

MS Visual Studio: Linea di Comando

- Aprire una shell di DOS tramite:
 - Start → MS VisualStudio 2008 → Visual Studio Tools
 - è una normale shell di DOS con aggiunte al path ed alle variabili d'ambiente
 - Digitare `cl /?` per visualizzare i vari comandi

1

Compilazione

- Digitare il nome del compilatore (**cl**) seguito dallo switch **/c** e da tutti i file .c da compilare
- Lo switch indica al compilatore... di compilare → vengono **creati i file oggetto**

ad esempio: `cl /c sourceFile.c`

2

Fase di Linking

- Digitare il nome del linker (**link**) seguito dal nome di tutti i file oggetto (.obj) da collegare
- Il linker genera un eseguibile avente nome uguale al file .obj contenente main() ed estensione .exe
- Per dare un nome diverso all'eseguibile utilizzare lo switch **/out:nomeDelFile**

ES: `link sourceFile.obj /out:eseguibile.exe`

3

Hello World! - Compilazione

- Creare un file di testo vuoto di nome Hello.c (ad es. usando Notepad)
- Digitare il programma:

```
#include <stdio.h>
int main()
{
    printf("Hello World!");
}
```

- Aprire la **shell** di VS
- Digitare: `cl /c hello.c`
- Compilatore genera un file **hello.obj**

Il file generato non è ancora eseguibile poiché ***non è ancora stato collegato con le librerie di sistema***

- Invocare il **linker**. Digitando che cosa?

4

Header Files

- Gli header file contengono le **dichiarazioni di variabili e funzioni** di un modulo
 - Le **definizioni** sono contenute “da qualche altra parte”
Tipicamente si lavora con coppie (NomeModulo.h, NomeModulo.c) dove il file .c contiene le definizioni di ciò che è stato dichiarato nel .h
 - Nelle situazioni semplici in cui tutti i file .h e .c si trovano nello stesso direttorio, il compilatore (meglio, il preprocessore) è in grado di trovare da solo tutti i file .h utili
- In che modo?

5

Header Files & Command Line

- In qualche caso è necessario **distribuire librerie e “nascondere” il codice** (vedi distribuzione Linux): che cosa si distribuisce?
- ...basta distribuire il solo **risultato della compilazione (.obj)** più i **file header (.h)** necessari per compilare altro codice che si basa sui costrutti contenuti nella “libreria” distribuita...
- ...non è indispensabile distribuire il codice sorgente – contenuto nei file .c

6

Solo Compilazione

Si supponga che il modulo sia composto da un file header (myMod.h) e un file sorgente (myMod.c)

- **`cl /c myMod.c`**

Si ottiene un file myMod.obj → file oggetto, compilato ma non eseguibile (è un modulo e non ha un entry point, quindi non ha main)

- **è possibile distribuire il modulo dando solo myMod.h (con le dichiarazioni) e myMod.obj (con il codice compilato)**

- **Chi vuole utilizzare il modulo deve:**

- Fare riferimento a myMod.h per le dichiarazioni
- Utilizzare myMod.obj in fase di link

7

Un passo avanti

- **In realtà il compilatore `cl` è in grado di invocare automaticamente anche il linker...**

- **basta evitare di specificare lo switch `/c` che impedisce il link automatico!**

- **`cl listaFileSorgenti.c`**

- Un file oggetto per ogni file sorgente (come prima)
- Un file eseguibile avente nome uguale al nome del file sorgente contenente entry point (funzione main)

- **Oppure: `cl listaFileSorgenti.c /o run.exe`**

- Idem come sopra + nome eseguibile come specificato – invoca lo switch **`/out:nomeEseguibile`** del linker

8

Utilizzo di moduli compilati

■ `cl myProg.c c:/moduli/obj/myMod.obj
/I c:/moduli/header /o eseg.exe`

Si compila il file `myProg.c`...

- che fa riferimento all'header `myMod.h` che viene trovato nel direttorio degli header file specificato
- si utilizza `myMod.obj` per il link
- e si ottiene come risultato `eseg.exe`