Como sobreviver a ARQCP

2024/2025

Notas:

- Este documento é um resumo incompleto que deve ser completado pela prática, resolvendo os exercícios.

1 Uso do Linux

- Pode usar mais do que uma janela ao mesmo tempo sem as fechar (ex: um terminal para os seus programas mais outro para os testes). Pode/deve usar uma janela para o editor e outra para compilar.
- Escolha um editor que tenha ajuda sobre a sintaxe e mostre os números de linha do código.
- Use um resumo dos comandos do editor que escolheu para tirar partido das suas possibilidades.
- No antiX tem instalados os seguintes editores: `emacs`, `geany`, `joe`, `micro`, `nano`, `pico`, e `vim` (ordem alfabética).
- Para interromper um programa a combinação de teclas é `Ctrl-C`. A combinação de teclas `Ctrl-Z` suspende a execução de um programa, podendo-se retomar a sua execução com o comando `fg`.
- Pode comutar entre mais do que um programa no mesmo terminal usando o `tmux`[1], além de poder ter mais do que um *tab* no mesmo terminal.

2 Compilação

- Crie sempre um ficheiro `Makefile` para compilar usando o `make`.
- Compile sempre com o máximo de avisos ligados (`-Wall -Wextra -fanalyzer -g`).
- Veja bem os comandos que estão a ser executados em tempo de compilação.
- Comece a corrigir os seus programas sempre pelo primeiro aviso/erro.
- Veja ao mesmo tempo o seu código fonte e as mensagens de erro do compilador.
- Certos editores verificam a sintaxe ao gravar.
- Pode verificar código em C/C++ usando o comando `cppcheck *.c`

[1] `https://github.com/tmux/tmux/wiki`

3 Correr um programa

- Antes de correr um programa corrija todos os erros e avisos.
- Como vai correr o seu programa várias vezes, se o seu programa não ler dados da entrada, vai ser mais rápido.
- Os testes além de verificarem o valor de retorno das funções, testam o comportamento das funções.
- Quando um teste dá um erro diz onde a linha de código dos testes onde falhou.
- Se um programa passar os testes, isso não quer dizer que o programa está correcto. Só quer dizer que passou os testes que foram executados.
- Pode ver erros de memória (e onde um programa está a "estourar") usando o valgrind: `valgrind ./prog`
- Pode fazer a depuração de um programa usando o gdb (ou em modo gráfico com o ddd): `gdb -tui prog`
- O gdb não se comporta bem se os programas não forem compilados integralmente com a flag `-g`, ou quando as secções de código em assembly não estiverem bem definidas.

4 Erros mais comuns

- Variáveis não inicializadas em C.
- Cuidado com o `sizeof` que é um operador que funciona em tempo de compilação e que quando se pede o tamanho de um apontador, devolve o tamanho do apontador.
- Uso incorrecto de strings (atribuição, comparação):

```
str="One";   // errado!
...
if (str == "")  // errado!
...
```

- Note que a inicialização de strings é permitida, e deve haver espaço para o 0 final:

```
char str2[]="Two";  /* ou char str2[4]="Two"; */
```

- Isto é a inicialização de um apontador para uma string, que pode ser lida, mas não pode ser alterada:

```
char* str3="Three";
```

- Ao copiar uma string deve copiar sempre o 0 final.
- O `*` não faz parte do nome do apontador e é um operador. Uma asneira é:

```
*ptr++;  // errado!
```

- A linha seguinte de código C não faz nada:

```
str - string_size;  // errado!
```

- Respeite as convenções do uso dos registos em Assembly (os testes "estouram" se não as respeitar).
- Tenha presente o modelo de programação. Exemplo: o registo `%al` faz parte do `%ax`, que faz parte do `%eax`, que faz parte do `%rax`.
- Não confunda 'l' com '1': `mov1` em vez de `movl`