

# **GDB**

## **The GNU Debugger**

# Il problema del programmatore

- Il programmatore perfetto non esiste!
- Nessuna sicurezza che il programma funzioni “al primo colpo”
- Più il programma è grande più ci possono essere errori (bug)
- La maggior parte dei problemi si riscontrano in fase di testing
- (spesso proprio durante una demo!)

# Tipi di problemi

- Il programma non viene eseguito come ci si aspetta
  - Risultati indesiderati
  - Comportamenti non previsti
- Il programma si blocca, andando in crash
  - Accesso a zone di memoria non autorizzate
  - Cattivo utilizzo delle risorse
  - ...

**Necessario un metodo efficiente per individuare gli errori**

# Possibili soluzioni

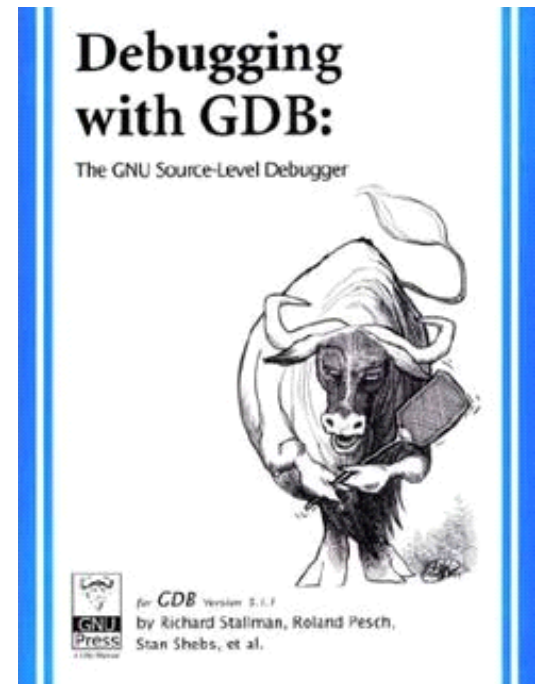
- Lavorare “sulla carta”
  - Trace table, flow chart, etc.
  - Molto lento
  - Impraticabile con programmi enormi
  - Non sempre si individuano gli errori
- Stampa di info aggiuntive nel programma
  - Poco elegante e scomodo
  - Non sempre efficace

# Scelta migliore: Debugger

- Programma usato per analizzare il comportamento di un altro programma
- “*De*” in senso privativo, “*Bug*” = errore
- Con un debugger si può:
  - Eseguire il programma un’istruzione alla volta
  - Fermare l’esecuzione in un dato punto
  - Ispezionare il contenuto di variabili
  - Sapere a che punto è l’esecuzione

# GDB: Debugger per C e C++

- Gnu DeBugger
- Sviluppato dallo stesso Richard Stallman (autore di gcc)
- Interfaccia a linea di comando



# Compilazione

- C'è bisogno di avere più informazioni per il debug sul file eseguibile
- Tipicamente tali informazioni vengono inserite dal compilatore
- Con gcc:

```
gcc -g nomeFile.c
```

```
gcc -g nomeFile.c -o nomeEseguibile
```

# Avvio GDB

***gdb nomeEseguibile***

```
labos:~/labos$ gcc -g -o palindroma palindroma.c
labos:~/labos$ gdb palindroma
GNU gdb 6.4.90-debian
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...Using host libthread_db
library "/lib/tls/i686/cmov/libthread_db.so.1".

(gdb) _
```

- **quit** uscire da GDB

# Prompt di GDB

- Interfaccia simile a Bash
- I tasti  *cursore* sfogliano la history dei comandi
- Il tasto  *tab* completa il comando inserito
- Il tasto invio senza nulla esegue **l'ultimo comando inserito**
- **help** mostra la lista e l'help dei comandi

# Comandi base

Nome comando	Breve descrizione
<b>run, r</b>	Esegue il programma caricato
<b>print, p</b>	Stampa il valore di un'espressione (una var.)
<b>where</b>	Mostra il backtrace dello stack (punto di esecuzione)
<b>break, b</b>	Imposta un break-point
<b>list, l</b>	Mostra il codice numerando le righe
<b>next, n</b>	Avanza alla successiva linea di codice
<b>step, s</b>	Come <b>next</b> , ma entra nelle chiamate a funzione
<b>display</b>	Simile a <b>print</b> , ma stampa ad ogni passo
<b>watch</b>	Imposta un watch-point
<b>continue, c</b>	Continua l'esecuzione

# Comando run

- Fa partire l'esecuzione del programma
- Possibile aggiunta degli argomenti da passare al programma
  - Es. run pippo 3
- L'esecuzione continua fino ad un **punto di break** (vedere dopo)

# Comando print

- **print (p)**: stampa il contenuto di una variabile
  - **print** *variabile*
  - es. **print** *a*
- Si può anche stampare il risultato di un'espressione
  - **print** *espressione*
  - es. **print** *a+b*

# Comando where

- Mostra lo stack corrente delle chiamate a funzioni
- Molto utile per sapere in che punto dell'esecuzione ci si trova
  - può non bastare il numero di riga!
- Esempio:
  - Ci si trova nella funzione potenza (riga 6), invocata dal main (riga 12)

```
(gdb) where
#0 potenza (base=2, esp=3) at potenza.c:6
#1 0x080483b6 in main () at potenza.c:12
```

# Punto di break

- Punto in cui si vuol sospendere l'esecuzione per analizzarla
- Esistono due tipi fondamentali:
  - **Normal Break-Point** (o solo **Break-Point**)
    - Associati ad una riga del codice
  - **Watch-Point**
    - Associati al cambiamento del valore di una variabile
- Gdb tiene traccia della lista di tutti i point
- I point possono essere visualizzati o cancellati
- Ad ogni point si associa un numero progressivo

# Normal Break-Point

- Creazione col comando break (b):
  - In base al numero di riga  
*break numeroRiga*
    - Comando *list* per visualizzare le righe numerate
  - In base al nome di una funzione  
*break nomeFunzione*
- Visualizzazione dei break-points con info:  
*info break* (o *info watch*)
- Cancellazione di un break-point con delete (d):  
*delete numeroPoint*

# Comandi next e step

- **next (n)** e **step (s)** eseguono l'istruzione corrente e mostrano quella successiva
- Se l'istruzione è una chiamata a funzione:
  - **next** esegue il programma fino all'uscita dalla chiamata
    - (non entra nelle sotto-funzioni)
  - **step** si ferma alla prima istruzione della chiamata
    - (entra nelle sotto-funzioni)
- hanno senso solo se il programma è in esecuzione (e sospeso)

# Comando display

- **display**: stampa il risultato di un'espressione al termine di ogni passo
- Gdb tiene traccia di tutti i display
  - **info display** ne mostra la lista (con il numero associato)
  - **delete display *numero*** cancella il display indicato da numero

# Watch-Point

- Watch-Point: L'esecuzione viene sospesa quando una delle variabili in "watching" cambia di valore
  - Viene mostrato il contenuto "prima" e "dopo"
- Creazione col comando watch:
  - Associato alla variabile (o espressione) che si vuole analizzare  
*watch nomeVariabile*
- Visualizzazione dei watch-points con info:  
*info watch* (o *info break*)
- Cancellazione di un watch-point con delete (d):  
*delete numeroPoint*

# Comando continue

- **continue (c)** fa riprendere l'esecuzione fino al prossimo breakpoint
  - Utile quando si è interessati solo a punti particolari dell'esecuzione

# Ddd: un front-end grafico

- **ddd**: Data Display Debugger
- Interfaccia grafica per gdb
- Necessario installarlo prima!

