Linea di comando

- Aprire una shell di DOS tramite:
  - Start → MS VisualStudio 2005 → Visual Studio Tools
    - E' una normale shell di DOS con aggiunte al path ed alle variabili d'ambiente
  - Digitare **`cl /?`** per vedere i 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
- Es: cl /c sourceFile.c

2

Link

- 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 il main ed estensione – ovviamente – .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>
main()
{
    printf("Hello World!");
}
```

- Aprire la shell di VS
- Digitare: **`cl /c hello.c`**
- Se non ci sono errori di sintassi, il compilatore genera un file hello.obj
- Il file generato non è ancora eseguibile poiché non è ancora stato collegato con le librerie di sistema

4

Hello World! - Link

- Per generare l'eseguibile, invocare il linker passandogli tutti i file oggetto da collegare (uno)

- link hello.obj

Oppure

- link hello.obj /out:helloworld.exe

5

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

6

Header Files & Command Line

- Nelle situazioni semplici in cui tutti i file .h e .c si trovano nello stesso direttorio, il compilatore (il preprocessore) è in grado di trovare da solo tutti i file .h che gli servono

  → In che modo?

- E se i file sono sparsi in diversi direttori?

7

Header Files & Command Line

- In qualche caso è necessario distribuire librerie e "nascondere" il codice: 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 necessario (ed è meglio!) distribuire il codice sorgente – contenuto nei file .c

8

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 il main)

9

Distribuzione di moduli compilati

- Ora è possibile distribuire il modulo dando solamente i file 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

10

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!
- E' possibile, quindi, compilare e linkare in un unico colpo!

11

Compile & Link

- cl listaFileSorgenti.c
  - Un file oggetto per ogni file sorgente (come prima
  - Un file eseguibile avente nome uguale al nome del file sorgente contenente l'entry point
- Oppure: cl listaFileSorgenti.c /o run.exe
  - Idem come sopra + nome eseguibile come specificato – invoca lo switch /out:nomeEseguibile del linker

12

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

13