Arquitectura de Computadoras
GDB

Tabla de símbolos

Cuando un programa está compilado y linkeado, la información sobre nombres de variables, funciones, etc (la **tabla de símbolos**) se pierde: la computadora no los necesita para ejecutar su programa. Sin embargo, para permitir ver estos nombres identificatorios en GDB, queremos incluir la tabla de símbolos. Para decirle al compilador y linker que guarden la información de la tabla, usar los flags:

- Si compilo con nasm y linkeo con ld:

```
nasm -g -F DWARF -f elf gdbDemo.asm
nasm -g -F DWARF -f elf gdbLibDemo.asm
ld -g -m elf_i386 gdbDemo.o gdbLibDemo.o -o gdbDemo
```

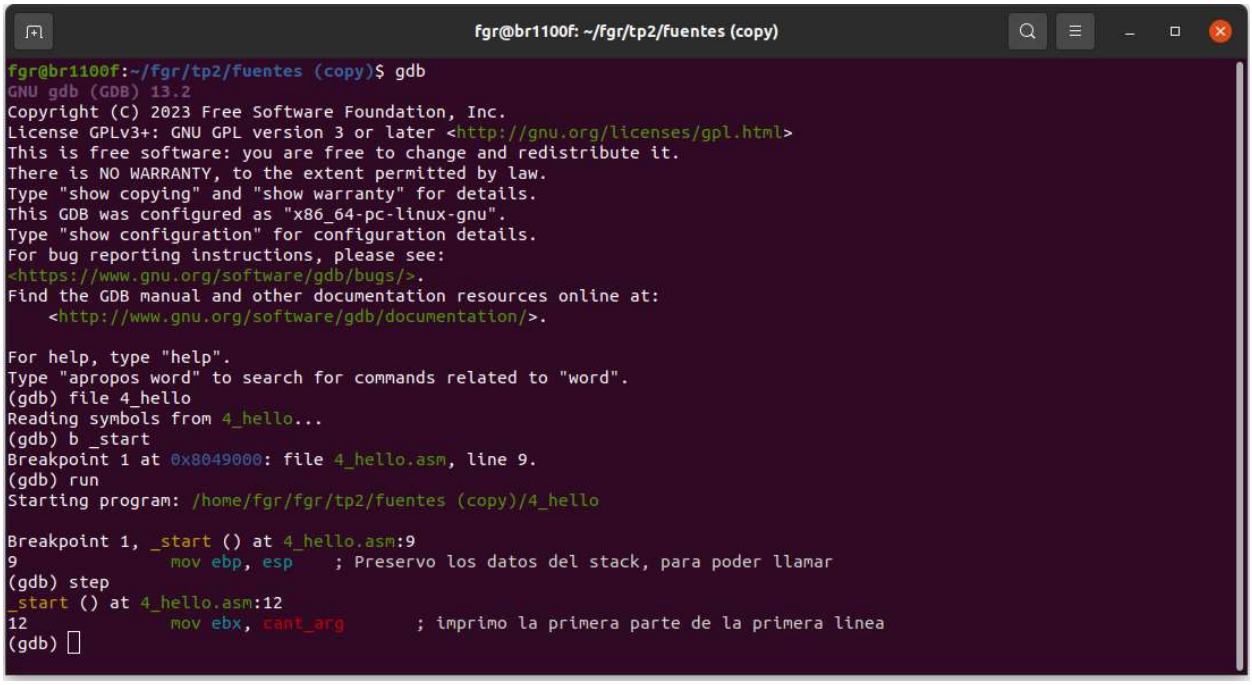
- Si compilo y linkeo con gcc:

```
gcc -g  gdbDemo.c gdbLibDemo.c -o gdbDemo
```

GDB necesita la tabla de símbolos, de lo contrario solo puede mostrar el assembler.

GUI para GDB

El GDB sin interfaz adicional, es una consola para debuguear que nos mostrara información del programa únicamente cuando nosotros ingresemos el comando apropiado.



Existe una forma de tener una visualización un poco más amigable: un flag “-tui” con el cual se puede correr GDB, lo cual nos provee un layout un poco mas amigable (posibilidad de ver el assembler, código fuente y/o registros). El layout es algo limitado (por ejemplo, no permite mostrar código fuente + assembler + registros simultáneamente) aunque es mucho más cómoda que la alternativa sin interfaz. El uso de este flag debe ser habilitado durante la compilación del código fuente de GDB, por lo que puede ser que según la versión de GDB que tengan, se lo reconozca o no.

De todos modos, la alternativa indicada a continuación nos va a permitir visualizar más información en pantalla y de modo muy cómodamente organizado, por lo que es algo mejor que el uso de “-tui”.

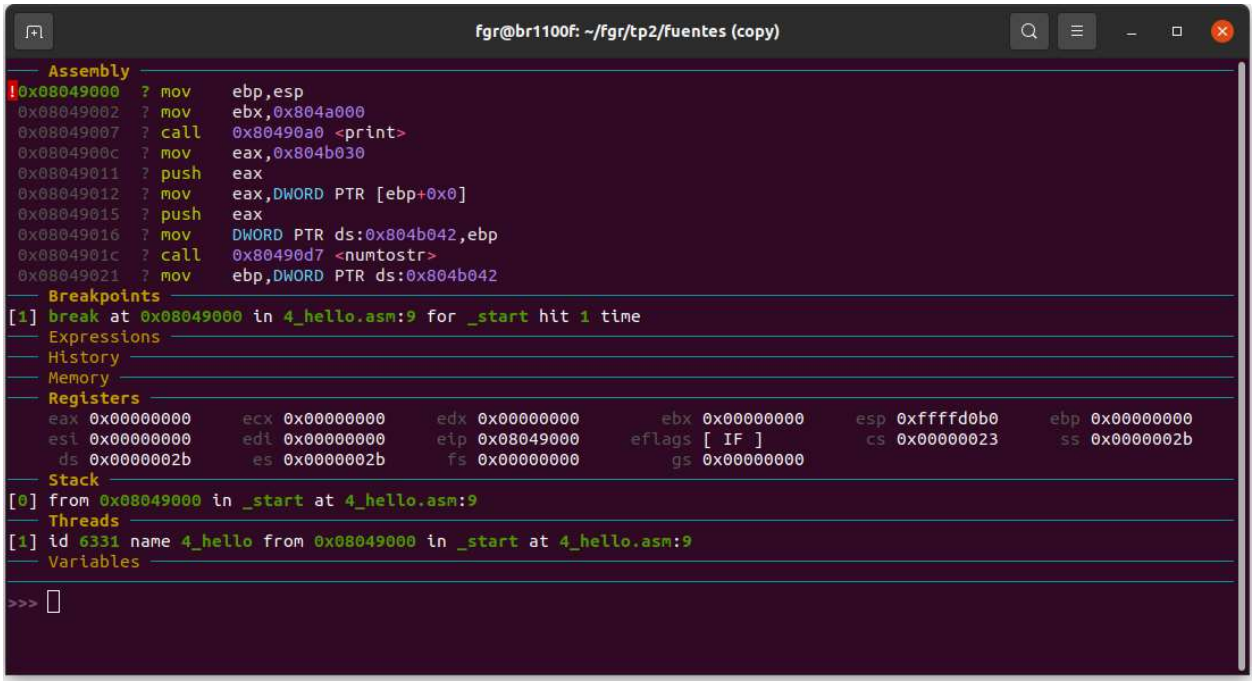
Mostrar más información simultáneamente: Dashboard (.gdbinit)

GDB tiene una API Python con la que se puede programar una mejor visualización de información. En el link a continuación, van a encontrar un **.gdbinit** que se debe colocar en el home (tu-user@nombreDevice:~$, es decir, dentro de /home/tu-user/).

```
https://github.com/cyrus-and/gdb-dashboard/
```

Alternativamente, el mencionado archivo, también lo dejamos disponible dentro del campus (carpeta comprimida “gdbArchivos.zip”) con el nombre de “.gdbinit__stack_meter_en_home” (deben renombrarlo **.gdbinit**).

Esta interfase visual para GDB (llamada Dashboard) escrita en Python, mostrara una pantalla como:



Muestra distinta información (seleccionable) una encima de la otra: Assembly, Breakpoints, Memoria, Registros, etc. Lo cual es una enorme mejora respecto del GDB sin la interfaz.

Una vez que ubicaron el archivo .gdbinit en el /home/tu-user/ pueden probar de correr el gdb. Si les indica que no tienen instalado python:

```
sudo apt install python3
```

Si luego les sigue indicando alguna otra dependencia, les va puntualizar exactamente cual, por lo que también la podrán instalar sin inconveniente.

El Dashboard va a imprimir la pantalla con datos a partir de que carguen un archivo para debuguear y se produzca la ejecución de alguna instrucción (comando step del GDB, por ejemplo).

Pygments

Para agregar colores al código:

```
sudo apt install python3-pip
pip install pygments
```

Mostrar la información en subdivisiones de la misma ventana: tmux

Esa ventana donde tenemos información casi en forma de pila, una arriba de la otra, la podemos mejorar. Dashboard soporta direccionar cada una de sus secciones “Source”, “Assembly”, “Registers”, etc, hacia distintas ventanas abiertas de consolas. Usando el programa tmux y otro script de inicialización que se puede bajar del siguiente link:

```
https://github.com/ficoos/tgdbdb
```

Podemos abrir una única consola y que tmux la divida en varias pero dentro de la misma ventana, para que otro script envíe cada salida del Dashboard a una zona especifica de nuestra ventana subdividida.

Las instrucciones para instalar todo las indica el mismo link, pero las repito aquí:

1. Instalar **tmux**
2. El archivo **.tgdbinit** colocarlo en el home (~$), junto con el .gdbinit.
3. Se indica “agregar tgdb a $PATH”. Esta variable de sistema contiene los directorios donde se buscan ejecutables. Una alternativa, en lugar de modificar $PATH, es colocar al contenido del repositorio del tgdb en un lugar apropiado (por ejemplo /usr/local/share/tgdbdb/).

   → Si están en Ubuntu, pueden directamente arrastrar la carpeta al destino con el mouse.

   → Si están en wsl no les dejará arrastrar archivos directamente desde el explorador a una ubicación que no sea \\wsl.localhost\Ubuntu-20.04\nombreDeUsuario\ por lo que primero deberán pasar los archivos a un directorio allí, y una vez dentro de Linux pueden moverlos:

```
sudo mv ~/tgdbdb /usr/local/share/
```

Posteriormente, nos fijamos cuales son los directorios que contiene la variable $PATH

```
echo $PATH
```

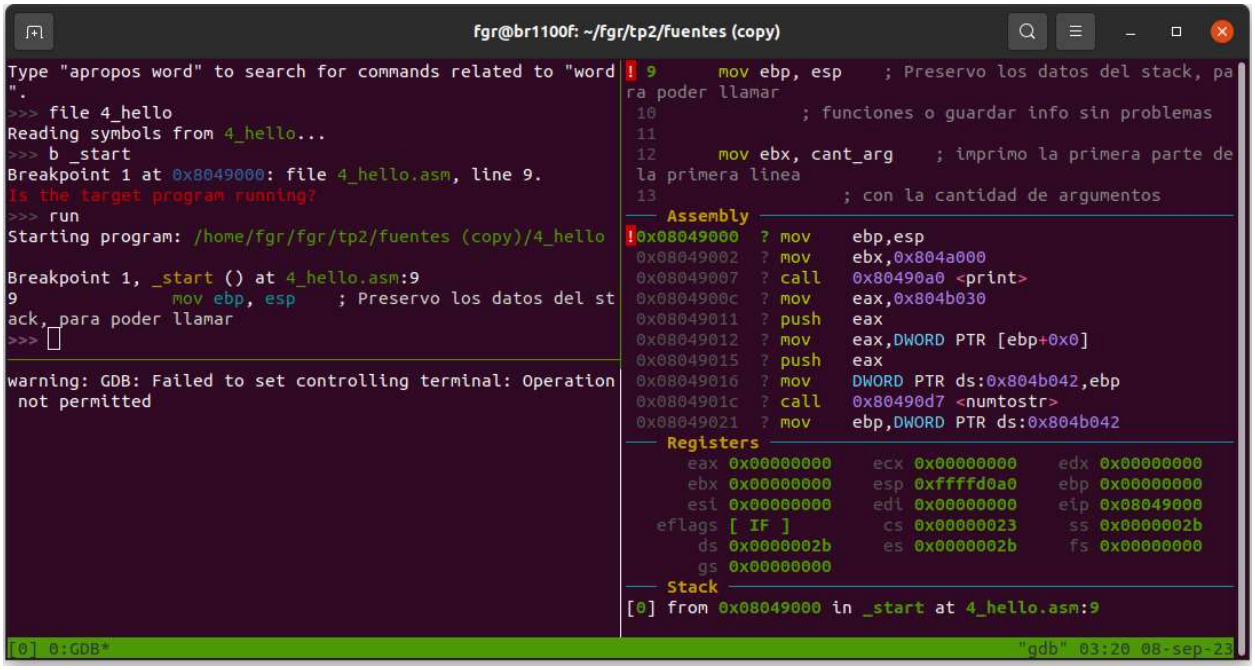
Por ejemplo, a mí me ha devuelto:

```
/home/fgr/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sb
in:/bin:/usr/games:/usr/local/games:/snap/bin
```

Ahora, sabiendo que /usr/local/bin/ se encuenta en $PATH, genero un symlink del archivo **tgdb** dentro de /usr/local/bin/ que apunte a donde se encuentra actualmente el archivo ejecutable: /usr/local/share/tgdbdb/tgdb

```
ln -s /usr/local/share/tgdbdb/tgdb /usr/local/bin/
```

Listo, deberían correr el comando tgdb y les aparecerá la pantalla:



Por defecto, tgdb nos genera una pantalla dividida en 3. Esto lo podemos mejorar aun mas. A continuación les paso unos ajustes adicionales.

Adicionales

Algunos comandos GDB

Indico algunos comandos del GDB, luego miren la cartilla (dejo 2: “GDB Cheat Sheet.pdf y “GDB Reference.pdf”).

```
>>> file nombre_de_ejecutable    → abre el archivo para debuguear
>>> set args estos son varios argumentos    → argumentos para el programa
>>> show args      → muestra los args
>>> b _start       → crea un breakpoint en _start
>>> run            → corre programa (para que se detenga se puso
                     breakpoint)
>>> s              → “step” ejecuta la siguiente instrucción
>>> [tecla ENTER]  → repite ultimo comando, en este caso, el “step”
```

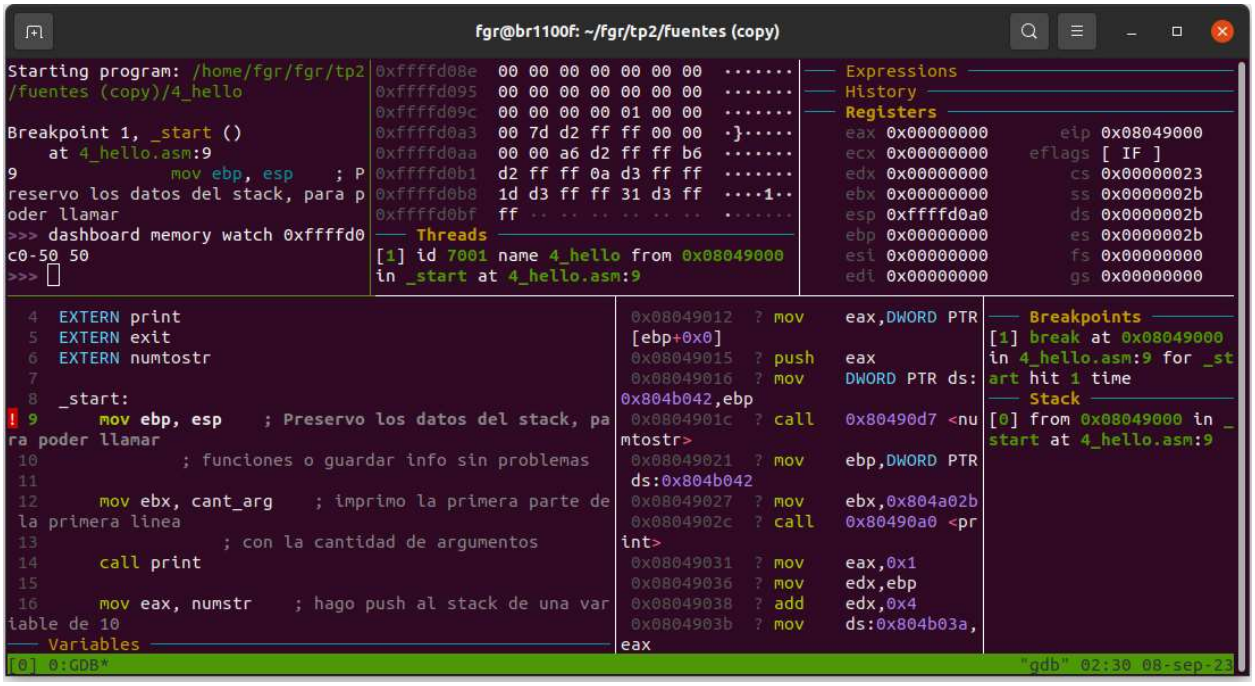
Algunos comandos Dashboard

La ayuda del dashboard la consiguen mediante:

```
>>> help dashboard
>>> help dashboard memory    → despliega ayuda especificamente sobre
                               comandos relativos a la ventana “memory”
```

Ajustando el layout (.tgdbinit__tmux_meter_en_home)

Para organizar las ventanas condensando mayor cantidad de información en distintas partes de la pantalla, les paso un archivo llamado **.tgdbinit__tmux_meter_en_home** con el layout siguiente:



El mencionado archivo, también ejecuta el comando.

```
>>> set disassembly-flavor intel
```

Para no tener que hacerlo cada vez. A este archivo renómbrenlo como .tgdbinit y lo pueden usar en lugar del original.

- Ocultar información

Pueden definir ustedes que info quieren ver en la pantalla y asi solo ver la info que les resulte útil, si por ejemplo, la sección de “Threads” molesta, se puede deshabilitar desde GDB:

```
dashboard threads    → encendido/apagado de visualización de “threads”
```

Si bien en la descripción en pantalla se muestra “Threads” (T mayúscula al inicio) en el comando ingresen con minúscula porque no lo reconocerá de otro modo, eso para cualquiera de las secciones.

- Mostrar memoria

```
dashboard memory watch 0xFFFFD0C0 50      → 50 bytes a partir de la direccion
```

Habilitar scrolling (.tmux.conf)

En la ventana de GDB (la que muestra mensajes y prompt) ahora se hizo mas pequeña y a veces queremos ver el output viejo, pero no nos permite hacer scroll hacia arriba. Eso se arregla habilitando el mouse en tmux: crear el archivo **.tmux.conf** con la siguiente línea:

```
set -g mouse on
```

Colocar a este archivo en el home, junto con .gdbinit y .tgdbinit.

Redefinir el tamaño de las ventanas

- Una manera rápida de mover la división horizontal, es clickear en la ventana que desea ajustar, presionar CTRL+B y usar flecha arriba/abajo para mover la division. Los cambios de esta manera, se resetean al salir de GDB.
- La otra alternativa es editar el archivo .tgdbinit, donde se otorga el tamaño porcentual de cada división.