Uso do GDB em ARQCP

(versão 1.5)

DEI - ISEP

2023/2024

lao@isep.ipp.pt

Executar o GDB

- Permite fazer depuração aos programas criados em ARQCP (tanto para código em C como para código Assembly)
- Crie ficheiro executável criado utilizando a opção **`-g`** do **`gcc`**
- Para executar o programa em modo de depuração dentro do GDB podemos executar na linha de comandos:

```
gdb -tui ficheiro_executavel
```

Dentro do GDB

- Se estiver interessado em visualizar o conteúdos dos registo pode utilizar o comando:

  `layout regs`

- Se, a qualquer altura, da depuração reparar que a informação apresentada pelo GDB está “estranha” execute o comando:

  `refresh`

- Quando quiser repetir um comando acabado de inserir basta pressionar a tecla **`ENTER`**

- O comando **`quit`** permite abandonar o GDB

Comandos de controlo da execução

- **`break n`** — Coloca um *breakpoint* na linha **`n`** (**`n`** pode ser o nome de uma função)
- **`break nome_ficheiro:n`** — Coloca um *breakpoint* na linha **`n`** do ficheiro nome_ficheiro
- **`clear n`** — Remove o *breakpoint* da linha **`n`**
- **`clear`** — Remove todos os *breakpoints*
- **`delete n`** — Remove um *breakpoint* (**`n`** é o número, atribuído pelo GDB, aquando da criação do *breakpoint*)
- **`run`** — Arrancar o programa
  - Se não tiver sido colocado nenhum *breakpoint* a execução do programa não é interrompida
  - Se o programa já estiver a executar permite reiniciar a execução
- **`start`** — O mesmo que colocar um *breakpoint* temporário, no início, seguido de **`run`**
  - Se o programa já estiver a executar permite reiniciar a execução
- **`kill`** — Permite terminar o programa que está a executar
- **`step`** — Avança para a próxima linha de código (entra em funções)
- **`next`** — Avança para a próxima linha de código (não entra em funções)
- **`continue`** — Continua a normal execução do programa até atingir um *breakpoint*

Comandos de informação

- **`print variavel`** — Imprime informação acerca de **`variável`**
- **`print/f variavel`** — Imprime informação acerca de **`variável`**, substituir **`f`** pelo formato pretendido:
  - **`a`** — Pointer
  - **`c`** — Character
  - **`d`** — Integer, signed decimal
  - **`u`** — Integer, unsigned decimal
  - **`f`** — Float point number
  - **`t`** — Integer, binary (t = “two”)
  - **`x`** — Integer, hexadecimal
  - **`o`** — Integer, octal
  - **`s`** — C string
- **`display variavel`** — Atribui um *display number* e vai imprimindo informação de **`variavel`** à medida que a aplicação executa
- **`undisplay display_num`** — Remove um determinado *display number*
- **`disable display_num`** — Desativa um determinado *display number*
- **`enable display_num`** — Ativa um determinado *display number*
- **`info locals`** — Imprime informação acerca de todas as variáveis locais
- **`watch`** — Coloca um *watchpoint* numa variável ou expressão, informa da alteração do valor. Exemplos:
  - **`watch x`**
  - **`watch x>0`**
  - **`watch *ptr==y`**

Comandos de informação

- **`backtrace`** — Apresenta as *frames* ativas do programa
- **`frame n`** — Seleciona a *stack frame* **`n`**
- **`info frame`** — Apresenta informação sobre a *frame* ativa
- **`x/nfu end_mem`** — Imprime **`n`** elementos em memória armazenados a partir de **`end_mem`**.
  - Deve substituir **`f`** pelo formato que pretende imprimir a informação e pode tomar os valores:
    - **`a`** — Pointer
    - **`c`** — Character
    - **`d`** — Integer, signed decimal
    - **`u`** — Integer, unsigned decimal
    - **`t`** — Integer, binary (t = “two”)
    - **`x`** — Integer, hexadecimal
    - **`s`** — C string
  - Deve substituir **`u`** pela unidade, que pode ser:
    - **`b`** — Byte
    - **`h`** — Half-word (2 Bytes)
    - **`w`** — Word (4 Bytes) (se a unidade for omitida será utilizada a unidade *word*)
    - **`g`** — Giant word (8 Bytes)

Comandos de informação (exemplos)

- **`print $ax`** — Imprime o valor do registo **`%ax`**
- **`print $ah`** — Imprime o valor do registo **`%ah`**

- **`print *((int*) $rsi)`** — Imprime o inteiro apontado pelo endereço de %rsi
- **`x/d $rsi`** — Imprime o inteiro apontado pelo endereço de %rsi

- **`x/4dg $rsp`** — permite verificar as últimas quatro *giant words* inseridas na *stack* que serão interpretadas como sendo inteiros decimais com sinal.
  - Aquando da impressão, o elemento mais à esquerda será o último elemento inserido na *stack* (topo da *stack*).

Outros comandos

- **`set`** — Permite modificar o valor de variáveis ou registos. Exemplos:
  - **`set x=2`**
  - **`set $eax=4`**

- **`focus src`** — Ativa o *scroll* para a janela de código fonte
- **`focus regs`** — Ativa o *scroll* para a janela de que apresenta os registos do processador
- **`focus cmd`** — Ativa o *scroll* para a linha de comandos

Mais informação

- http://www.yolinux.com/TUTORIALS/GDB-Commands.html

- Se pesquisar no Google por: *GDB cheatsheet*, encontrará vários documentos, com comandos frequentemente utilizados, cujos autores amavelmente disponibilizaram

- https://www.gnu.org/software/gdb/documentation/