GDB Debugger

Cos'è il Debug?

- Il debug di un programma è la fase di ricerca e correzione di errori che si manifestano a run-time
- In genere questo accade quando sono presenti frammenti di codice errati detti bug

Come funziona un Debugger?

- Ogni programma C, essendo esso un linguaggio imperativo, è composto da una sequenza di istruzioni che modifica lo stato del programma
- Lo stato di un programma è l'insieme dei valori di tutte le variabili in un dato istante
- Il Debugger ci permette di eseguire le istruzioni di un dato programma una per volta e di analizzare lo stato dello stesso ad ogni passo

Manifestazione dei bug

- I bug che si manifestano a run-time possono essere suddivisi in 3 categorie:

  1. Interruzione inaspettata dell'esecuzione
  2. Mancata conclusione dell'esecuzione
  3. L'esecuzione termina fornendo risultati errati

Interruzione inaspettatta

- Ispezioniamo lo stato del programma nel punto in cui si è verificata l'interruzione
- Scopriamo quali modifiche dello stato hanno determinato l'interruzione
- Monitoriamo l'esecuzione del programma per determinare in quale punto lo stato è cambiato in modo errato

Mancata Conclusione

- Verifichiamo quali condizionali o istruzioni causano un loop nell'esecuzione
- Determiniamo se il loop è determinato da:
  - Uno o più condizionali errati
  - Un errato cambiamento stato
    - monitoriamo l'esecuzione del programma per determinare il punto in cui lo stato è cambiato in modo errato

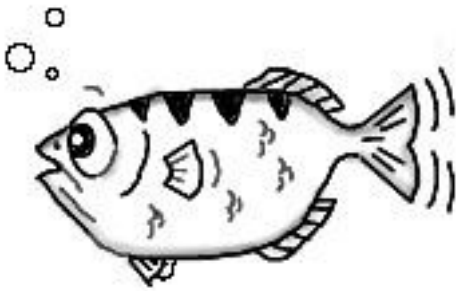
Esecuzione con risultato errato

- Verificando lo stato prima del risultato individuiamo le strutture dati che causano l'errore nel risultato
- Suddividiamo il programma in sezioni e verifichiamo se le singole sezioni di codice operano correttamente sulle strutture dati che causano l'errore e sulle parti dello stato da cui queste dipendono
- Individuiamo le sezioni di codice che modificano erroneamente lo stato del programma

GDB: The Gnu Project Debugger

- GDB è il debugger ufficiale del progetto GNU
- Lo sviluppo di GDB è stato intrapreso nel 1986 da Richard Stallman
- GDB è software libero rilasciato nei termini della licenza GPL (General Public License)
- GDB supporta i linguaggi C, C++, Objective C, Fortran, Java, Pascal, Assembler, Modula-2 e Ada

http://sourceware.org/gdb/



Per iniziare...

- Per iniziare ad utilizzare il debugger GDB è necessario compilare il programma con l'opzione -g:

```
[studente@lab]$ gcc –o test –g test.c
```

- L'opzione -g del compilatore gcc produce un file eseguibile contente informazioni di debug utilizzate da GDB
- Per eseguire il programma cn gdb si utilizza la sintassi:

```
[studente@lab]$ gdb test

. . .

(gdb)
```

- Il prompt (gdb) indica che gdb è in attesa di istruzioni

Comandi di base

I Comandi di base del GDB sono:

- run Avvia l'esecuzione del programma fino al prossimo breakpoint
- start Avvia l'esecuzione arrestandosi al main
- help Fornisce informazioni sui comandi
- quit Termina l'esecuzione del debugger
- she Esegue un comando in una shell

N.B.: Gli argomenti da passare al programma devono essere specificati all'avvio mediante i comandi run e start

Il Comando break

- Il comando break inserisce un breakpoint in un dato punto del programma
- Un breakpoint è un punto del programma in cui la sua esecuzione deve essere arrestata in attesa di istruzioni dell'utente

  Sintassi di break:

  - `(gdb) break` *`FUNCTION`*

    Inserisce un breakpoint all'entrata della funzione FUNCTION

  - `(gdb) break LINE`

    Inserisce un breakpoint alla linea LINE del file sorgente corrente

  - `(gdb) break … if COND`

    Questo break, valuta l'espressione COND ogni volta che il breakpoint viene raggiunto e fa fermare il programma solo se il valore dell'espressione COND è false

Monitorare una variabile

I comandi che permettono di visualizzare il valore di una variabile durante un'interruzione sono:

- `(gdb) print var`

  Stampa il valore della variabile var

- `(gdb) display var`

  Stampa il valore della variabile var ad ogni interruzione

L'operatore @ consente di visualizzare un numero arbitrario di elementi di un array:

- `(gdb) print A[0]@5`

  Visualizza i primi 5 elementi dell'array A

Monitorare una variabile: watch

- Il comando watch permette di monitorare una variabile quando viene letta e/o scritta, arrestando l'esecuzione del programma.

  Sintassi di watch:

  - `(gdb) watch` *`var`*

    Interrompe l'esecuzione se il programma modifica var ovvero quando vi accede in scrittura visualizzando il nuovo ed il vecchio valore d var.

  - `(gdb) rwatch` *`var`*

    Interrompe l'esecuzione se var è letta dal programma

  - `(gdb) awatch` *`var`*

    Interrompe l'esecuzione quando var è letta o scritta dal programma

Comando delete (d)

- Ogni volta che settiamo un break o un watch il debugger ci darà un output del tipo:

  ```
  Breakpoint 1 at 0x804844d: file pro.c, line 7
  ```

  dove l'intero dopo `Breakpoint` è l'identificativo del break.
- Il comando delete elimina un breakpoint o un watch:
  - `(gdb) delete NUM`

    Elimina il breakpoint o il watch contrassegnato dall'intero NUM.

Comandi per l'esecuzione

Dopo aver settato i break ed i watch ed aver mandato in run il programma bisogna continuare l'esecuzione:

- next esegue l'istruzione successiva e qualora questa sia una funzione, esegue l'intera procedura
- step esegue l'istruzione successiva e qualora questa sia una funzione, interrompe l'esecuzione alla prima istruzione della stessa
- cont esegue le istruzioni fino al prossimo breakpoint
- finish continua l'esecuzione di tutto il codice della funzione in cui si trova

Altri comandi

- list stampa le linee del codice sorgente corrente
  - list NUM stampa le linee di codice centrate attorno alla linea numero NUM
  - list FUN stampa le linee di codice centrate attorno all'inizio del codice della funzione FUN
- clear elimina tutti i break point
- info breakpoints stampa una tabella dei breakpoints e dei watchpoints settati, che non sono stati cancellati
- set var=val setta il valore della variabile var al valore val
- jump NUM salta alla linea NUM e fa partire il programma, a meno che non ci sia un break