Fundamentos de Software - Prácticas
Módulo II. Compilación y depuración de programas

Práctica 8: Depuración de programas

Departamento de Lenguajes y Sistemas Informáticos – Universidad de Granada

UNIVERSIDAD DE GRANADA

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Conocer cómo la utilidad `gdb` es capaz de seguir la traza de ejecución en ejecutables compilados con código fuente en un lenguaje admitido por `gcc` (C++)**
    - gdb (GNU Debugger): depurador portable para plataformas Unix y C, C++ o Fortran.
    - Trazar y modificar la ejecución de un programa.
    - Se puede controlar y alterar los valores de las variables internas del programa.
    - Por defecto se controla mediante una interfaz de línea de órdenes.

Ver qué pasa dentro de un programa cuando éste se ejecuta, o qué pasó cuando el programa dio un fallo y abortó

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Conocer cómo la utilidad `gdb` es capaz de seguir la traza de ejecución en ejecutables compilados con código fuente en un lenguaje admitido por `gcc` (C++)**
    1. Compilar el programa con g++ (o gcc) con la opción g que añade información necesaria para `gdb` de cara a poder depurar el programa
    2. Ejecutar el depurador `gdb`
    3. Dentro del intérprete del depurador, ejecutar el programa con la orden `run`
    4. Mostrar el resultado que aparece tras la ejecución.

```
g++ -g main.cpp hello.cpp
factorial.cpp -o ejemplo1

gdb ejemplo1
```

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Conocer las herramientas básicas de depurado y obtención de información de gdb**
    - **Comprobación de ayuda y listar código:**
      - `help:` permite obtener ayuda genérica del programa.
      - `list:` listar código
    - **Comprobación de variables y estados:**
      - Tabla 8.2 (`display, print, delete display, examine, show values, p/x $pc, x/i $pc, disassemble, whatis, info locals`)

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Conocer las herramientas básicas de depurado y obtención de información de gdb**
    - **Puntos de ruptura simples:**
      - Añadir puntos de ruptura simple para examinar qué hace el programa en un determinado lugar.
        - `break` (nombre de una función, la dirección lógica donde parar, o un número de línea.
        - `continue` (continuar el programa hasta el final o hasta el próximo punto de ruptura)
        - `next / step`: avanzar a la siguiente instrucción del programa
        - `info breakpoints`**:** ver los puntos de ruptura activos
        - `delete`**:** eliminar un punto de ruptura

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Saber construir guiones para gdb para automatizar la depuración.**
    - **Guion de gdb:**
      - Archivo de texto con diversas líneas de órdenes para la depuración de un programa

```
break multiplica
run
display x
display y
display final
continue
continue
delete display 1
delete display 2
continue
```

```
$gdb -x guion.gdb ejemplo1
```

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Conocer el manejo de marcos (frames) en gdb**
  - **Marcos en gdb:**
    - Pila (call stack): Direcciones donde se van a ejecutar determinadas funciones del programa
    - *Stack frames*: Secciones contiguas de la pila conteniendo un conjunto de datos asociados con una llamada a una función: argumentos, variables locales y la dirección en la que se ejecuta.
    - *innermost frame*: Marco de la función que se está ejecutando en un momento determinado (identificado por su dirección, *frame pointer register*)

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Conocer el manejo de marcos (frames) en gdb**
    - **Marcos en gdb:**
      - gdb emplea marcos en la depuración de un programa:
        - `info frame`: muestra información acerca del marco actual.
        - `backtrace full`: muestra la información referente a las variables locales y el resto de información asociada al marco.
        - `next(n)/step(s)`: Si estamos en una instrucción de llamada a un subprograma, `step` entra en el marco donde se encuentra ese subprograma ejecutando sus instrucciones paso a paso, `next` ejecuta el subprograma como si se tratase de una instrucción simple todo él
        - `down / up`**:** podemos elegir subir o bajar en la pila de marcos (función siguiente o anterior en la que se produjo la llamada)

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Saber utilizar las órdenes de gdb para modificar la ejecución de un programa y sus datos.**
    - **Puntos de Ruptura Condicionales:**
      - Habilita el punto de ruptura si se cumple la condición.
        - `break 13 if tmp > 10`
    - **Cambio de valores en variables:**
      - gdb permite cambiar el valor de una variable mientras se está depurando un programa.
        - `set variable variable=valor`

Práctica 8: Depuración de programas

- OBJETIVOS:
  - **Conocer las órdenes avanzadas de depuración de procesos en gdb**
    - `gdb` permite depurar programas que ya se encuentran ejecutándose en el sistema operativo (daemons):
      - programa con un tiempo de ejecución largo o que disponga de alguna instrucción que permita su detención (p.ej., petición de datos).
      - Una vez dentro de la utilidad `gdb`, ejecutar:
        - `attach PID` (identificador del proceso)
    - Integración de `gdb` con los editores el sistema:
      - Lanzar el editor desde `gdb`: cambiar el valor de la variable asociada
    - Ejecutar órdenes de la Shell desde `gdb`: ejecutar `shell` desde gdb

Práctica 7: Compilación de programas

- COMANDOS Y UTILIDADES:
  - `g++`
  - `gdb`
  - `make`