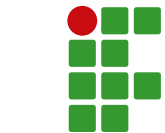
Depuração

Programação de Computadores I – Ciência da Computação

Prof. Daniel Saad Nogueira Nunes



INSTITUTO FEDERAL
Brasília

Campus
Taguatinga

Sumário

Sumário

Depuração

- É inevitável que os programas produzidos apresentem *bugs*, ou falhas de lógica
- Quanto maior o programa, maior a probabilidade de erros.
- A depuração consiste em encontrar e eliminar defeitos em softwares.
- Podemos usar depuradores (debuggers) para auxiliar neste processo.

Depuração

[.18]figuras/bugginhoUsar printf para achar o erro. Pode sim amiguinho!

Depuração

- Brincadeiras a parte, utilizar comandos de impressão em telas ou em arquivos para checar variáveis é um método de depuração denominado *Print debugging* (ou *tracing*).
- Nem sempre é efetivo.

Depuração

- Alguns depuradores possuem uma série de ferramentas que ajudam a localizar erros, tais como:
  - Parar a execução em determinadas linhas de códigos (ou funções).
  - Pular chamadas de funções e continuar a análise após a sua chamada.
  - Imprimir o conteúdo das variáveis. em determinada linha de código.
- Estes depuradores se encontram frequentemente atrelados à IDEs ou via linha de comando.
- Aprenderemos a utilizar o `GDB`.

Sumário

GDB

- Para utilizar o GDB, é necessário compilar os códigos fonte com a flag -g
- `gcc -g arquivo.c -o <nome_executável>`
- Se houver mais arquivos fontes, é necessário compilar todos com a flag -g e ligá-los para formação do executável com a tabela de símbolos de depuração.

GDB

- Primeiramente, é necessário carregar o executável através do GDB.
  - `gdb ./<nome_do_executável>`

Sumário

GDB

- Breakpoints, são pontos de parada.
- Toda vez que o depurador atinge uma linha do código marcada com breakpoint, ele para de rodar.
- É interessante colocar breakpoints em pontos problemáticos do código para detectar as falhas de lógica.
- Sintaxe:
  - `break <número_da_linha>`.
  - `break <nome_da_função>`.
- O comando `tbreak` possui a mesma sintaxe, mas assim que o programa atinge o este ponto temporário, ele é removido.

Sumário

GDB

- Para dar início à execução do programa, utilizamos:
  - `run`
- O programa rodará até o final, ou até atingir o breakpoint mais próximo.

Sumário

Comandos GDB

- `clear`: deleta todos os breakpoints.
- `clear <nome_da_função>`: deleta todos os breakpoints da função.
- `clear <número_da_linha>`: deleta os breakpoints relativos à linha especificada.

Sumário

Comandos GDB

- `continue`: continue executando até atingir o próximo breakpoint ou até o término do programa.
- `step`: executa a próxima instrução. Caso seja uma função, entra na função.
- `step n`: performa n `steps`.
- `s`: abreviação de step.
- `next`: executa a próxima instrução. Caso seja uma função, executa a função e pula para a próxima instrução.
- `next n`: performa n `next`.
- `n`: abreviação de next.

Comandos GDB

- `until <nome_da_funcao>`: continua a execução até atingir o nome da função.
- `until <número_da_linha>`: continua a execução até atingir o número de linha especificado.

Sumário

Comandos GDB

- `where`: mostra o número da linha corrente e o nome da função que está sendo executada no momento.
- `backtrace`: Imprime as funções empilhadas.

Comandos GDB

- `list`: imprime o código fonte.
- `list <nome_da_função>`: imprime o código fonte a partir da função especificada.
- `list <número_de_linha>`: imprime o código fonte a partir do número da linha especificado.
- `list <start>, <end>`: imprime da linha `start` até a linha `end` do código fonte.

Comandos GDB

- `print <nome_da_variável>`: imprime o valor da variável.
- `print <nome_da_variável>`: imprime o valor da variável.
- `print *<vetor>@<tamanho>`: imprime `tamanho` valores do `vetor`.
- `p`: abreviação de `print`.
- `p/x <nome_da_variável>`: imprime a variável em hexadecimal.
- `p/d <nome_da_variável>`: imprime a variável como inteiro com sinal.
- `p/u <nome_da_variável>`: imprime a variável como inteiro sem sinal.
- `p/o <nome_da_variável>`: imprime a variável como octal.

Sumário

IDEs e Ferramentas

- É possível usar o GDB de uma maneira mais gráfica e agradável.
- Diversas IDEs suportam a visualização da depuração, das quais podemos citar: `Code`, `Codeblocks`, `CLion` e `QtCreator`.
- Existem `front-ends` para o `gdb`, como a ferramenta `nemiver`.