Le Librerie

Programmazione Avanzata e Parallela
2022/2023

Alberto Casagrande

Riusabilità del Codice

Spesso usiamo una stessa funzione in diversi progetti

Possiamo evitare di dovrela riscrivere in ogni programma?

Le **librerie** sono collezioni di funzioni che possono essere usate da programmi diversi

Possiamo usare librerie sviluppate da altri e sviluppare le nostre

Librerie Statiche e Dinamiche

Ci sono due tipi di librerie:

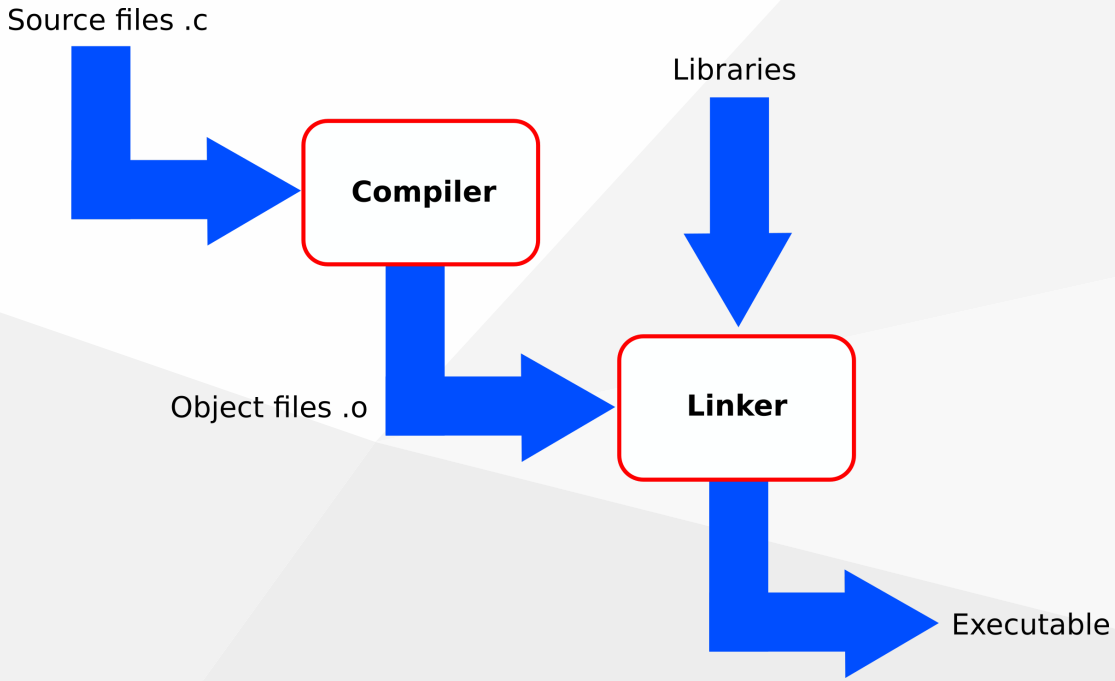
- **Librerie Statiche:** il loro codice binario viene incluso nell'eseguibile del programma.

  In GNU/Linux, la libreria `<name>` si trova nel file `lib<name>.a`
- **Librerie Dinamiche:** il loro codice binario è caricato al'esecuzione del programma da un file esterno.

  In GNU/Linux, la libreria `<name>` si trova nel file `lib<name>.so`

Linker

Sono programmi che *linkano* i file oggetto prodotti da un compilatore.



`g++` invoca automaticamente `ld`, il linker di GNU

Implementare una Libreria Dinamica

1. scrivi le funzioni in dei file sorgente e.g., `prima_lib.cpp`
2. raccogli le firme delle funzioni in un file di *header* e.g., `prima_lib.hpp`
3. compila la libreria dinamica usando le opzioni:
   - `-fPIC` rende il codice indipendente dalla posizione in memoria
   - `-shared` produce un oggetto che può essere linkato dinamicamente

```
g++ -fPIC -shared prima_lib.cpp -o libflib.so
```

Implementare una Libreria Statica

1. scrivi le funzioni in dei file sorgente e.g., `prima_lib.cpp`
2. raccogli le firme delle funzioni in un file di *header* e.g., `prima_lib.hpp`
3. compila (e non linka) i sorgenti con l'opzione g++ `-c`:

```
g++ -c prima_lib.cpp
```

Implementare una Libreria Statica (Cont'd)

4. crea il file della libreria usando il comando `ar` con le opzioni:
    - **r** rimpiazza il contenuto prececende
    - **c** crea un nuovo archivio
    - **s** costuisce un indice per l'archivio

```
ar rcs libflib_static.a prima_lib.o
```

Includere gli Header e Altre Amenità

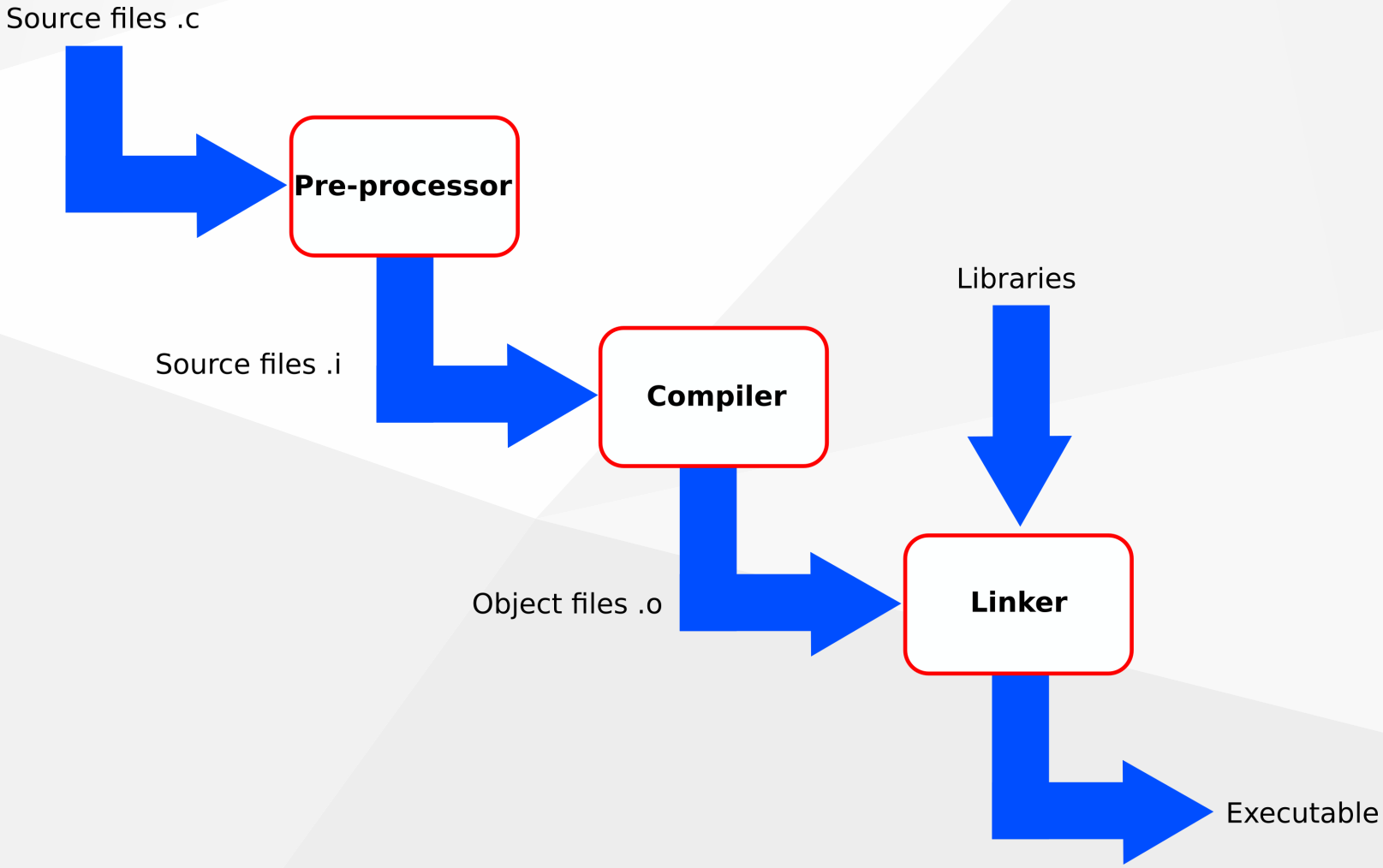
Per invocare un funzione di una libreria dobbiamo prima dichirarla

Quindi o riscriviamo la sua firma in ogni nuovo programma o...

Incorporiamo i file di *header* nei programmi

Questo può essere fatto dal **pre-processore**

Il Processo di Compilazione



Funzioni del Pre-Processore

Il pre-processore può:

- incorporare interi file in un sorgente
- definire e dimenticare delle *macro*
- valutare le *macro*

Tutte le **direttive** del pre-processore iniziano con `#`

Esempi di Direttive

```
#include <iostream>          // incorpora il file `iostream` da una
                             // directory di sistema

#include "prima_lib.hpp"     // incorpora il file `prima_lib.hpp` dalla
                             // directory corrente

#define MIN(X, Y) ((X) < (Y) ? (X) : (Y))  // definisce la macro "MIN"

#ifdef MIN    // testa se la macro "MIN" è definita e se è questo il caso
              // usa quanto segue fino a #endif

#undef MIN    // dimentica la macro "MIN"

#endif        // termina il blocco iniziato da #ifdef
```

Linkare una Libreria Dinamica

1. include l'header della libreria nel sorgente
2. compila il programma con le opzioni:
   - `-L<lib_path>` se la libreria non è una directory standard al momento della compilazione
   - `-Wl,-rpath=<lib_path>` se la libreria non sarà in una directory standard al momento dell'esecuzione
   - `-l<name>` per linkare la libreria nel file `lib<name>.so`

Linkare una Libreria Dinamica: Esempio

```
g++ -L. -Wl,-rpath=../pluto/ programa.cpp -lflib -o programma
```

- `-lflib` linka il programma alla libreria nel file `libflib.so`
- `-L.` cerca le librerie anche in `.` (la directory corrente)
- `-Wl,-rpath=../pluto/` al momento dell'esecuzione del programma cerca i file delle librerie anche nella directory `../pluto/`

Linkare una Libreria Statica

1. include l'header della libreria nel sorgente
2. compila il programma con le opzioni
   - `-L<lib_path>` se la libreria non è una directory standard al momento della compilazione
   - `-l<name>` per linkare la libreria nel file `lib<name>.a`

```
g++ -L. programa.cpp -lflib_static -o programma
```