Laboratorio 4:

Debugging

- Il debugger è un programma che ci consente di analizzare il comportamento di altri programmi dandoci la possibilità di conoscere lo stato interno delle variabili, effettuare dei breakpoint, dei watch ed altre operazioni che ci aiuteranno a identificare comportamenti anomali e non desiderati del programma.
- GDB è l'acronimo per **G**NU **D**e**B**ugger, ovvero il debugger ufficiale del progetto GNU.
- Supporta una buona quantità di linguaggi tra cui C, Java, C++ e tanti altri

Debugging

- GDB è un debugger completamente testuale a linea di comando. È il debugger presente su qualsiasi distribuzione Unix/GNU Linux ed è disponibile anche per Windows.
- Esistono inoltre numerosi debugger grafici e IDE di sviluppo che sfruttano GDB.

- Oggi lavoriamo in Linux

Preparazione del debug

- Prima di fare il debug dobbiamo rendere il nostro codice appetibile al debugger. Questa fase varia da linguaggio a linguaggio. Con C il procedimento è semplicissimo e non si discosta dal normale procedimento che usiamo per compilare i sorgenti.
- Ad esempio.per preparare il programma helloworld al debugging, in una finestra di shell digitare:

  gcc helloworld.c -o helloworld.exe –g

  Fatta eccezione per il flag -g il comando sopra è lo stesso che avremmo usato per compilare normalmente il sorgente *helloworld.c*.

Avviare il debugger

- Superata la fase preliminare non ci resta che avviare il nostro debugger e dargli in pasto il nostro eseguibile da verificare. Per far ciò dobbiamo lanciare gdb dando come parametro il nome dell'eseguibile su cui effettuare il debug.

  gdb helloworld.exe

- A questo punto apparirà il prompt del debugger, che nella maggior parte dei casi appare come *(gdb)*. Per uscire da tale modalità è sufficiente digitare **quit** o in maniera più breve solo **q**.

Eseguire il programma

- Il lavoro di debugging è molto sottile e complesso. Ci limiteremo oggi a qualche concetto di base.
- Dando il comando **run** (**r**) faremo in modo di eseguire la nostra applicazione che al momento è ancora in attesa.

```
(gdb) run
Starting program:
/server2/2/2009/bdicamil/prova/hello.exe
Hello World!
Program exited with code 014.
```

I breakpoint

- Dare come primo comando di un debug run non è una cosa molto sensata e potrebbe rivelare solo una certa tipologia di bug.
- Uno strumento molto potente che ci offrono i debugger sono i *breakpoint*, che come suggerisce il termine stesso sono dei punti di rottura nel normale flusso di esecuzione del programma.
- Un breakpoint viene creato utilizzando l'istruzione break

I breakpoint

break 1

break 4

- **break 1** inserisce un breakpoint alla prima riga dell'eseguibile, mentre **break 4** imposta un break alla linea 4 dell'eseguibile che abbiamo passato come parametro a gdb.
- Se diamo ora il comando

  run

  questo eseguirà il nostro programma fino al breakpoint (salvo che non entri prima in un loop infinito).

I breakpoint

- Durante questa fase di stallo indotta del programma è possibile ricavare il valore delle variabili utilizzando il comando

  print nome_variabile

  o assegnare ad una variabile il valore desiderato utilizzando il comando

  set variable nome_variabile=valore

I breakpoint

Il comando:

- **next**

  esegue la riga successiva (non entra nelle funzioni chiamate dal main)
- **step** esegue la riga successiva (entra nelle funzioni chiamate dal main)
- **c**

  riprende l'esecuzione dopo un breakpoint e si ferma al breakpoint successivo o alla fine del programma.

I breakpoint

Il comando:

- **kill**

  "uccide" (kill) l'esecuzione del programma
- **delete**

  cancella tutti i breakpoint
- CTRL-c blocca l'esecuzione se finisco in un loop infinito

Esercizio

ESEGUIRE L' ESERCIZIO IN LINUX

- Fare il debug del programma helloworld.c impostando il breakpoint alla linea 1 e utilizzando i diversi comandi visti nelle slide precedenti
- Scrivere in emacs il programma vertice.c che calcola le coordinate del vertice della parabola di equazione $y=ax^2+bx+c$ con a=1, b=2, c=0 (vedi codice nella pagina seguente). Compilarlo e farne il debug stampando e poi modificando i valori di a, b e/o c.
- Scrivere in emacs il programma whilecyc.c che stampa a schrmo x=3 x=2 x=1 (vedi codice nella pagina seguente). Commentare le istruzioni di aggiornamento della condizione (x--;), compilarlo e farne il debug.

Vertice.c

```
#include<stdio.h>
main(){
float a=1, b=2, c=0;
float D, Vx, Vy;
D=b*b-4*a*c;
Vx=-b/(2*a);
Vy=-D/(4*a);
printf("\n Il vertice della parabola ha coordinate
   \t%f\t%f\n",Vx,Vy);
}
```

whilecyc.c

```
#include<stdio.h>
main() {
int x=3;
while (x>0)
{ printf("x=%d\n",x);
  x--; }
 }
```