

Laboratorio di Tecnologie dell'Informazione

Ing. Marco Bertini marco.bertini@unifi.it http://www.micc.unifi.it/bertini/



How the compiler works Programs and libraries



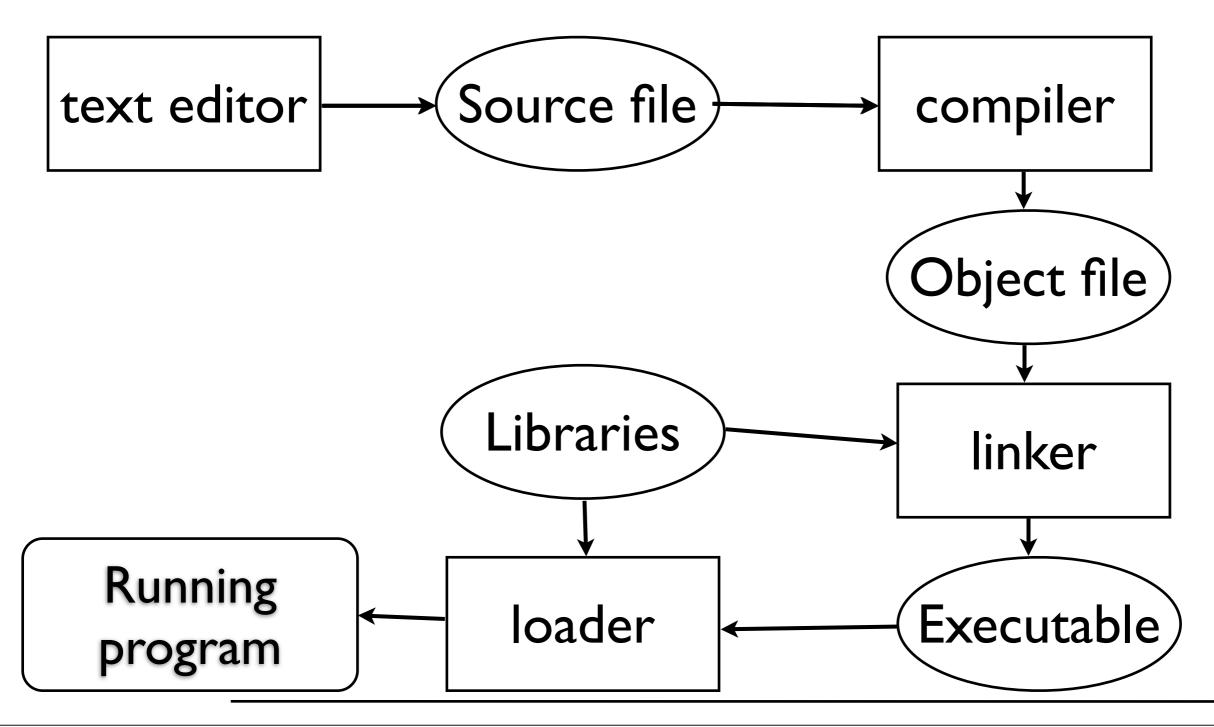
The compiler

"In C++, everytime someone writes ">> 3" instead of "/ 8", I bet the compiler is like, "OH DAMN! I would have never thought of that!"
- Jon Shiring (Call of Duty 4 / MW2)

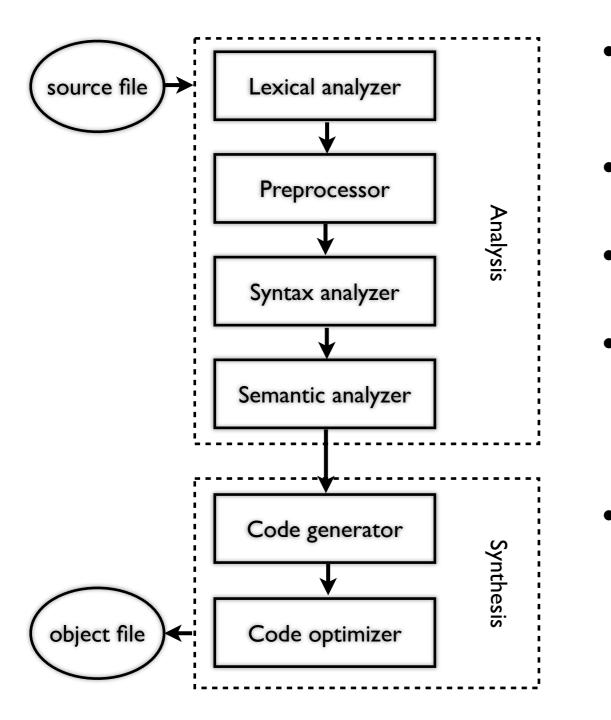
What is a compiler ?

- A compiler is a computer program (or set of programs) that translate source code from a high-level programming language to a lower level language (e.g., assembly language or machine code).
- A compiler typically performs: lexical analysis (tokenization), preprocessing, parsing, semantic analysis, code generation, and code optimization.

From source code to a running program



From source to object file



- Lexical analysis: breaks the source code text into small pieces called tokens. Each token is a single atomic unit of the language, for instance a keyword, identifier or symbol name.
- Preprocessing: in C/C++ macro substitution and conditional compilation
- Syntax analysis: token sequences are parsed to identify the syntactic structure of the program.
- Semantic analysis: semantic checks such as type checking (checking for type errors), or object binding (associating variable and function references with their definitions), or definite assignment (requiring all local variables to be initialized before use), rejecting incorrect programs or issuing warnings
- Code generation: translation into the output language, usually the native machine language of the system. This involves resource and storage decisions (e.g. deciding which variables to fit into registers and memory), and the selection and scheduling of appropriate machine instructions and their associated addressing modes. Debug data may also need to be generated to facilitate debugging.
- Code optimization: transformation into functionally equivalent but faster (or smaller) forms.

How the compilation works

Consider the following C++ program (e.g. stored in hello.cpp):

#include <iostream>

#define ANSWER 42

```
using namespace std;
// this is a C++ comment
int main() {
```

cout << "The Answer to the Ultimate Question of Life, the Universe, and Everything is " << ANSWER << endl;

```
return 0;
```

}

How the compilation works

 Conside Check the result of the preprocessor stored ilusing:

#define AN the iostream file is included at the using name beginning of the output, then there's the // this is int main() cout substituted define.

Life, the Universe, and Everything is " << ANSWER << endl;

```
return 0;
```

}

Assembly/object creation

Check the assembly output of the program with:
 g++ -S hello.cpp

The object file is created with:
 g++ -c hello.cpp

Assembly/object creation

Check the assembly output of the program with:
 g++ -S hello.cpp

The object file is created with:
 g++ -c hello.cpp

Performs all the compilation steps (also preprocessing)

Linking

- Use a linker like Id to link the libraries to the object file; on Ubuntu try:
 ld -lstdc++ hello.o -o hello
- or use g++ linking (will add required standard libraries):
 g++ hello.o -o hello

Linking

- Use a linker like Id to link the libraries to the object file; on Ubuntu try:
 ld -lstdc++ hello.o -o hello
- or use g++ linking (will add required standard libraries):
 g++ hello.o -o hello

add -v to g++ linking to see what's going on with ld

Linking

- The linker will merge the object files of various sources, e.g. if the program was split in more than one translation unit
- You must tell where object files and libraries are stored
 - the linker will check some default directories for libraries

Optimize and debug

- Add debug information to the output: it will help when debugging a program: when using g++ add the -G flag
- Request g++ optimization with the flags -0x (x=1...3) for fast execution or -0S for optimized size

Optimize and debug

Always use it when developing and testing a program !

- Add debug information to the output: it will help when debugging a program: when using g++ add the -G flag
- Request g++ optimization with the flags -0x (x=1...3) for fast execution or -0S for optimized size

Optimize and debug

Always use it when developing and testing a program !

- Add debug information to the output: it will help when debugging a program: when using g++ add the -G flag
- Request g++ optimization with the flags -0x (x=1...3) for fa Compilation becomes slower optimized size and slower...

Typically optimization is set when releasing a program



Libraries



- A software library is a set of software functions used by an application program.
- Libraries contain code and data that provide services to independent programs.
 - This encourages the sharing and changing of code and data in a modular fashion, and eases the distribution of the code and data.

Using libraries in C/C++

- To use a library in C/C++ you need to:
- Include the headers that provide the prototypes of functions and classes that you need in your code
- 2. Tell the linker where are the library files (and which files if the library is composed by more than one) that are needed by your code
 - In C++ some libraries are made only by header files... more information when studying C++ templates.

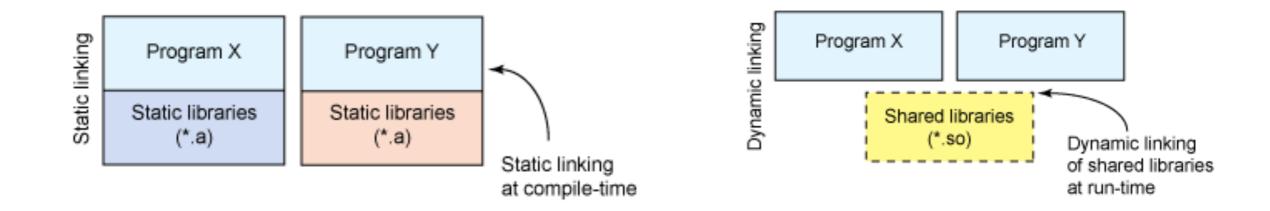
The C++ Standard Library

- In C++, the C++ Standard Library is a collection of classes and functions, which are written in the core language and part of the C++ ISO Standard itself.
- The C++ Standard Library provides
 - several generic containers, functions to utilise and manipulate these containers;
 - generic strings and streams (including interactive and file I/O);
 - support for some language features, and math.



- O.S.es like Linux, OS X and Windows support two types of libraries, each with its own advantages and disadvantages:
- The static library contains functionality that is bound to a program statically at compile time.
- The dynamic/shared library is loaded when an application is loaded and binding occurs at run time.
- In C/C++ you also have header files with the prototypes of the functions/classes that are provided by the library

Static vs. Dynamic linking



 To check if a program is statically or dynamically linked, and see what dynamic libraries are linked use ldd (Linux) or otool (OS X):

[ian@echidna ~]\$ ldd /sbin/sln /sbin/ldconfig /bin/ln
/sbin/sln:

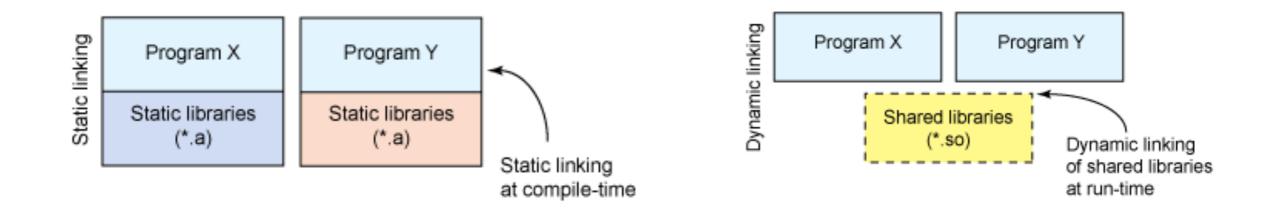
not a dynamic executable
/sbin/ldconfig:

not a dynamic executable

/bin/ln:

```
linux-vdso.so.1 => (0x00007fff644af000)
libc.so.6 => /lib64/libc.so.6 (0x00000037eb800000)
/lib64/ld-linux-x86-64.so.2 (0x0000037eb400000)
```

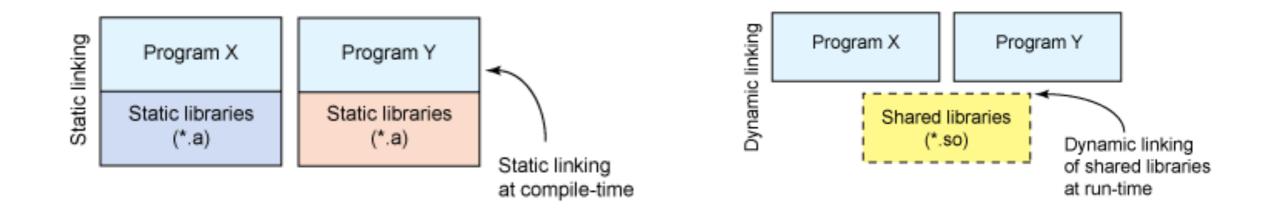
Static vs. Dynamic linking



 To check if a program is statically or dynamically linked, and see what dynamic libraries are linked use ldd (Linux) or otool (OS X):

[ian@echidna ~]\$ ldd /sbin/sln /sbin/ldconfig /bin/ln
/sbin/sln:
 not a dynamic executable
/sbin/ldconfig:
 not a dynamic executable
/bin/ln:
 linux-vdso.so.1 => (0x00007fff644af000)
 libc.so.6 => /lib64/libc.so.6 (0x0000037eb800000)
 /lib64/ld-linux-x86-64.so.2 (0x0000037eb400000)

Static vs. Dynamic linking



 To check if a program is statically or dynamically linked, and see what dynamic libraries are linked use ldd (Linux) or Otool (OS X):

[ian@echidna ~]\$ ldd /sbin/sln /sbin/ldconfig /bin/ln	
/sbin/sln:	
not a dynamic executable	-Static linking
/sbin/ldconfig:	
not a dynamic executable	
/bin/ln: 🗸	-Dynamic linking
linux-vdso.so.1 => (0x00007fff644af000) 🔨	-//
libc.so.6 => /lib64/libc.so.6 (0x00000037eb800000) /lib64/ld-linux-x86-64.so.2 (0x00000037eb400000)	^{>} Dynamic libraries



- Static libraries are simply a collection of ordinary object files; this collection is created using an archiver program (e.g. ar in *NIX systems).
 - Conventionally, static libraries end with the ".a" suffix (*NIX system) or ".lib" (Windows).
- Static libraries permit users to link to programs without having to recompile its code, saving recompilation time. There's no need to install libraries along with programs.
- With a static library, every running program has its own copy of the library.



- Statically linked programs incorporate only those parts of the library that they use (not the whole library!).
- To create a static library, or to add additional object files to an existing static library, use a command like this:
 - ar rcs my_library.a file1.o file2.o
- The library file is used by the linker to create the final program file

Statically linked executables

- Statically linked executables contain all the library functions that they need to execute:
 - all library functions are linked into the executable.
- They are complete programs that do not depend on external libraries to run:
 - there is no need to install prerequisites.

Dynamic/Shared libraries

- Dynamic/Shared libraries are libraries that are loaded by programs when they start.
- They can be shared by multiple programs.
- Shared libraries can save memory, not just disk space. The O.S. can keep a single copy of a shared library in memory, sharing it among multiple applications. That has a pretty noticeable effect on performance.

Shared library versions

- Shared libraries use version numbers to allow for upgrades to the libraries used by applications while preserving compatibility for older applications.
- Shared objects have two different names: the soname and the real name. The soname consists of the prefix "lib", followed by the name of the library, a ".so" followed by another dot, and a number indicating the major version number (in OS X the dotted numbers precede the ".dylib" extension). The real name adds to the soname a period, a minor number, another period, and the release number. The last period and release number are optional.
- There is also the *linker name*, which may be used to refer to the soname without the version number information. Clients using this library refer to it using the linker name.

Why using library versions ?

- The major/minor number and release number support configuration control by letting you know exactly what version(s) of the library are installed. With a statically linked executable, there is some guarantee that nothing will change on you. With dynamic linking, you don't have that guarantee.
- What happens if a new version of the library comes out? Especially, what happens if the new version changes the calling sequence for a given function?
- Version numbers to the rescue: when a program is linked against a library, it has the version number it's designed for stored in it. The dynamic linker can check for a matching version number. If the library has changed, the version number won't match, and the program won't be linked to the newer version of library.

Shared libraries paths

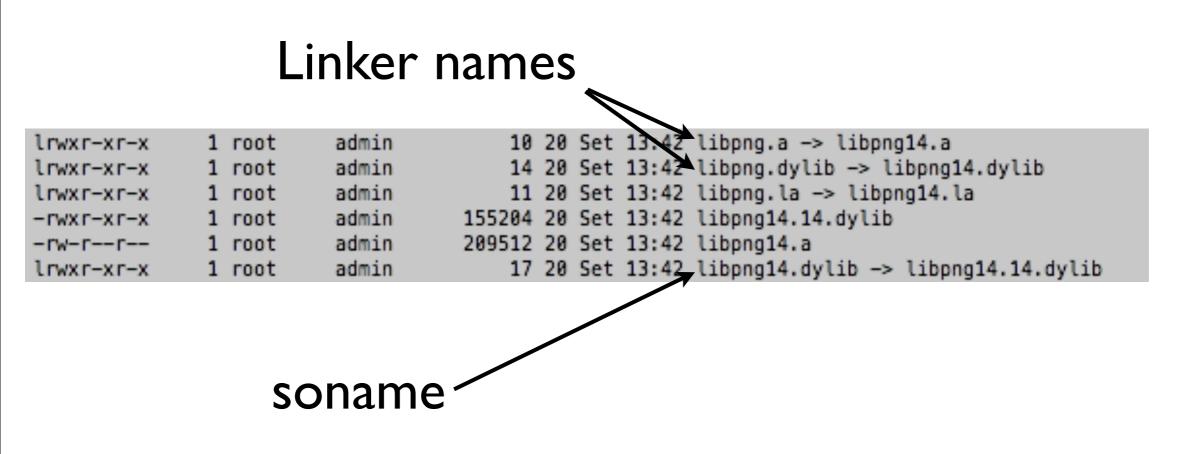
- Since linking is dynamic the library files should be somewhere they can be found by the O.S. dynamic linker
 - e.g. /usr/lib or /usr/local/lib
 - It's possible to add other directories to the standard library paths (e.g. using LD_LIBRARY_PATH or DYLD_LIBRARY_PATH environment variables)

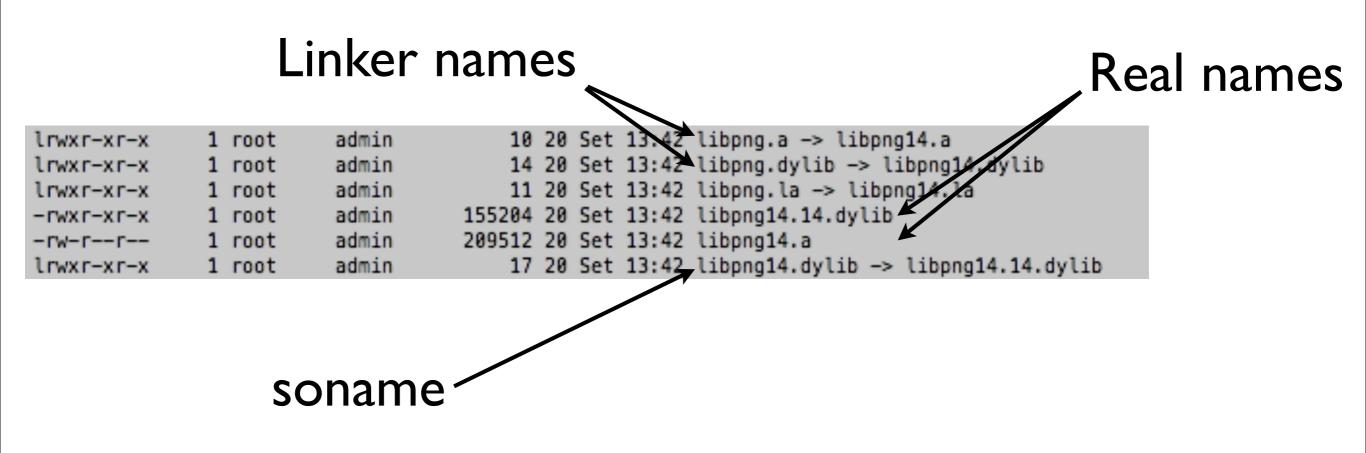
Dynamically linked executables

- Dynamically linked executables are smaller programs than statically linked executables:
 - they are incomplete in the sense that they require functions from external shared libraries in order to run.
- Dynamic linking permits a package to specify prerequisite libraries without needing to include the libraries in the package.
 - D.L. executables can share one copy of a library on disk and in memory (at running time). Most programs today use dynamic linking.

lrwxr-xr-x	1 root	admin	10 20	3 Set	13:42	libpng.a -> libpng14.a
lrwxr-xr-x	1 root	admin	14 20	0 Set	13:42	libpng.dylib -> libpng14.dylib
lrwxr-xr-x	1 root	admin	11 20	0 Set	13:42	libpng.la –> libpng14.la
-rwxr-xr-x	1 root	admin	155204 20	0 Set	13:42	libpng14.14.dylib
-rw-rr	1 root	admin	209512 20	0 Set	13:42	libpng14.a
lrwxr-xr-x	1 root	admin	17 20	0 Set	13:42	libpng14.dylib -> libpng14.14.dylib

	L	inker	names	
lrwxr-xr-x lrwxr-xr-x lrwxr-xr-x -rwxr-xr-x	1 root 1 root 1 root 1 root	admin admin admin admin	10 20 Set 13.42 libpng.a - 14 20 Set 13:42 libpng.dy 11 20 Set 13:42 libpng.la 155204 20 Set 13:42 libpng14.1	lib -> libpng14.dylib -> libpng14.la
-rw-rr lrwxr-xr-x	1 root 1 root	admin admin	209512 20 Set 13:42 libpng14.a	





Use libraries in Eclipse

00		Properties for TestMathLibrary	
type filter text	Settings		() + () + ▼
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging Settings Tool Chain Editor C/C++ General Code Coverage cppcheclipse Project Facets Project References Run/Debug Settings Task Repository Task Tags Validation WikiText 	Configuration: Debug [Active]	Tool Settings Build Steps Build Artifact Binary Parsers C Error Parsers Libraries (-1) stdc++ System Library search path (-L)	

These options are equivalent to command line -I and -L

Use libraries in Eclipse

⊖ ○ ○ Properties for TestMathLibrary			
type filter text	Settings	(
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging Settings Tool Chain Editor C/C++ General Code Coverage cppcheclipse Project Facets Project References Run/Debug Settings Task Repository Task Tags Validation WikiText 	Configuration: Debug [Active]	Manage Configurations Tool Settings Build Steps Build Artifact Binary Parsers Stror Parsers Libraries (-1) Stdc++ System Provide list of libraries (no trailing "lib") Library search path (-L)	

These options are equivalent to command line -I and -L

Use libraries in Eclipse

00		Properties for TestMathLibrary
type filter text	Settings	(
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging Settings Tool Chain Editor C/C++ General Code Coverage cppcheclipse Project Facets Project References Run/Debug Settings Task Repository Task Tags Validation WikiText 	Configuration: Debug [Active]	Ibrary search path (-L) I manage Configurations

These options are equivalent to command line -I and -L



Creating and using a library

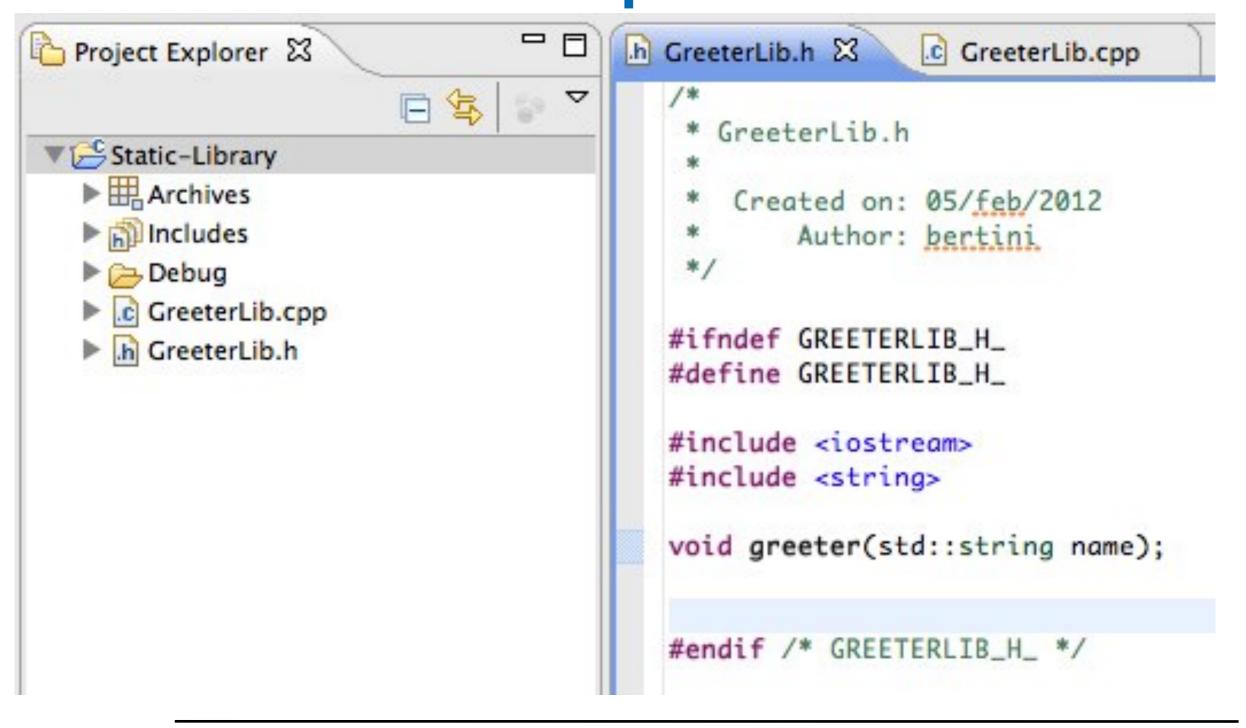
Writing a library

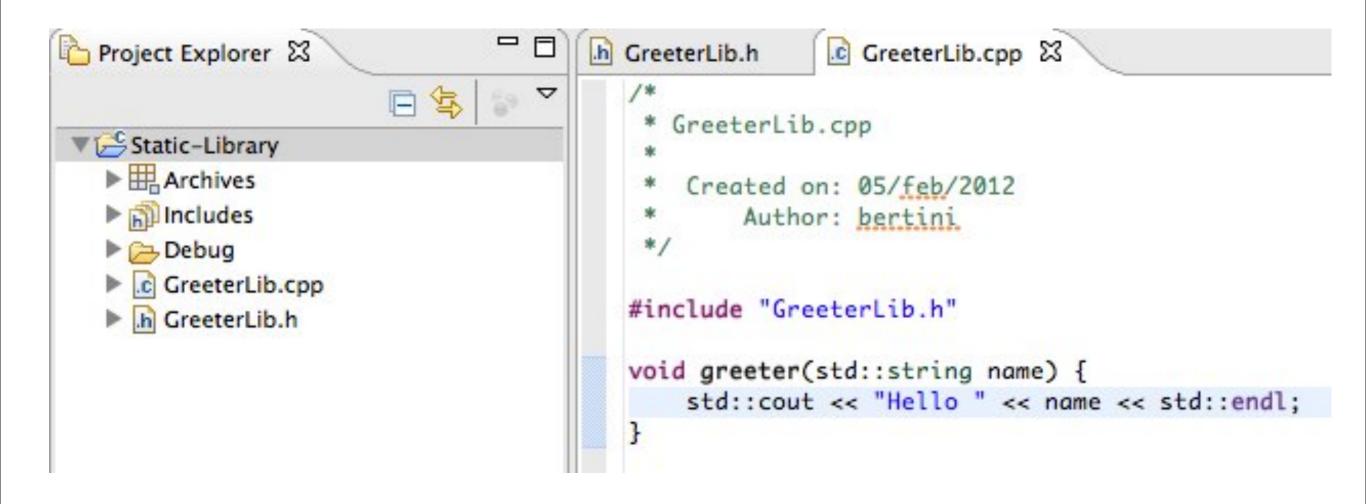
- There are basically two files that have to be written for a usable library:
- The first is a header file, which declares all the functions/classes/types exported by the library.
 - It will be included by the client in the code.
- The second is the definition of the functions/ classes to be compiled and placed as the shared object.
 - the object file created through compilation will be used by the linker, to create the library.

📬 • 😭 • 🔄 📾 📄 📸 • 😂 • दि •	
Makefile Project with Existing C	ode
🔂 C++ Project	
C Project	
Project	
Convert to a C/C++ Project (Ad	lds C/C++ Nature)
Convert to a C/C++ Autotools	Project
Source Folder	
😂 Folder	
C Source File	
h Header File	
File from Template	
G Class	
👕 Task	
📑 Example	
Cther	жN

● ○ ○ C++	Project
C++ Project Create C++ project of selected type	
Project name: Static-Librar Use default location Location: /Users/bertini/Documents/presen Choose file system: default	atazioni/lezioni/laboratorio i Browse
Project type:	Toolchains:
 Executable Empty Project Hello World C++ Project Cross-Compile Project Shared Library Static Library Empty Project Makefile project GNU Autotools 	Cross GCC MacOSX GCC
Show project types and toolchains only if the second secon	they are supported on the platform

00	C++ Project	
elect Configura		
Project type: Toolchains: Configurations:	Static Library MacOSX GCC	
 ✓ Sebug ✓ Selease 		Select all Deselect all
		Advanced settings
Additional config	ettings" button to edit project's properties. urations can be added after project creation figurations" buttons either on toolbar or on	
?	< Back Next >	Cancel Finish





CDT Build Console [Static-Library]	
**** Build of configuration Debug for project Static-Library ****	
make all Building file:/GreeterLib.cpp Invoking: GCC C++ Compiler	Compiling source to object file
g++ -00 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"GreeterLib.d" Finished building:/GreeterLib.cpp	-MT"GreeterLib.d" -o "GreeterLib.o" "/GreeterLib.cpp"
Building target: libStatic-Library.a Invoking: GCC Archiver ar -r "libStatic-Library.a" ./GreeterLib.o	 Creating library
ar: creating archive libStatic-Library.a Finished building target: libStatic-Library.a	
**** Build Finished ****	

000	Properties for Static-Library	
(type filter text	Settings	<pr +="" pr<="" td="" ⇒="" ▼<=""></pr>
 Resource Linked Resources Resource Filters Builders C/C++ Build Build Variables Discovery Options Environment Logging Settings Tool Chain Editor C/C++ General Code Coverage copcheclipse Project Facets Project References 	Settings Configuration: Debug [Active] Tool Settings Build Artifact Artifact Type: Static Library Artifact name: \${ProjName} Artifact extension: a Output prefix: lib	Manage Configurations
Run/Debug Settings Server Task Repository Task Tags Validation WikiText	You can change the default name of the library file	Restore Defaults Apply

- We need to tell the compiler where are the header files of the library
- We need to include the files in our client code
- We need to tell the linker where is the library file (".a") and the name of the library (remind the convention used !)
- Eclipse will use this information to create the required makefile

00	C+	+ Project
C++ Project Create C++ project of s	elected type	
Project name: Statical	y-Linked-Executab	le
Use default location		
Location: /Users/berti Choose file s		sentazioni/laboratorio i Browse
Project type:		Toolchains:
 Executable Empty Project Hello World C Cross-Compil Shared Library Static Library Makefile project GNU Autotools 	++ Project	MacOSX GCC
Show project types a	nd toolchains only	if they are supported on the platform Next > Cancel Finish

elect Configurat	ions	
Select platforms a	nd configurations you wish to deploy o	on 📃
	Executable MacOSX GCC	
Configurations:		
Debug 🛞 Release		Select all
() () neicuse		Deselect all
		Advanced cettings
		Advanced settings
Jse "Advanced set	tings" button to edit project's properti	es.
	rations can be added after project crea	
Jse "Manage confi	gurations" buttons either on toolbar o	r on property pages.

000	Prope	rties for Statically-Linked-Executable	
(type filter text	Settings		↓ ↓ ↓
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging 	Configuration: Debug [Active]	tings 🎤 Build Steps 🛛 🙅 Build Artifact 📄	Manage Configurations Manage Configurations
Settings Tool Chain Editor VC/C++ General Code Analysis Code Style Documentation Export Settings File Types Indexer Language Mappings Paths and Symbols Code Coverage Code Coverage Copcheclipse Project Facets Project References Run/Debug Settings Server Task Repository Task Tags Validation WikiText	 MacOS X C++ Linker General Libraries Miscellaneous Shared Library Settings GCC Assembler General GCC C++ Compiler Preprocessor Includes Optimization Symbols Includes Optimization Debugging Warnings Miscellaneous Symbols Includes Optimization Debugging Warnings Miscellaneous 	Include files (-include)	
			Restore Defaults Apply
?			Cancel OK

000	Properties for Statically-Linked-Executable	
(type filter text	Settings	↓ ↓ ↓ ↓
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging Sattings 	Configuration: Debug [Active]	Manage Configurations rsers & Error Parsers
Environment	MacOS X C++ Linker General Libraries Miscellaneous Shared Library Settings Add directory path Directory: [/Users/bertini/Documents/presentazioni/lezioni/laboratorio informatica 2011-20 Cancel Workspace File system OK Preprocessor Symbols Includes Optimization Debugging Warnings Miscellaneous	Restore Defaults Apply
?		Cancel OK

00	Properti	ies for Statically-Linked-Executable	
(type filter text	Settings		<> + ⇒ + ▼
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging 	Configuration: Debug [Active]	igs 🏾 🔊 Build Steps 🛛 🙅 Build Artifact 📄 🗟 Binary	Manage Configurations Parsers
Settings Tool Chain Editor C/C++ General Code Analysis Code Style Documentation Export Settings File Types Indexer Language Mappings Paths and Symbols Code Coverage Copecheclipse Project Facets Project References Run/Debug Settings Server Task Repository Task Tags Validation WikiText	 MacOS X C++ Linker General Libraries Miscellaneous Shared Library Settings GCC Assembler General GCC C++ Compiler Preprocessor Includes Optimization Debugging Warnings Miscellaneous Symbols Includes Optimization Debugging Warnings Miscellaneous Symbols Includes Optimization Debugging Warnings Miscellaneous Miscellaneous Miscellaneous Miscellaneous Miscellaneous Miscellaneous Miscellaneous Miscellaneous 	Libraries (-I) Library search path (-L)	Restore Defaults
?			Cancel OK

000	Prope	erties for Statically-Linked-Executable	
(type filter text	Settings		\$•\$• •
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging 	Configuration: Debug [Active]		Manage Configurations Error Parsers
Settings Tool Chain Editor C/C++ General Code Coverage cppcheclipse Project Facets Project References Run/Debug Settings Task Repository Task Tags Validation WikiText	MacOS X C++ Linker General Libraries Miscellaneous Shared Library Settings G C	Libraries (-I) Static-Library Edit Dialog Cancel OK */Users/bertini/Documents/presentazioni/lezioni/laboratorio in	Informatica 2011-2012/workspace/Static
			Restore Defaults Apply
?			OK Cancel

stdu cout as "Duranic ballos "	1000		riopenties	Tor sturrenty Entred Executes		
<pre>std::cout << "Dynamic hello: " << }</pre>	type filter text	Settings				$(\Box \star \Box)$
<pre>void _init() { std::cout << "Inside _init() - Dy }</pre>	 Resource Builders C/C++ Build Build Variables Discovery Options Environment 	Configuration: De	ebug [Active]		*]	Manage Configurations.
<pre>void _fini() { std::cout << "Inside _fini() - Dy }</pre>	Logging		neous Jibrary Settings	Build Steps Puild Artifaction Provide Artifact	t 🛛 <u>नि</u> Binary Parsers 🛛 🐼 Error Pa	rsers
			blor	Edit directory path		
0 🖸	Apri		and the second se			
	Select a folder from	m file system:			orkspace/Static-Library/Debug"	1
▶ 🔠 📰 📖 📰 🕇 Debug	\$	(٩			
Imagini -compiler_libraries Tutti i miei documenti rkspace Scrivania rkspace Documenti -compiler_libraries bertini -compiler_libraries Applicazioni -compiler_libraries Download -compiler_libraries Utility -compiler_libraries	s.key in .metadata Dynamic-Library DynamicallExect Static-Library Statically-LExect	Greeter	Lib.cpp	 GreeterLib.d GreeterLib.o libStatic-Library.a makefile objects.mk sources.mk subdir.mk 	ezioni/laboratorio informatica 20	111-2012/workspace/Stati
uova cartella	_			Annulla Apri	Resto	re Defaults Apply

000	Prop	perties for Statically-Linked-Executable	
type filter text	Settings		()+ ()+ +
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging 	Configuration: Debug [Active		Anage Configurations Parsers
Settings Tool Chain Editor C/C++ General Code Coverage cppcheclipse Project Facets Project References Run/Debug Settings Task Repository Task Tags Validation WikiText	 MacOS X C++ Linker General Libraries Miscellaneous Directory: Inrectory: GCC C Compiler Preprocessor Symbols Includes Optimization Debugging Warnings Miscellaneous 	Libraries (-I) Static-Library Edit directory path ments/presentazioni/lezioni/laboratorio informatica 2011 Cancel Workspace Library search path (-L) "/Users/bertini/Documents/presentazioni/lezioni/lal	File system OK
?			OK Cancel

lunedì 17 marzo 14

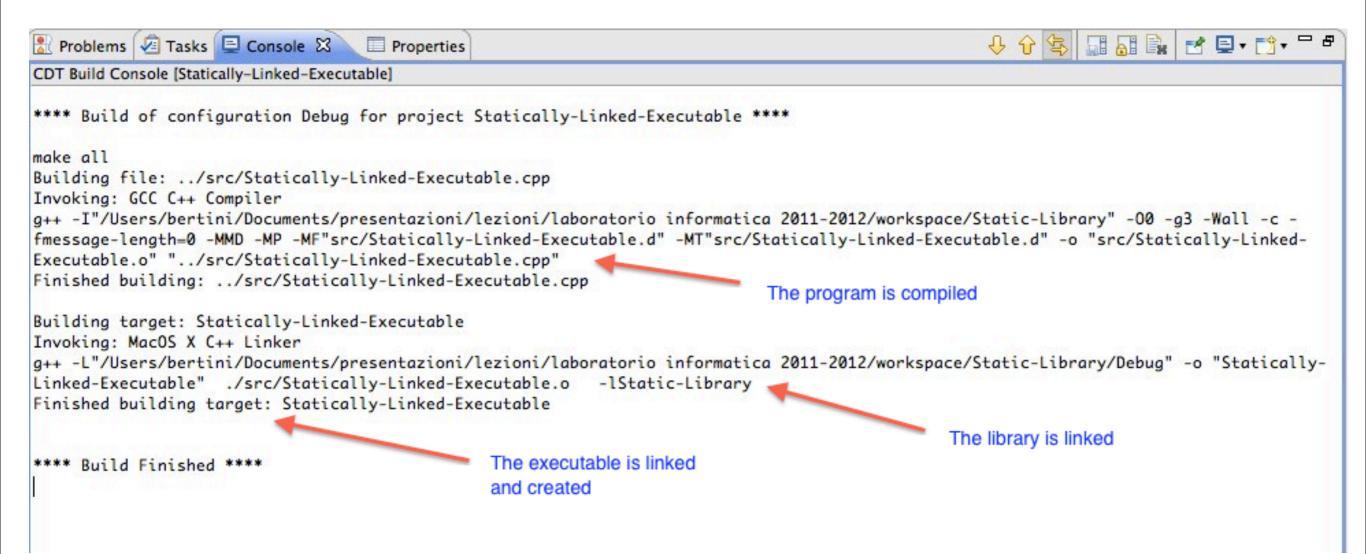
000	Proper	rties for Statically-Linked-Executable	
type filter text	Settings		<
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging Settings Tool Chain Editor C/C++ General Code Coverage cppcheclipse Project Facets Project References Run/Debug Settings Task Repository Task Tags Validation WikiText 	Configuration: Debug [Active]	tings 🍡 Build Steps 🙅 Build Artifact 🗟 Binary Parse	Anage Configurations ers Serror Parsers Service S
	Ceneral Libraries Miscellaneous Shared Library Settings Shared Library Settings Shared Library Settings Ceneral Ceneral Secondaria General Preprocessor Compiler Comp	Libraries (-I) Static-Library Library search path (-L) '/Users/bertini/Documents/presentazioni/lezioni/laborato	
	Debugging Warnings Miscellaneous		Restore Defaults Apply OK Cancel

\varTheta 🔿 🚱 C/C++ - Statically-Linked-Executable/src/Statically-Linked-Executable.cpp - Eclipse - /Users/bertini/Documents/p

- 0

╡ <mark>┎╩╺┎╩</mark> ╺┎╝╔╗╔╝ <mark>╔</mark> ╢╔ ┊╋┊╸╬╡╸╬╴╲╸╸╲╺	<u>*</u> * 🔂 • 🖸 •	♂ •]≪•⊗•] ৡ• ()• %₌• %₂•] 😕 😂 🔗 •] 🗾 🐨 🔳 🔳
Project Explorer		h GreeterLib.h	GreeterLib.cpp	☑ Statically-Linked-Executable.cpp 🛛
		11		

E Statically-Linked-Executable ▼ Static-Library ► Archives ► Debug	<pre>// Name : Statically-Linked-Executable.cpp // Author : // Version : // Copyright : Your copyright notice // Description : Hello World in C++, Ansi-style //</pre>
 Debug GreeterLib.cpp GreeterLib.h 	<pre>#include <iostream> #include "GreeterLib.h" int main() { greeter("Marco"); return 0;</iostream></pre>
	3



C++ Project	
C Project	
Project	
Convert to a C	/C++ Project (Adds C/C++ Nature)
Convert to a C	C++ Autotools Project
Source Folder	
😂 Folder	
Source File	
h Header File	
File from Temp	olate
G Class	
🕆 Task	
📑 Example	
T Other	æ

roject nar					
	fault location				
ocation:	/Users/bertini/Docun	nents/present	azioni/lezioni/labo	oratorio i Browse	8
	Choose file system:	default	\$		
roject typ			Toolchains:		
Exe			Cross GCC		_
_	Empty Project		MacOSX GCC		
👄 H	lello World C++ Proje	t			
	Cross-Compile Project				
_	red Library				
	mpty Project ic Library				
	efile project				
_	JAutotools				
GNU					
GNU					
GNU					

00	C++ Project	
elect Configu Select platform	rations s and configurations you wish to deploy on	
Project type: Toolchains: Configurations:	Shared Library MacOSX GCC	
 ✓ Sebug ✓ Selease 		Select all Deselect all
		Advanced settings
Additional confi	settings" button to edit project's properties. gurations can be added after project creation. nfigurations" buttons either on toolbar or on pr	operty pages.
?	< Back Next >	Cancel Finish

000	Properties for Dynamic-Library	
(type filter text (3)	Settings	• • • •
 Resource Builders C/C++ Build Build Variables Discovery Options Environment Logging 	Configuration: Debug [Active]	•
Settings Tool Chain Editor	Artifact Type: Shared Library	\$
►C/C++ General Code Coverage	Artifact name: \${ProjName}	•
▶ cppcheclipse	Artifact extension: dylib	•
Project Facets Project References Run/Debug Settings Server Task Repository Task Tags Validation WikiText	Output prefix: Iib You can change the default name of the library You have to write the major/minor and release number if you want to take advantage of the versioning naming convention	
	Restore Defaults	Apply
?	Cancel	ОК

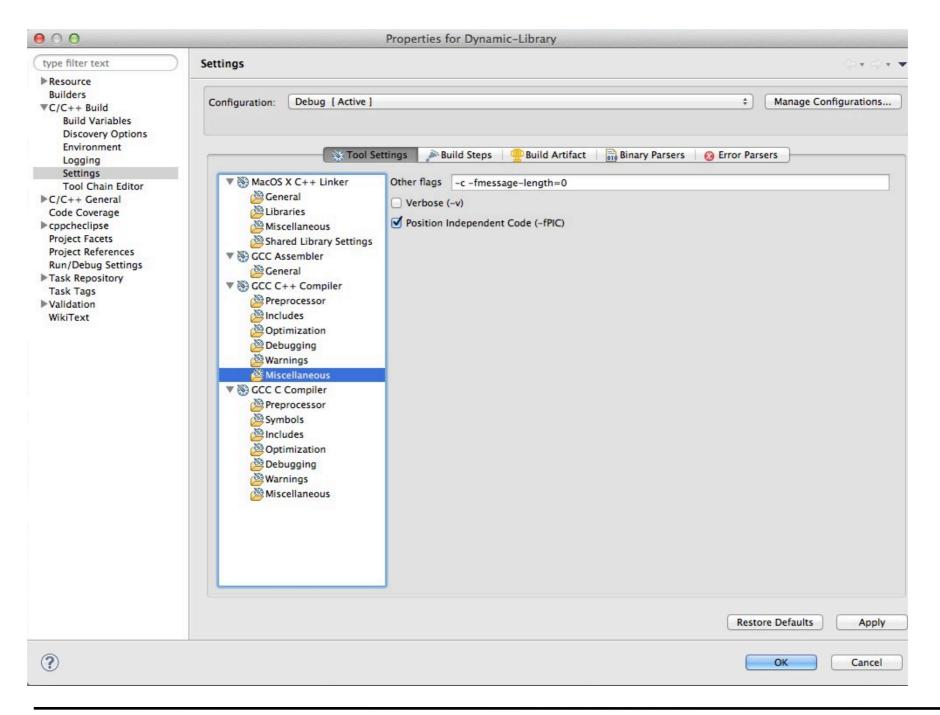
000	Properties f	for Dynamic-Library	
type filter text	Settings		
 Resource Builders C/C++ Build Build Variables Discovery Options 	Configuration: Debug [Active]		+
Discovery Options Environment Logging Settings Tool Chain Editor C/C++ General Code Coverage cppcheclipse Project Facets Project References Run/Debug Settings Server Task Repository Task Tags Validation WikiText	Artifact Type: Shared Library Artifact name: Greeter Artifact extension: dylib Output prefix: lib	uild Steps Build Artifact Binary Parsers R Fror Parsers	
		Restore Defaults	Apply
?		Cancel	ОК

Project Explorer	GreeterLib	o.h CreeterLib.cpp	Statically-Linked-Executa	ble.cpp 🛙
	000	New	Header File	
 Dynamic-Library Dincludes Statically-Linked-Executable Statically-Linked-Executable Statically-Linked-Executable Statically-Linked-Executable Debug Debug Static-Library Archives Debug Debug CreeterLib.cpp GreeterLib.h 	Header File Create a new	header file.		h
	Source folder: Header file: Template:	Dynamic-Library Greeter.h Default C++ header temp	late ÷	Browse Configure
	?		Cancel	Finish

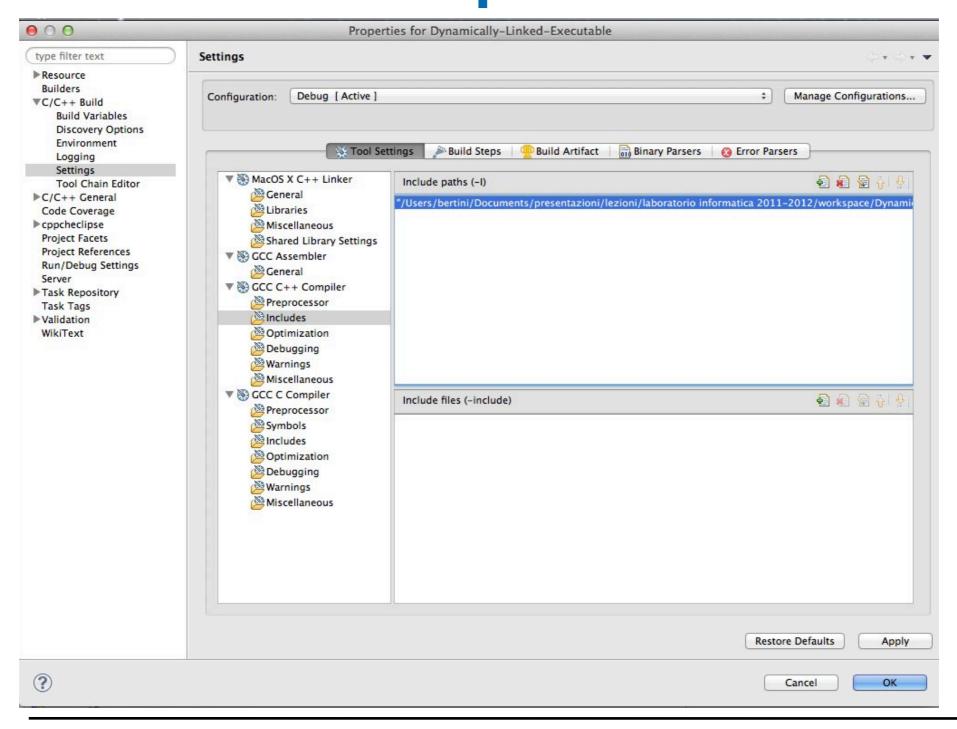
0 0 0 h C/C++ - Dynamic-Librar	y/Greeter.h - E	clipse – /Users/bertini/Documents/p	oresentazioni/le	zioni/laboratorio
] [] • 12 • 12 12 12 12] 12 • 12 • 12 • 12 • 12 • 12 • 12 • 12	? • ⊙ •] % •	⑧•] 棽• ○• ⅔• ዔ•] 😕 🖉 🖨	A•] 🥖 📚 🗉	T
Project Explorer 🕱 🗧 🗖	000	New Source File	· · ·	
Dynamic-Library	Source File Create a new	source file.		C
Statically-Linked-Executable Statically-Linked-Executable Sinaries Includes	Source folder: Source file:	Dynamic-Library Greeter.cpp		Browse
 Src Debug Static-Library Archives Includes Debug CreeterLib.cpp GreeterLib.h 	Template:	Default C++ source template	*	Configure
	?		Cancel	Finish

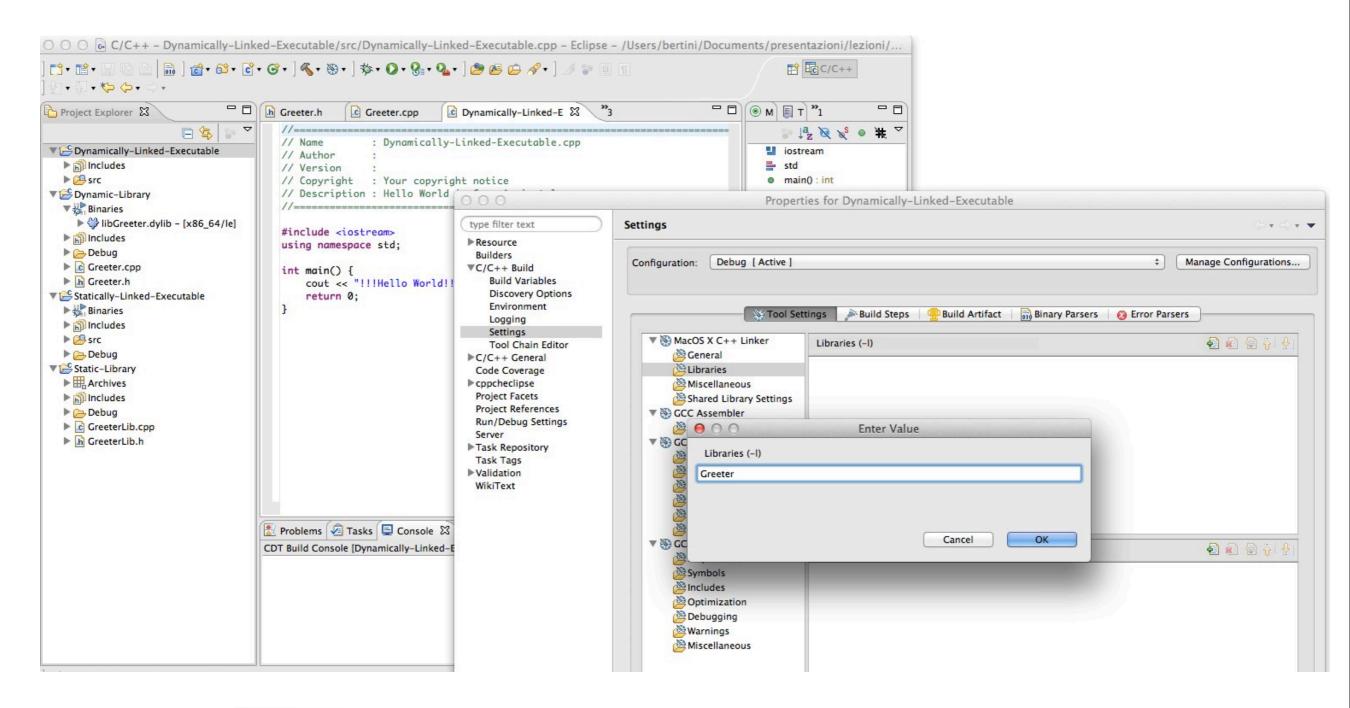


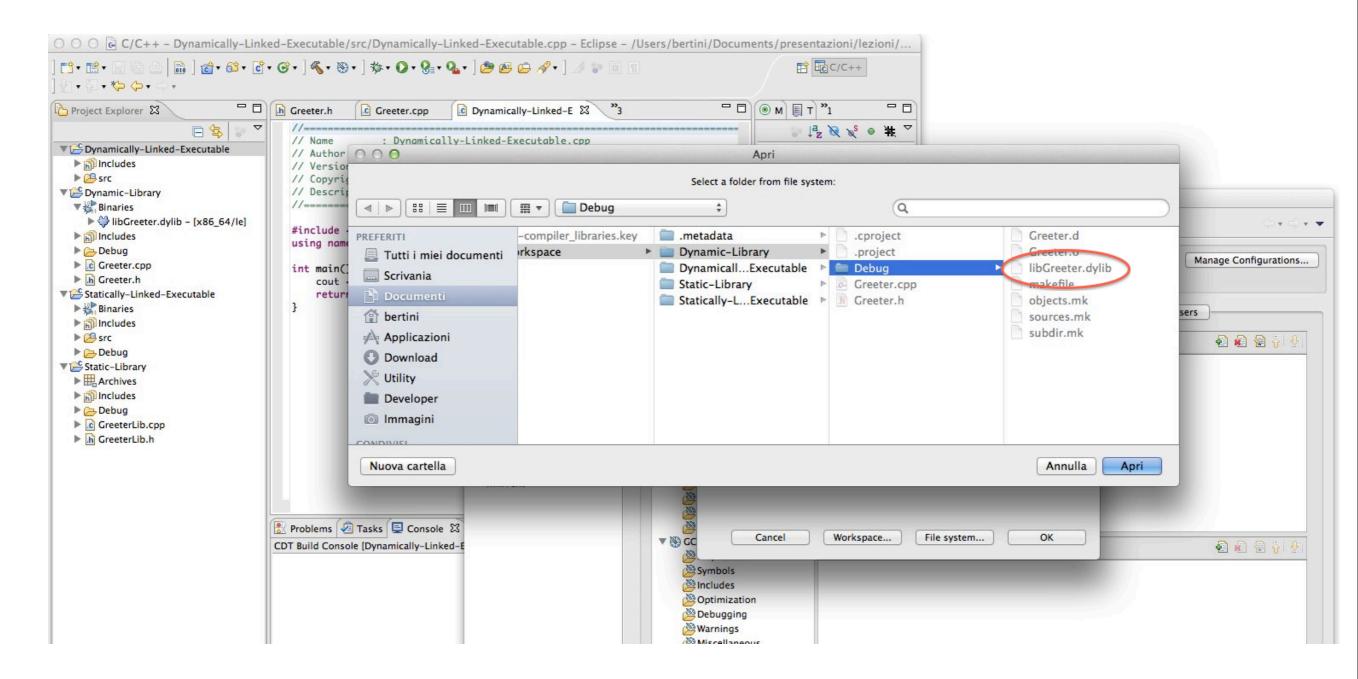
CDT Build Console [Dynamic-Library] **** Build of configuration Debug for project Dynamic-Library **** make all Building file: ../Greeter.cpp Invoking: GCC C++ Compiler g++ -00 -g3 -Wall -c -fmessage-length=0 -fPIC -MMD -MP -MF"Greeter.d" -MT"Greeter.d" -o "Greeter.o" "../Greeter.cpp" Finished building: ../Greeter.cpp Library source code is compiled, creating an object file Building target: libGreeter.dylib the -fPIC parameter is used to create Position-Independent Code. Invoking: MacOS X C++ Linker g++ -dynamiclib -o "libGreeter.dylib" ./Greeter.o It's better to use it to improve performance of the s/w using the library. Finished building target: libGreeter.dylib The linker creates a dynamic library **** Build Finished ****



- We need to tell the compiler where are the header files of the library
- We need to include the files in our client code
- We need to tell the linker where is the library file (".so" / ".dylib") and the name of the library (remind the convention used !)
- Eclipse will use this information to create the required makefile







Project Explorer 🛛 🗖 🗖	ⓑ Greeter.h ⓒ Greeter.cpp ⓒ Dynamically-Linked-E ⅔ "3	
 Project Explorer 23 Project Explorer 23 Dynamically-Linked-Executable Binaries Dobug Dynamic-Library Binaries Dobug Dobug Dobug Dobug Creeter.dylib - [x86_64/le] Dobug Creeter.h Creeter.h Creeter.h Statically-Linked-Executable Binaries Creeter.h Statically-Linked-Executable Statically-Linked-Executable Statically-Linked-Executable Statically-Linked-Executable Statically-Linked-Executable Statically-Linked-Executable Static-Library Chebug CreeterLib.cpp CreeterLib.cpp CreeterLib.h 	<pre>// Name : Dynamically-Linked-Executable.cpp // Author : // Version : // Copyright : Your copyright notice // Description : Hello World in C++, Ansi-style //</pre>	

lunedì 17 marzo 14

CDT Build Console [Dynamically-Linked-Executable]	
**** Build of configuration Debug for project Dynamically-Linked-Execut	able ****
make all	
Building file:/src/Dynamically-Linked-Executable.cpp	
Invoking: GCC C++ Compiler	
g++ -I"/Users/bertini/Documents/presentazioni/lezioni/laboratorio infor fmessage-length=0 -MMD -MP -MF"src/Dynamically-Linked-Executable.d" -MT Executable.o" "/src/Dynamically-Linked-Executable.cpp"	
Finished building:/src/Dynamically-Linked-Executable.cpp	Program source code is compiled into an object file
Building target: Dynamically-Linked-Executable	
Invoking: MacOS X C++ Linker	
g++ -L"/Users/bertini/Documents/presentazioni/lezioni/laboratorio infor	matica 2011-2012/workspace/Dynamic-Library/Debug" -o
"Dynamically-Linked-Executable" ./src/Dynamically-Linked-Executable.o	-lGreeter
Finished building target: Dynamically-Linked-Executable	
**** Build Finished ****	The library is linked by the linker
	he eventeble is
The program is linked and t	ne executable is
created	

Executing a dynamically linked program

- Remind that dynamically linked programs need to access the library (actually it is the dynamic linker that needs this)
 - Either copy the library to a path used by the dynamic linker (check info of your O.S.) or copy it in the same directory of the executable-



References and sources

These slides are based on the following articles

Suggested reading: dynamic/ shared libraries

- Learn Linux, 101: Manage shared libraries: <u>http://www.ibm.com/developerworks/linux/</u> <u>library/l-lpic1-v3-102-3/</u>
- Anatomy of Linux dynamic libraries: <u>http://www.ibm.com/developerworks/linux/</u> <u>library/l-dynamic-libraries/</u>
- Dissecting shared libraries: <u>http://www.ibm.com/developerworks/linux/</u> <u>library/l-shlibs/</u>

Suggested reading: writing dynamic/shared libraries

- Program Library HOWTO
 <u>http://www.linuxdoc.org/HOWTO/Program-Library-HOWTO/</u>
- Shared objects for the object disoriented! <u>http://www.ibm.com/developerworks/library/l-shobj/</u>
- Writing DLLs for Linux apps <u>http://www.ibm.com/developerworks/linux/library/l-dll/</u>