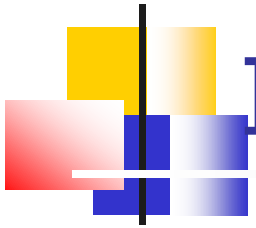
Introduzione al debug di un programma C

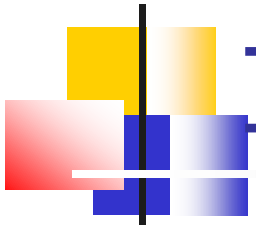
Gianpiero Cabodi e Paolo Pasini

Dip. Automatica e Informatica
Politecnico di Torino



Il Debug di un programma

- Il debug è la fase in cui si lavora per localizzare e rimuovere gli errori (BUG) di un programma
- Attenzione: errori riscontrabili in esecuzione!
  - Gli errori di compilazione si assumono già rimossi.
- Strategie più comuni:
  - Il programma fa molto output (per tracciarne l'esecuzione)
  - Utilizzo di programma apposito (Debugger)



Il Debugger

Si tratta di un programma che permette un'esecuzione "speciale" del programma, che consente di:

- Visualizzare il flusso di esecuzione del programma (sul sorgente C)
- Arrestare l'esecuzione in punti pre-stabiliti (break-points)
- Eseguire il programma passo-passo
- Visualizzare (e modificare) il contenuto delle variabili
- MOLTO ALTRO …

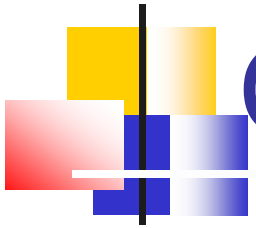
Il debugger GDB (GNU debugger)

Debugger free dell'ambiente GNU

- utilizzato nei sistemi Unix-like (es. Linux)
- disponibile anche in ambiente Windows

E' disponibile sia mediante comandi in linea, che mediante interfacce e ambienti grafici:

- DDD
- KDevelop
- Eclipse
- CodeBlocks
- Emacs

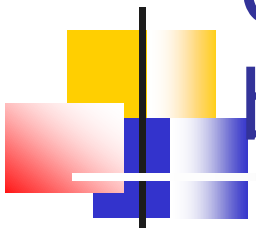


Comandi principali: eseguire

Tra parentesi l'abbreviazione minima utilizzabile

- run (r): avvia l'esecuzione di un programma dall'inizio
- continue (c): conrinual un'esecuzione precedentemente interrotta
- next (n): esegue un'instruzione
- step (s): esegue un'istruzione, entrando nelle funzioni chiamate

Comandi principali: breakpoints e watchpoints



- break (b) `42`: mette un breakpoint alla linea `42` del programma
- break `ordina`: mette un breakpoint alla chiamata della funzione `ordina`
- watch (wa) `somma`: mette un breakpoint alla modifica della variabile `somma`

I breakpoint sono numerati. Un breakpoint può successivamente essere candellato (delete) disabilitato (disable) o abilitato (enable)

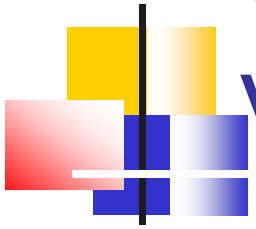
Un breakpoint può essere condizionato. Es.

```
cond 2 (i>=20 && j<0)
```

Comandi principali: visualizzare/modificare

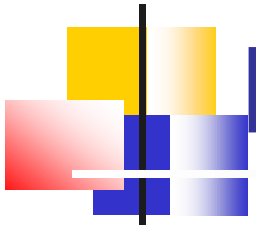
- print (p) `<variabile o espressione>`: visualizza un'espressione C (spesso nome di una variabile). Es.
  - print `x`
  - print p `studente`
  - print p `studente.nome`
- set `<variabile> = <valore>`: modifica contenuto di una variabile. Es.
  - set `x = 25`

Comandi principali: visualizzare stack (funzioni)



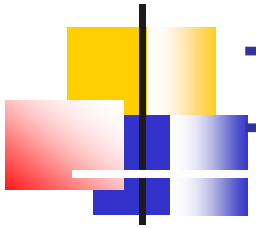
- up/down: permettono di risalire una catena di chiamate (annidate) a funzioni. Utilissimi nel caso di chiamate ricorsive.
- call <`funzione`>(<`parametri`>):

  permette la chiamate "volante" di una funzione.



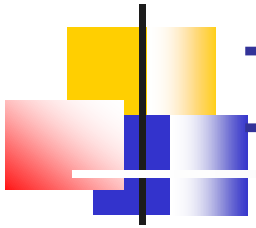
Debug in CodeBlocks

- Mediante l'interfaccia grafica codeblocks è possibile attivare (quasi) tutte le funzioni illustrate, sia mediante menu che tasti (f5, f7, f8, …)
- Per visualizzare variabili è opportuno utilizzare la finestra "watches"
- Se è installato GDB come debugger, è possibile utilizzare comandi diretti a gdb nella finestra "logs" (abilitata mediante view->logs)



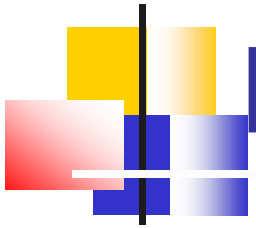
In pratica: come procedere

- Conviene sempre inserire output aggiuntivo (printf) con puro scopo di debug
- Le prime esecuzioni andrebbero effettuate passo-passo
  - esaminando i contenuti delle variabili principali (controllando che i valori siano corretti)
  - Esaminando il flusso (l'esecuzione "passa" nelle istruzioni previste (if, for, while, chiamate a funzioni, …



In pratica: come procedere

- Nel caso di programmi lunghi o esecuzioni iterative (o ricorsive) lunghe, utilizzare i breakpoint per fermare l'esecuzione nelle parti da esplorare.
- Nel caso di errori, sfruttare la possibilità di
  - Esaminare variabili o valutare espressioni
  - Nel caso in cui si voglia evitare ri-eseguire il programma dall'inizio (ma valutare eventuali correzioni "al volo"
    - Modificare il contenuto di variabili
    - Chiamare (eventualmente) funzioni



L'istruzione assert

Può essere estremamante utile inserire in un programma l'istruzione:

assert (<condizione-logica>);

Es.

assert (p->next != NULL);

assert (i>=0 && i<MAX);

Se l'assert fallisce (l'espressione è falsa) il programma termina l'esecuzione.

VANTAGGIO: i problemi vengono visualizzati molto presto.

L'effetto è simile a:

if (<condizione>) {printf("errore\n");}