Librerie

Programmazione di Sistema

A.A. 2017-18

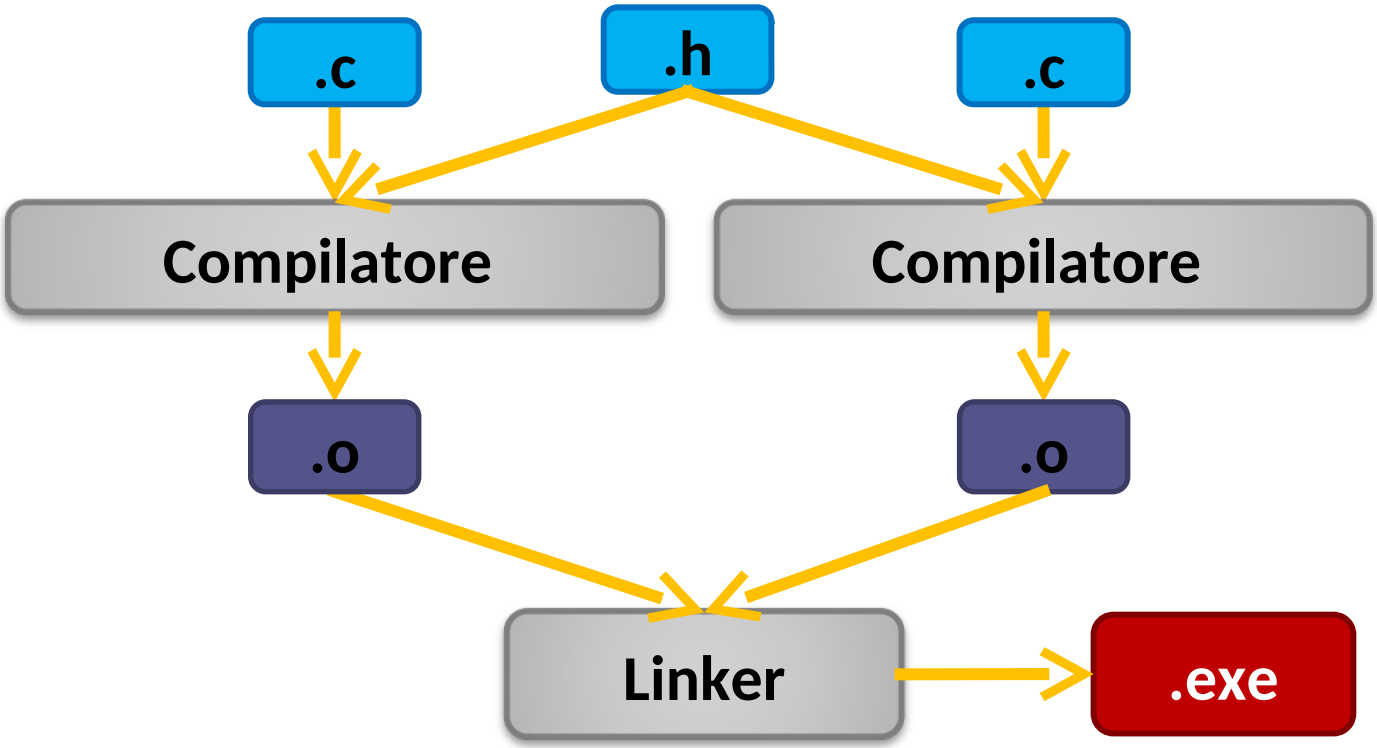
Argomenti

- Uso di librerie
- Librerie statiche
- Librerie dinamiche o condivise

Il processo di compilazione

- Il programma è suddiviso in un insieme di file sorgenti
  - Ciascuno è compilato separatamente
  - I file header permettono di dichiarare simboli definiti esternamente
- Il compilatore produce moduli oggetto
  - Contengono codice macchina con riferimenti pendenti alle funzioni/variabili esterne
- Il linker unisce più moduli oggetto in un eseguibile finale
  - Risolvendo i riferimenti ai simboli esterni ed inserendo gli opportuni indirizzi ad essi

Il processo di collegamento



Librerie

- Insieme di moduli oggetto archiviati in un unico file
  - Facilitano la gestione modulare dei programmi
  - Riducono i tempi di compilazione
  - Favoriscono l'incapsulamento
- Permettono di radunare funzioni e strutture dati pertinenti ad uno stesso dominio
  - Funzioni matematiche
  - Manipolazione di stringhe
  - Gestione della concorrenza

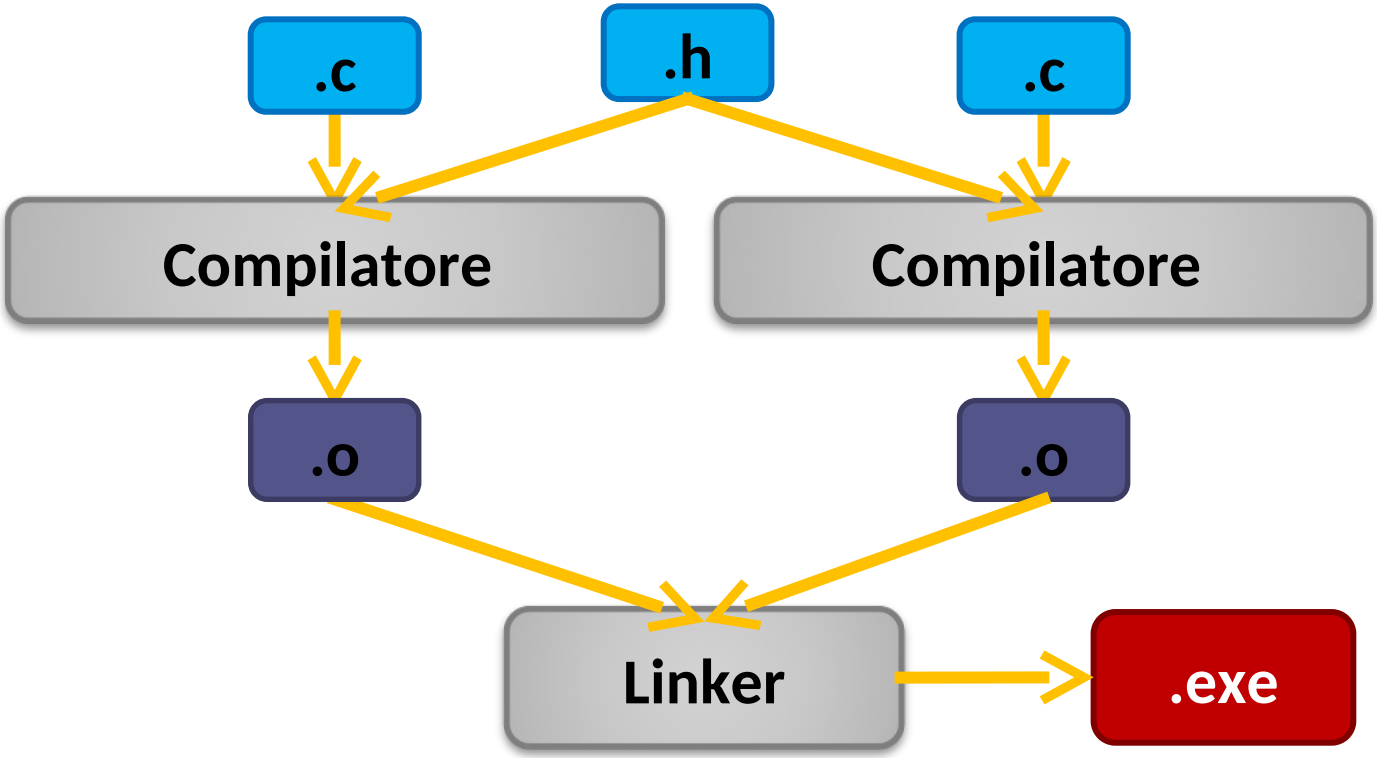
Contenuto di una libreria

- Funzioni e variabili esportate
  - Accessibili ai programmi che la utilizzano
- Funzioni e variabili private
  - Accessibili solo alle altre funzioni della stessa libreria
- Costanti ed altre risorse
- Le funzioni e le variabili esportate sono dichiarate in un file header
  - Viene incluso dai moduli sorgente che le utilizzano
  - I simboli sono preceduti dalla parola chiave «extern»

Caricamento delle librerie

- L'uso di una libreria richiede due fasi
  - Identificazione dei moduli necessari e loro caricamento in memoria
  - Aggiornamento degli indirizzi per puntare correttamente ai moduli caricati
- Le due operazioni possono essere fatte in fasi differenti
  - Durante il collegamento (linker)
  - Durante il caricamento del programma (loader)
  - Durante l’esecuzione (programma stesso)

Il processo di collegamento



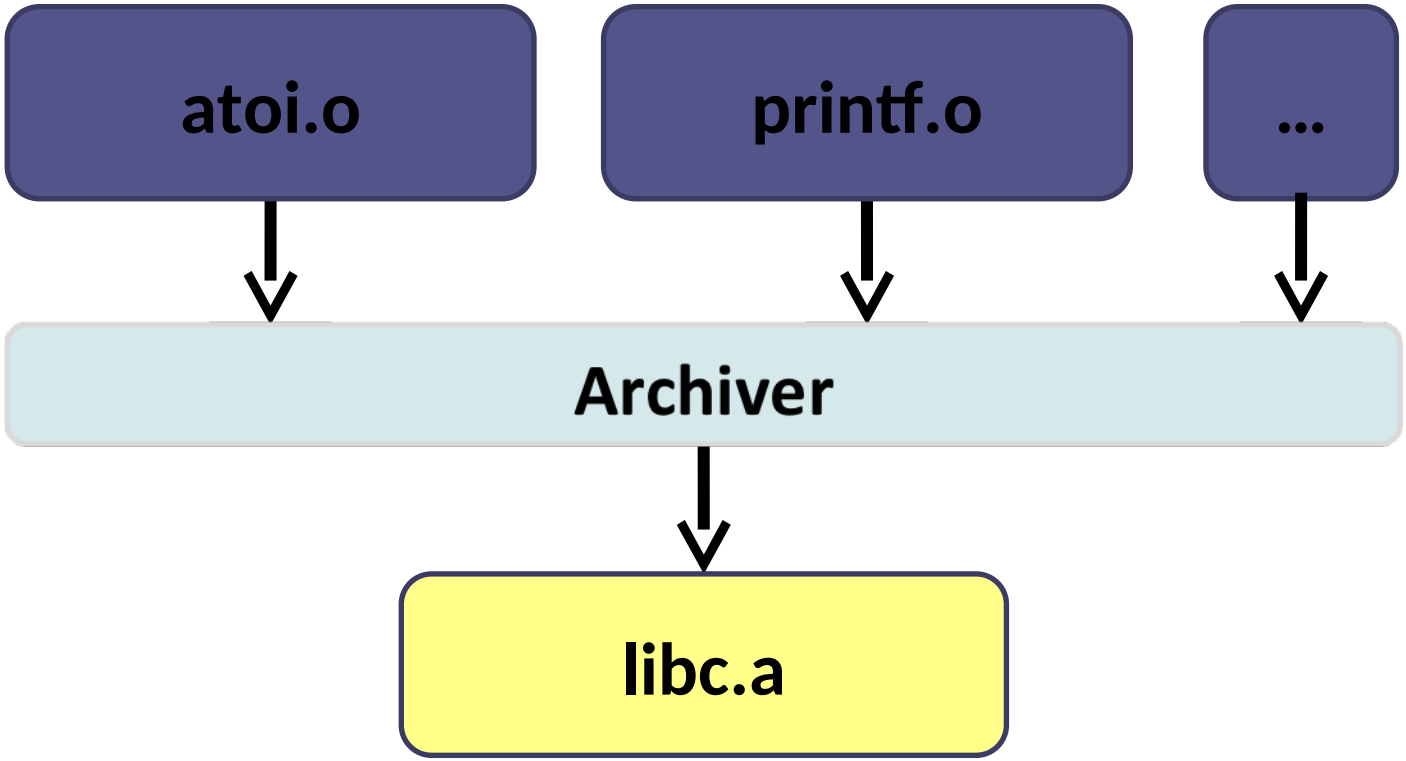
Tassonomia delle librerie

- Librerie statiche
- Librerie dinamiche o condivise
  - Collegate dinamicamente
  - Caricate dinamicamente

Librerie a collegamento statico

- Contengono funzionalità collegate staticamente al codice binario cliente in fase di compilazione
- Una libreria statica è un file archivio
  - Contiene un insieme di file object, creati dal compilatore a partire dal codice sorgente
  - L'impacchettamento viene effettuato dall'archiver

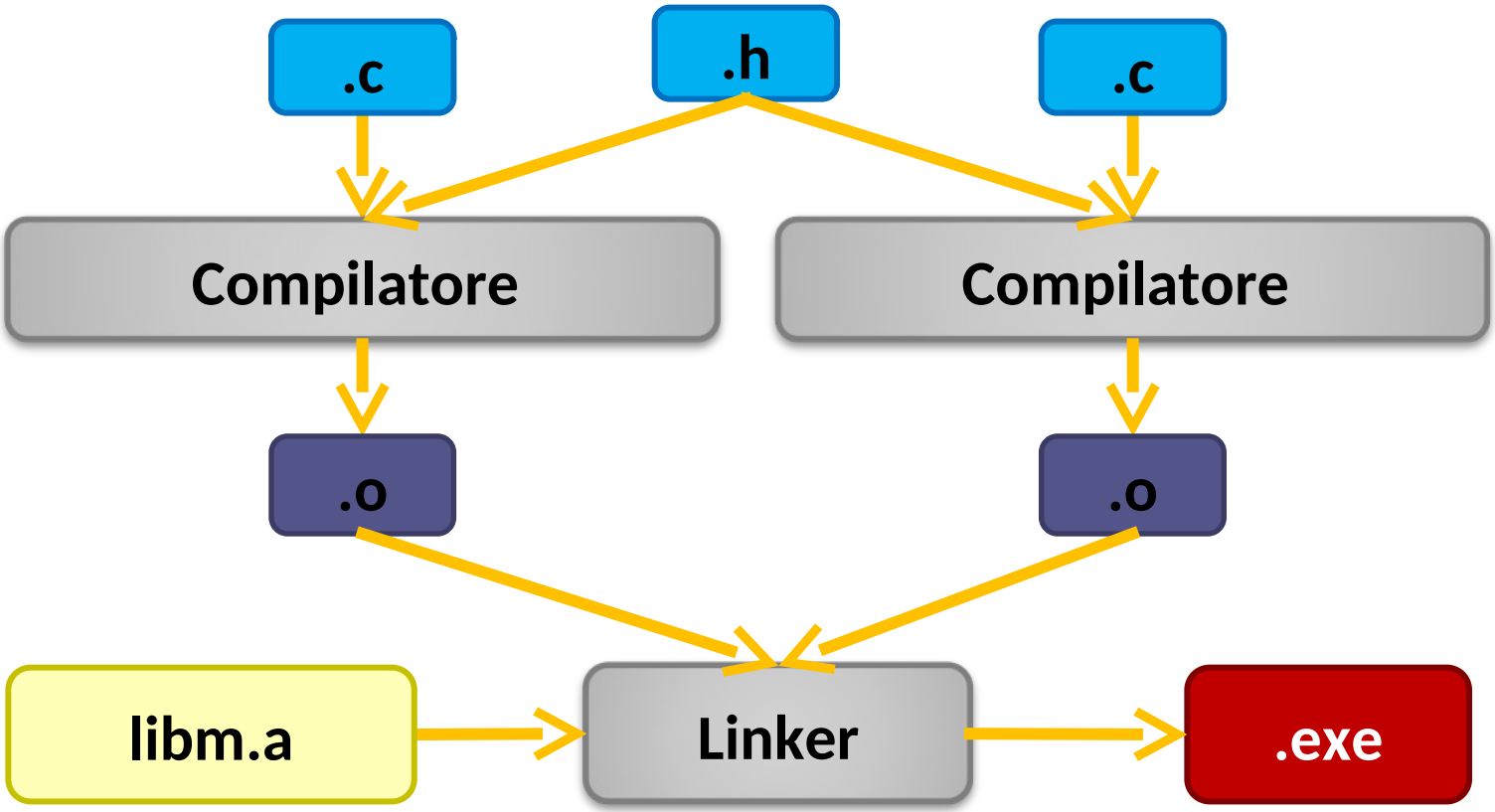
Archiver



Collegamento di una libreria statica

- Il linker identifica in quali moduli della libreria si trovano le funzioni di richiamate nel programma.
- Carica nell'eseguibile solo quelli necessari
- Terminata la fase di collegamento, i moduli ed i simboli della libreria statica risultano annegati nel codice binario originario

Collegamento statico



Collegamento statico

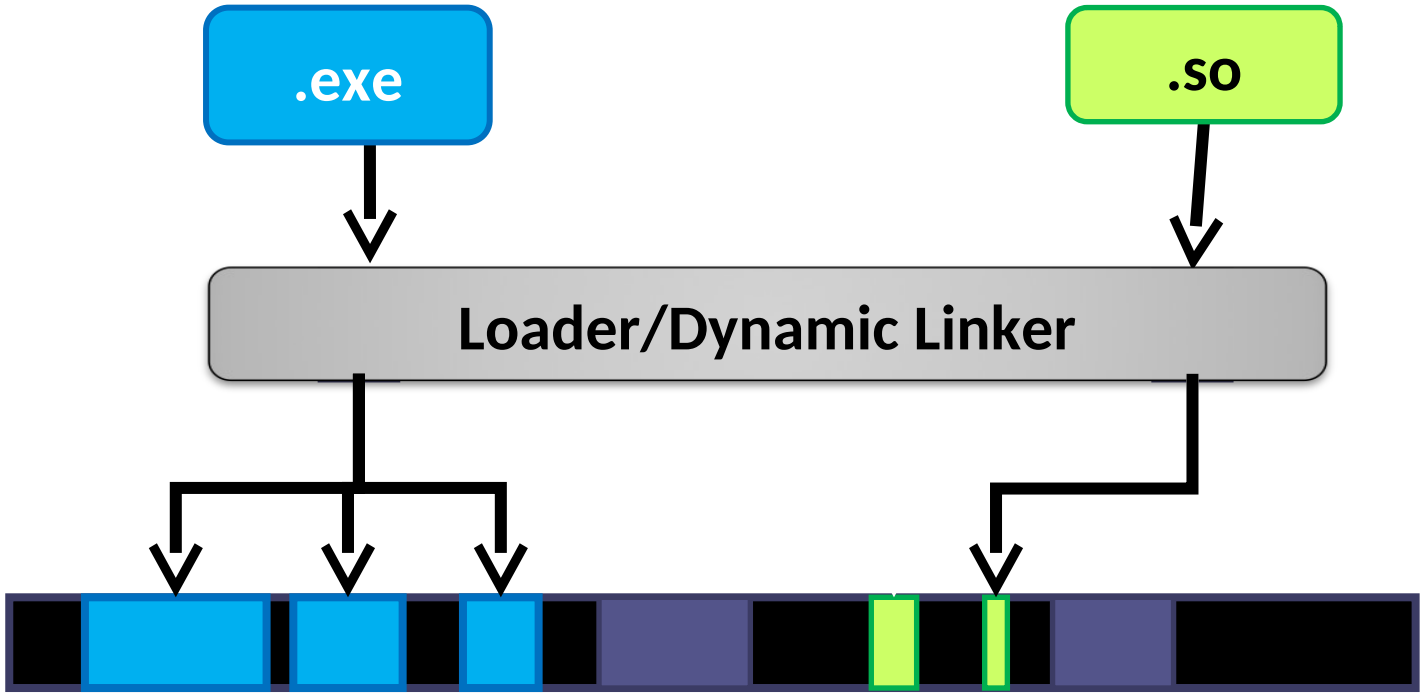
- Vantaggi
  - Il codice delle librerie è certamente presente nell'eseguibile.
  - Non ci sono dubbi sulla versione della libreria adottata
- Svantaggi
  - Gli stessi contenuti sono presenti in processi differenti
    - Il sistema non se può accorgere e utilizza pagine fisiche distinte
    - Bassa efficienza
  - L'uso delle librerie statiche comporta una riduzione della modularità del codice
    - Ogni applicazione che ne fa uso deve essere ricompilata ad ogni modifica

Librerie statiche

- Linux
  - In Linux, GCC offre come archiver il tool ar
  - Per convenzione le librerie statiche cominciano con il prefisso lib e hanno un'estensione .a
- Windows
  - Hanno un estensione .lib
  - Possono essere create a partire dai moduli oggetto con il programma "lib"
    - Visual Studio offre un procedimento guidato per la loro creazione

Librerie a collegamento dinamico

- Il file eseguibile non contiene i moduli della libreria
  - Vengono caricati successivamente, nello spazio di indirizzamento del processo



Collegamento dinamico

- All'atto della creazione del processo, il loader mappa nello spazio di indirizzamento tutte le librerie condivise
  - Risolvendo i simboli corrispondenti
- In Linux il dynamic linker è il programma ld.so
  - Anch'esso è una libreria ELF condivisa, priva di ulteriori riferimenti a librerie dinamiche
  - All'avvio di un applicativo, viene mappato in memoria

Dynamic Linker

- Mappa le librerie nello spazio di memoria del processo
- Aggiorna la tabella dei simboli (variabili e funzioni) per ogni libreria caricata

Collegamento dinamico – Windows

- Il dynamic linker in Windows fa parte del kernel stesso
  - Funzionamento simile al caricamento dinamico di un ELF in Linux

Caricamento dinamico

- È possibile controllare esplicitamente il caricamento delle librerie condivise
  - L'applicazione può avviarsi in assenza di qualsiasi libreria e farne richiesta quando necessario

Caricamento dinamico – Linux

- Linux espone le Dynamic Loading API
  - dlopen()
    - rende un file object accessibile al programma
  - dlsym()
    - recupera l'indirizzo di un simbolo (variabile o funzione) di un file object aperto
  - dlerror()
    - Ritorna l'ultimo errore occorso
  - dlclose()
    - Chiude il file object aperto

Esempio

```
#include <stdio.h>
#include <dlfcn.h>

void invoke ( char *lib, char *m, float
arg) {
  void* dl_handle = dlopen( lib,
RTLD_LAZY );
  if (!dl_handle)   return;
  float (*func)(float) = dlsym( dl_handle,
m );
  if (func==NULL) return;
  printf("Result:  %f\n", (*func)(arg) );
  dlclose( dl_handle );
}

int main( int argc, char *argv[] ){
    invoke ("libm.so", "cosf", 3.14156f);
}
```

Librerie a caricamento dinamico - Windows

- Moduli binari in formato Portable Executable con estensione ".dll"
  - Possono contenere funzioni, variabili globali, costanti, risorse
  - Una funzione di ingresso (DllMain)
- La funzione LoadLibrary(…) carica la libreria indicata e la mappa nello spazio di indirizzamento del processo
  - Restituisce la handle del modulo caricato

Funzione di ingresso - Windows

- La DLL può specificare un punto di ingresso opzionale
  - Chiamato quando un processo o un thread mappano e/o rilasciano la DLL nel proprio spazio di indirizzamento
- DllMain(HINSTANCE h, DWORD r, PVOID unused)
  - h, handle, è l'indirizzo a partire dal quale è stata mappata la DLL
  - r, indica il tipo di evento (DLL_PROCESS_ATTACH, DLL_PROCESS_DETEACH,…)

Condivisione dati DLL

- Normalmente, le variabili di una stessa DLL non sono condivise tra processi diversi
  - Ogni processo ne contiene una copia indipendente
  - Codice e risorse sono condivisi in sola lettura
- Le variabili globali esportate da una DLL sono memorizzate all'interno processo che la utilizza
  - Se più processi utilizzano la stessa DLL, esistono altrettante copie delle variabili globali
- È possibile creare zone di memoria condivise tra tutti i processi che usano una data DLL
  - Collocando le variabili globali in un apposito segmento della DLL

Esportare simboli DLL

- Per poter esportare ed importare funzioni e strutture dati
  - Usare direttive chiave (dllexport, dllimport)
  - Creare un file module definition (.def)

Direttive chiave

- __declspec(dllimport)
  - Per importare i simboli pubblici della DLL
- __declspec(dllexport)
  - Per esportare i simboli pubblici dalla DLL e renderli accessibili
- Dllexport/Dllimport
  - Tipicamente si usano all'interno di blocchi specifici
    - All'interno di un define statement per gli export
    - All'interno di un ifdef statement per gli import

Esempio

```
// SampleDLL.h
#ifdef EXPORTING_DLL
extern __declspec(dllexport) void
HelloWorld() ;
#else
extern __declspec(dllimport) void
HelloWorld() ;
#endif
```

```
// SampleDLL.c
#define EXPORTING_DLL
#include "sampleDLL.h"

BOOL APIENTRY DllMain(…)

void HelloWorld() {printf("Hello world");}
```


Esempio

```
// Static_Dll_usage.c
#include "sampleDLL.h"
void someMethod() {
      HelloWorld();
}
```

```
// Dynamic_Dll_usage.c
HINSTANCE hDLL
=LoadLibrary(L"sampleDLL.dll");
if (hDLL != NULL)
{
      DLLPROC Hw = (DLLPROC)

GetProcAddress(hDLL,L"HelloWorld");
      if (Hw != NULL) (*Hw)();
      FreeLibrary(hDLL);
}
```


Direttive chiave

- In fase di compilazione il compilatore può modificare il nome della funzione esportata per tenere conto del numero e tipo dei suoi parametri (Name mangling)
  - Usare direttiva extern per impedire tale comportamento

Module definition

- È possibile usare un file .def in cui dichiarare le funzioni DLL da esportare

```
// SampleDLL.def
//
LIBRARY "sampleDLL"

EXPORTS
      HelloWorld
```


Compilazione di DLL

- Visual Studio genera, oltre al .dll, anche un file .lib
  - Contiene uno stub delle API offerte
  - Deve essere linkato dai progetti che intendono utilizzare in modo statico la DLL

Uso dll – caso statico

- Il sorgente del programma che usa la DLL include il file di intestazione.
  - Si utilizzano i simboli esportati dalla DLL
- Il programma viene collegato staticamente con il file .lib associato
  - Esso carica e rilascia il .dll
  - Mappa automaticamente l'accesso a funzioni e variabili

Uso dll – caso dinamico

- Il programma principale non fa riferimenti diretti ai simboli definiti dalla DLL
- La DLL viene caricata esplicitamente tramite LoadLibrary()
- Si accede alle funzioni tramite GetProcAddress(…)
- Si rilascia la libreria tramite FreeLibrary() o tramite FreeLibraryAndExitThread()

Spunti di riflessione

- Si realizzi un programma che carica dinamicamente una libreria e ne invoca un metodo a scelta