

Minicurso da SAET

gdb e valgrind

O material do curso está disponível para *download* em
<http://maratona.td.utfpr.edu.br/minicursossaet.zip>

A: Entrada e saída no Linux

- Redirecionamento de entrada <
 `./programa < arquivo` roda o programa com a entrada no arquivo
- Redirecionamento de saída >
 `./programa > arquivo` salva a saída do programa no arquivo
- O comando `diff`
 `diff arquivo1 arquivo2` mostra as diferenças dos dois arquivos

Exemplo:

```
./bubble < in > saida
```

```
diff saida saidaesperada
```

Desafio: qual dos arquivos de saída o programa gera com a entrada `in`?

B: Compilação e comandos `b`, `r`, `n` e `p`

- Compile com `-g` para carregar os metadados para o `gdb`
 `gcc -o programa programa.c -g`
- Execute o `gdb` com o programa compilado
 `gdb programa`
- Comando `b [n]` (*breakpoint*) insere um *breakpoint* na linha `[n]`
 `b 14`
- Comando `r < arquivo` (*run*) executa o programa (com a entrada do `arquivo`)
 `r < in`
- Comando `p [var]` (*print*) mostra o valor da variável `[var]`
 `p a`
 `p v[i]`
 `p evair`
- Comando `n` (*next*) executa a próxima linha do programa
 `n`
 `p a`
 `n`
- Digite *Enter* para repetir o último comando.

Exemplo: debugar `buscabugada.c`

Desafio: debugar `subvetmaxbugado.c`

C: *breakpoint* em funções, e comandos `c` e `condition`

- Comando `b` pode ser usado em funções

```
b imprime
b insere
```
- Comando `c` (*continue*) continua a execução até o próximo *breakpoint* (ou até o fim)

```
c
```
- Comando `condition [n] [condicao]` pára a execução no *breakpoint* número `[n]` apenas se `[condicao]` for verdadeira

```
condition 1 i == 356
condition 1 a >= 0 && i == 42
```

Exemplo: ver variáveis com `i` específico no exemplo da parte A

Desafio: debugar `raizquadrada.c`

D: Comandos `tui enable` e `display`

- Comando `tui enable` habilita uma interface “*amigável*” para acompanhar a execução

```
tui enable
```
- Comando `display [var]` exibe o valor de uma variável a cada passo;

```
display a
display v[i]
```

Equivalente a rodar `p [var]` depois de cada comando.

Exemplo: debugar `sumabug.txt`

Desafio: debugar `pibuga.c`

E: *Segmentation Fault* e comando `where`

- Na ocorrência de um erro em tempo de execução, o comando `where` indica onde o erro ocorreu no código

```
where
```

Exemplo: debugar `vetorbugado.c`

Desafio: debugar `filabugada.c`

F: C++

O `gdb` também é compatível com C++

Exemplo: testar com `stl.cpp`

G: `valgrind`

- Ferramenta de análise da *memória* em tempo de execução;
- Encontra, entre outras coisas, memória alocada e não liberada;

```
valgrind ./programa < entrada
```
- Memória perdida *diretamente*: não há mais ponteiro para ela;
- Memória perdida *indiretamente*: os ponteiros para ela existem, mas em memória perdida diretamente.