\text{Evaluate square root of the result of the previous statement;}}$
  10. Evaluate $-b$ and sum it to the result of the previous statement;
  11. $\boxed{\text{Evaluate } 2a \text{ and divide the result of the previous statement by it;}}$
  12. Assign the result of the previous statement to x_1.

Will work?

I need to introduce some checks before using it!

Programming example

- So we can write our first procedure (assuming grouping of expression into statements:
  1. Introduce the variable a;
  2. Introduce the variable b;
  3. Introduce the variable c;
  4. Introduce the variable x_1;
  5. Assign a value to a;
  6. Check that a contains a value greater than 0;
  7. Assign a value to b;
  8. Assign a value to c;
  9. Evaluate b^2 and subtract $4ac$;
  10. Check that the result of the previous statement is greater than 0;
  11. Evaluate square root of the result of the previous statement;
  12. Evaluate $-b$ and sum it to the result of the previous statement;
  13. Evaluate $2a$, check that the result is greater than 0, and divide the result of the previous statement by it if it is greater than 0;
  14. Assign the result of the previous statement to x_1.

Will work?

Are we sure a CPU knows how to calculate powers and square roots?

Base programming grammar

- Minimal code: *int main() { return 0;}*
  - Defines a module (function):
    - Named: *main;*
    - Without any formal calling argument;
    - Does nothing;
    - Returns an integer value to the shell.
  - All C++ (and C) programs must have a sterring function called: *main()*
  - The program starts running jumping to this function execution.

Base programming grammar

- Minimal code: *int main() { return 0; }*
  - If no value is returned to the shell, the system interprets it as a successful execution.
  - Usually, by convention, a returned value different than zero, signals an error in the program execution !!!!!

Base programming grammar

- Minimal code: *int main() { return 0;}*
- Curly brackets: *{ }*, tag a code group, the beginning or the end of a module. They delimit a *scope* (**scope** (skōpe), n. **1**. the extent or range of one's understanding. **2**. the area of extent covered by something **3**. opportunity or freedom for movement or activity), mark the beginning and end of a function, struct, class …

Base programming grammar

- Minimal code: *int main() { return 0;}*
- Defines the module in which scope names, *i.e.* variables, have validity.

Base programming grammar

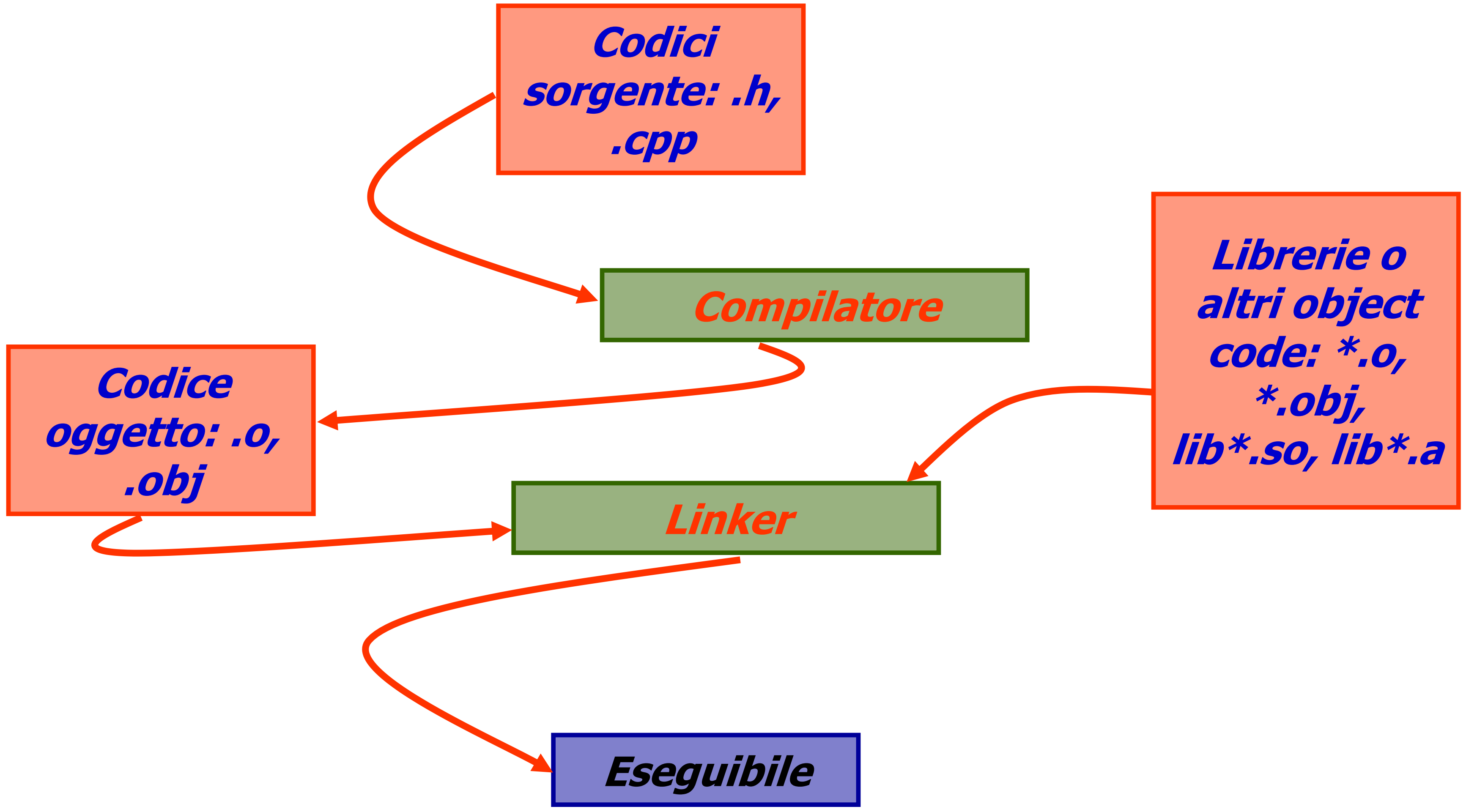
- The code delimited by a pair of *{ }* it is called block.

```
int main() {
  {
        // empty block
  }
  return 0;
}
```

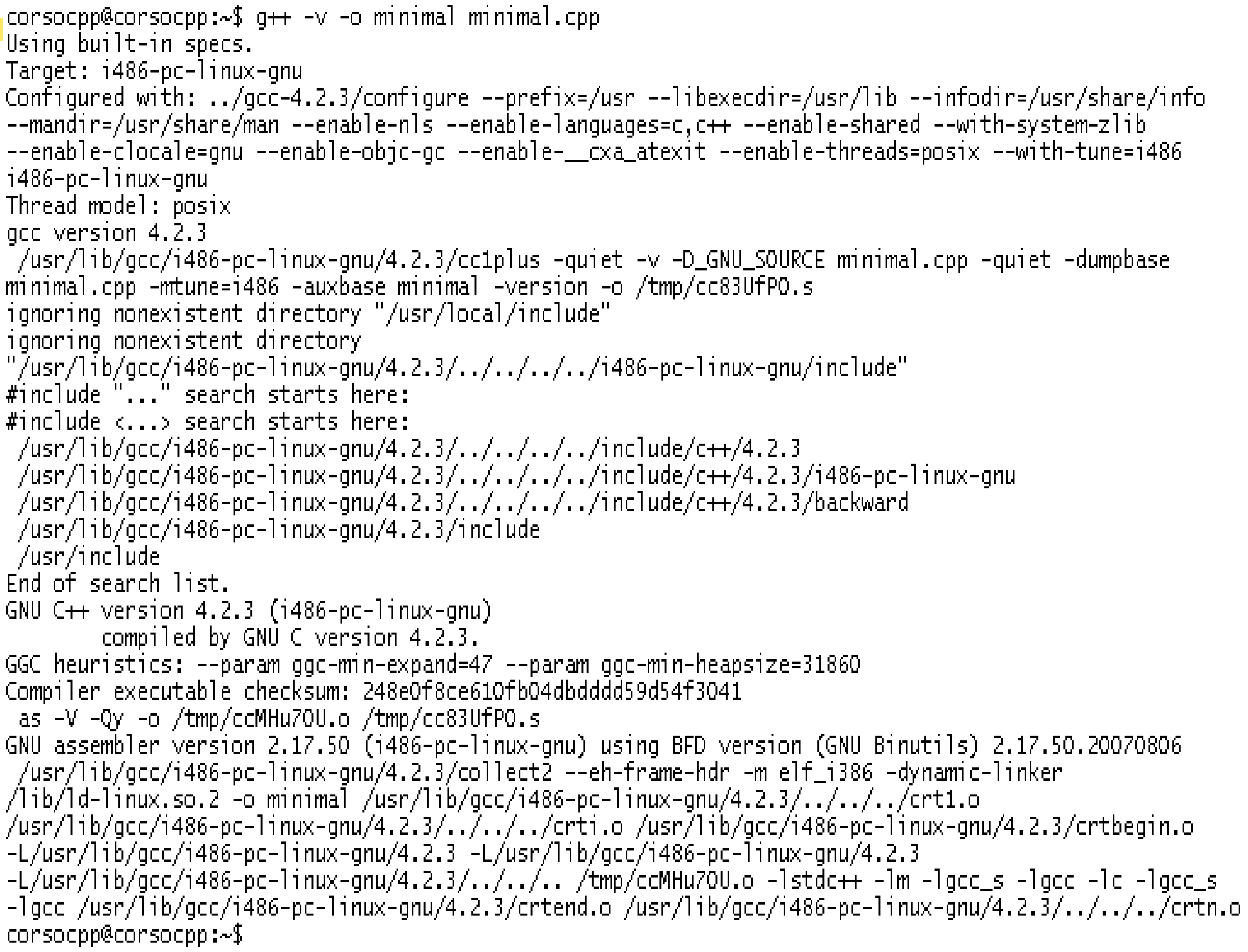
- Let's compile it:

```
g++ -v –o minimal minimal.cpp
```

Compiling and linking


Codici sorgente: .h, .cpp
Compilatore
Codice oggetto: .o, .obj
Librerie o altri object code: *.o, *.obj, lib*.so, lib*.a
Linker
Eseguibile

Compiling and linking



```
corsocpp@corsocpp:~$ g++ -v -o minimal minimal.cpp
Using built-in specs.
Target: i486-pc-linux-gnu
Configured with: ../gcc-4.2.3/configure --prefix=/usr --libexecdir=/usr/lib --infodir=/usr/share/info
--mandir=/usr/share/man --enable-nls --enable-languages=c,c++ --enable-shared --with-system-zlib
--enable-clocale=gnu --enable-objc-gc --enable-__cxa_atexit --enable-threads=posix --with-tune=i486
i486-pc-linux-gnu
Thread model: posix
gcc version 4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/cc1plus -quiet -v -D_GNU_SOURCE minimal.cpp -quiet -dumpbase
minimal.cpp -mtune=i486 -auxbase minimal -version -o /tmp/cc83UfPO.s
ignoring nonexistent directory "/usr/local/include"
ignoring nonexistent directory
"/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../i486-pc-linux-gnu/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/i486-pc-linux-gnu
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/backward
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/include
 /usr/include
End of search list.
GNU C++ version 4.2.3 (i486-pc-linux-gnu)
        compiled by GNU C version 4.2.3.
GGC heuristics: --param ggc-min-expand=47 --param ggc-min-heapsize=31860
Compiler executable checksum: 248e0f8ce610fb04dbdddd59d54f3041
 as -V -Qy -o /tmp/ccMHu7OU.o /tmp/cc83UfPO.s
GNU assembler version 2.17.50 (i486-pc-linux-gnu) using BFD version (GNU Binutils) 2.17.50.20070806
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/collect2 --eh-frame-hdr -m elf_i386 -dynamic-linker
/lib/ld-linux.so.2 -o minimal /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crt1.o
/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crti.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/crtbegin.o
-L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3 -L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3
-L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../.. /tmp/ccMHu7OU.o -lstdc++ -lm -lgcc_s -lgcc -lc -lgcc_s
-lgcc /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/crtend.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crtn.o
corsocpp@corsocpp:~$
```

Compiling and linking

Prompt

Comando

Opzioni Comando

```
corsocpp@corsocpp:~$ g++ -v -o minimal minimal.cpp
```

Preprocessore & compilatore

Assembler

Linker

Compiling and linking

```
corsocpp@corsocpp:~$ g++ -v -o minimal minimal.cpp
Using built-in specs.
Target: i486-pc-linux-gnu
Configured with: ../gcc-4.2.3/configure --prefix=/usr --libexecdir=/usr/lib --infodir=/usr/share/info
--mandir=/usr/share/man --enable-nls --enable-languages=c,c++ --enable-shared --with-system-zlib
--enable-clocale=gnu --enable-objc-gc --enable-__cxa_atexit --enable-threads=posix --with-tune=i486
i486-pc-linux-gnu
Thread model: posix
gcc version 4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/cc1plus -quiet -v -D_GNU_SOURCE minimal.cpp -quiet -dumpbase
minimal.cpp -mtune=i486 -auxbase minimal -version -o /tmp/cc83UfPO.s
ignoring nonexistent directory "/usr/local/include"
ignoring nonexistent directory
"/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../i486-pc-linux-gnu/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/i486-pc-linux-gnu
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/backward
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/include
 /usr/include
End of search list.
GNU C++ version 4.2.3 (i486-pc-linux-gnu)
        compiled by GNU C version 4.2.3.
GGC heuristics: --param ggc-min-expand=47 --param ggc-min-heapsize=31860
Compiler executable checksum: 248e0f8ce610fb04dbdddd59d54f3041
 as -V -Qy -o /tmp/ccMHu7OU.o /tmp/cc83UfPO.s
GNU assembler version 2.17.50 (i486-pc-linux-gnu) using BFD version (GNU Binutils) 2.17.50.20070806
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/collect2 --eh-frame-hdr -m elf_i386 -dynamic-linker
/lib/ld-linux.so.2 -o minimal /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crt1.o
/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crti.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/crtbegin.o
-L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3 -L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3
-L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../.. /tmp/ccMHu7OU.o -lstdc++ -lm -lgcc_s -lgcc -lc -lgcc_s
-lgcc /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/crtend.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crtn.o
corsocpp@corsocpp:~$
```

Preprocessore & compilatore

Output in assembler

Output in object code

Assembler

Linker

Eseguibile

Debugging

- We can follow “sep by step” an executable evolution and check its parameters, like memory, uing a *debugger*. The GNU suite *debugger* is: *gdb* .
- To use it in an efficient way, the executable must be created using the *gcc* option: *-g*, this option produce extra information in the executable, like, for example the source code corresponding to an machine code block, that can be interpreted by *gdb:*

```
g++ -g –o minimal minimal.cpp
```

Debugging

- We use the debugger specifying the name of the executable we would like to debug:
  gdb minimal
- There is no graphic user interface. Let's start using the help typying: *help*
- Most used commands include:
  - *break:* declares an address or a line in the code we like to stop the execution;
  - *print:* shows a variable content or the value resulting evaluating a statement;
  - *next:* continues program execution up the next code line, can be shortened to: *n;*
  - *step:* continues program execution until a new code line has to be excexuted, can be shortened to: *s;*
  - *continue:* continues program execution up to the next break point;
  - *list:* lists the source code.
- Let's read the help present for each command.

C++ Historical note and the basic grammar

Historical note

- Who is he?
  - Bjarne Stroustrup, C++ daddy
  - Born in the 1980: "C with Classes", starts to be circulated in the 1983, in the same year was named: C++..
- Since 1990, committees have been created to define C++ standards
  - " … was invented because I wanted to write some event-driven simulations for wich Simula67 would have been ideal, except for efficiency considerations."

Historical note

"C++ was designed primarily so that my friends and I would not have to program in assembler, C, or various modern high-level languages. Its main purpose was to make writing good programs easier and more pleasant for the individual programmer."

B.S., *The C++ Programming language 3rd ed.*

What is C++?

- What is C++ ?
  - C++ is a general-purpose programming language with a bias towards systems programming that:
    - Is a better C;
    - Supports data abstraction;
    - Supports object-oriented programming;
    - Supports generic programming

What is C++?

- What is C++ ?
  - **Procedural Programming**, that is: *Decide which procedures you want; use the best algorithms you can find.*
    - The focus is on the processing, i.e. the algorithm needed to perform the desired computation. A procedural language support this paradigm by providing facilities for passing arguments to functions and returning values from them.
    - C++ improves C, as a procedural language, i.e. I can write a C code.

What is C++?

- What is C++ ?
  - **Modular Programming**, that is: *Decide which modules you want; partition the program so that data is hidden within modules.*
    - Increasing code complexity calls for the need of modularization. Algorithms can be subdivided into blocks implementing part of the procedure hiding data needed in this blocks.
    - Programmer must provide a module interface so the other blocks can use the module, via the interface, or the externally modifiable data, hidden in the module.

What is C++?

- What is C++ ?
  - **Modular Programming**, that is: *Decide which modules you want; partition the program so that data is hidden within modules.*
    - In C++ modules and data can be grouped into namespaces, implementing interfaces and distinguish module of data with identical names but caming from different interfaces.
    - Often modules became so complex that are difficult to maintain. An interface using simply a namespace is not enough.
    - C++ provides facilities to implement a module as an user defined type.

What is C++?

- What is C++ ?
  - **User-Definited Types**, that is: *Decide which types you want; provide a full set of operations for each type.*
    - C++ provides the same support of the base type (integers, floating points, characters etc. etc.) to user defined complex type, so it is possible to use the same rules to manage these types. The most complex of this type is a class
    - Very often modules became user defined types! So that a programmer can store together data and code! Data and procedure implementing calculation on these data. As well as the base type, a module can be dynamically created! When I need the code I can create a memory area to store it along with the data.

What is C++?

- What is C++ ?
  - **User-Definited Types**, that is: *Decide which types you want; provide a full set of operations for each type.*
    - The increase of a type complexity increase the need to safe data and hide procedure details to the user. We want the user focus to be on interfaces, not on the procedure details. Interface must be kept as stable and generic as possible so to avoid changes also if the internal module structure is changing.
    - C++, exploiting the usage of *data abstraction, provides* facilities to create generic interfaces that can have actual different implementations according to the type used.

What is C++?

- What is C++ ?
  - **Object-Oriented Programming**, that is: *Decide which classe you want; provide a full set of operations for each class; make commonality explicit by using inheritance.*
    - If abstract data can be defined also virtual class (abstract) can be. This class do not correspond to any actual class but define an interface that can be used by an user without specification of the actual type.
    - If I have to write code to manage *Inmate*, *Physician*, *Paramedic* and *Employees*, to count the accesses into a building, can I write a code that deals only with *Persons* ?

What is C++?

- What is C++ ?
  - **Generic Programming**, that is: *Decide wich algorithms you want; parametrize them so that they work for a variety of suitable types and data structures.*
    - C++ provides the *template* construct, enabling the possibility of building classes independent from the type they are using.
    - The majority of the C++ libraries: STL (Standard Template Library), exploit this mechanism and provide objects, generic algorithms or facilities that can be specialized by users to a specific type. Because C++ offers to the user defined type the same support of the base type, both can be used.