

Sistemas Operativos 2021/2022

Ficha Nº 2 Fundamentos de programação em C com gcc

Tópicos ->

Uso das ferramentas gcc
Ficheiros código-fonte, objecto e executável. Compilação e *linkagem*
Constituição de projectos em C com vários ficheiros
Argumentos em linha de comandos, variáveis de ambiente.

Exercícios

- Exercícios 1 a 5: Destinam-se a mostrar como funcionam os projectos em C e como se usam as ferramentas gcc, gdb e make, e abordam as diferenças e semelhanças entre o uso destas ferramentas e um IDE.
- Exercícios 6 a 9: Focam a passagem de argumentos em linha de comandos aos processos.
- Exercícios 10 a 12: Focam o uso de variáveis de ambiente.

1. Programação em C com as ferramentas gcc. Ficheiros objecto e executáveis.

- Construa um programa que imprime “ola mundo”.
- Compile-o com o gcc produzindo o executável. Experimente correr o programa assim feito.
- Produza apenas o ficheiro objecto (opção -c). Verifique que não o consegue executar, mesmo que lhe mude os atributos de ficheiro.
- Produza o ficheiro *assembly* (opção -S). Veja o conteúdo desse ficheiro com um editor de texto.
- Partindo do ficheiro objecto, produza o executável.

Objectivos do exercício - Funcionamento do gcc. Ficheiros intermédios *assembly* e objecto

- Entender os passos envolvidos na obtenção de um executável.
- Explorar opções básicas de gcc em linha de comandos: -S -c

2. Debugger gdb.

- Construa um programa que pede um número ao utilizador, de seguida apresenta esse valor elevado ao quadrado e finalmente pede um conjunto de caracteres ao utilizador apenas para fazer uma pausa antes de terminar.
- Compile-o com o gcc produzindo o executável. Teste o programa com alguns valores.
- Utilize o debugger gdb para analisar a execução do +programa e o conteúdo das suas variáveis.

Objectivos do exercício - Funcionamento do gdb

- Tomar conhecimento do debugger gdb e das suas possibilidades mais salientes.

3. Programação em C com gcc – projectos com vários ficheiros .h/.c e bibliotecas.

- a) Construa um programa com uma função “imprime” que imprime “ola mundo”. O programa deve estar organizado da seguinte forma:
- programa.c – tem a função *main* e usa a função *imprime*.
 - imprime.h – tem o protótipo da função *imprime*.
 - imprime.c – tem a implementação da função *imprime*.
- b) Compile o programa um ficheiro c de cada vez, gerando o respectivo ficheiro objecto
- c) Obtenha o ficheiro executável partindo dos ficheiros objecto que obteve.

Objectivos do exercício - Funcionamento do gcc. Ficheiros intermédios *assembly* e *objecto*

- Entender a organização de projectos com mais do que um ficheiro objecto.
- Explorar opções básicas de *gcc* em linha de comandos: *-s -c*

4. Utilitário *make*.

O utilitário *make* permite automatizar e gerir as tarefas relacionadas com a construção de programas. É bastante útil quando o projecto envolve vários ficheiros e bibliotecas. Este exercício ilustra muito brevemente as suas potencialidades.

- a) Usando os ficheiros do exercício 3 construa um ficheiro *make* para as seguintes opções:
- Construção do programa completo.
 - Construção do módulo (ficheiro objecto) *programa*.
 - Construção do módulo *imprime*.
 - Eliminação de ficheiros objecto e executáveis relacionados com o projecto.

Objectivos do exercício - Funcionamento do utilitário *make*

- Entender a lógica dos ficheiros *make*.
- Explorar opções básicas do utilitário *make*.

5. Utilização de ambiente integrado para compilação de projectos.

Se tiver um IDE no seu ambiente Unix, refaça o exercício 3 usando esse IDE.

- a) Compile e execute o programa inteiramente a partir do IDE.
- b) Use o IDE apenas para escrever o código. Através de um terminal, navegue para a directoria onde é armazenado o código fonte e compile e execute o programa usando *gcc*.

Use os comandos *whereis* ou *find* para o ajudar a encontrar essa directoria, ou então averigue nas propriedades do projecto (de acordo com o IDE em questão) qual a directoria onde estão os ficheiros.

Nota. Nesta disciplina, caso decida usar um IDE nas aulas práticas, a execução dos programas deverá ser sempre feita através da linha de comandos de uma forma semelhante à da alínea b).

Objectivos do exercício - Uso simultâneo de *gcc* e IDE

- Entender como usar o IDE para o desenvolvimento de código e a linha de comandos para a compilação a execução dos programas.
- Comandos *whereis* e *find*.

6. Argumentos de linha de comandos.

Elabore um programa que apresente no ecrã o nome que o programa tem (o nome do ficheiro executável), ou mais concretamente, *o nome que foi usado na linha de comandos para o executar*. Se tiver um IDE, elabore o programa com o editor do IDE. Caso contrário use o editor *pico*. Em qualquer dos casos, a execução deverá ser feita a partir da linha de comandos.

- Faça e execute o programa. Responda à questão “Como sabe o seu programa o nome que tem o seu ficheiro executável?”
- Se tiver um IDE, repita o processo, mas lance a execução a partir do IDE. O programa continua a saber o nome do seu ficheiro executável? Como? Qual a linha de comandos que foi usada? Averigüe onde pode configurar a linha de comandos no seu IDE para lançar o seu programa.

Objectivos do exercício - Passagem de argumentos no lançamento de programas

- Entender a forma como funciona o lançamento da execução dos programas em Unix.
- Entender as diferenças e as semelhanças entre lançar um programa explicitamente pela linha de comandos e através de um IDE.
- Entender como se pode passar informação para um programa quando se lança a sua execução.
- Compreender e saber usar os argumentos `int argc` e `char * argv[]` na função `main`.

7. Argumentos de linha de comandos.

Elabore um programa que escreve no ecrã todas as palavras que foram escritas na linha de comandos à frente do nome do programa, quando se lança a sua execução. As palavras são escritas uma em cada linha.

- Elabore o programa usando `argc`.
- Elabore o programa sem usar `argc`.

Objectivos do exercício - Argumentos no lançamento de programas

- Treinar os mecanismos de passagem de informação para os programas quando na altura da sua execução.
- Treinar o uso dos argumentos `int argc` e `char * argv[]` na função `main`.

8. Sintaxe de argumentos na linha de comandos.

Usando o programa do exercício anterior:

- Execute o programa escrevendo as palavras *um dois* à frente do nome do programa.
- Execute o programa, mas agora com “*um dois*” (dentro de aspas). Qual a diferença no output? O que pode concluir?
- Execute o programa indicando agora o carácter *. Veja o que aparece no ecrã. O que pode concluir acerca do funcionamento da *shell* na interpretação dos comandos?

Objectivos do exercício - Funcionamento dos argumentos no lançamento de programas

- Entender o papel da *shell* no pre-processamento da cadeia de caracteres usada no lançamento dos programas.
- Entender o papel de aspas e * na linha de comandos.

9. IDE - Integrated Development Environment

Se tiver um IDE no seu ambiente de trabalho Unix, e usando o programa do exercício anterior:

- Identifique, no IDE, o local onde especifica as palavras que o programa deve apresentar quando o executa a partir do IDE (anote esses detalhes para não esquecer essa informação).
- Use essa funcionalidade do IDE e lance o programa a partir do IDE especificando as palavras: um dois. O programa consegue obter essas palavras? Pode concluir que o mecanismo de passagem de informação ao programa que é lançado existe sempre?
- Usando o IDE, especifique as palavras “um dois” com as aspas indicadas. O que aparece? (o output pode depender do IDE).
- Usando o IDE, especifique o caracter *. O que aparece? (o output pode depender do IDE).

Objectivos do exercício - Mecanismos de passagem de parâmetros no lançamento de programas

- Entender que o mecanismo de argumentos no lançamento de programas existe sempre, mesmo quando o programa não é directamente lançado em linha de comandos.
- Entender que pode haver aspectos no processamento dos parâmetros que depende de quem lança o novo programa (IDE vs Shell/linha de comandos) (mais sobre isto num exercício na ficha seguinte).

10. Variáveis de ambiente – na linha de comandos

Este exercício não envolve programação em C. É necessário para entender a passagem de variáveis de ambiente de um processo para outro.

- Consulte as variáveis de ambiente na linha de comandos. Escolha uma e memorize o seu nome e valor. Utilize: **set** **set | more** **set | grep nome-da-variável** **echo \$nome-da-variável**
- Modifique o valor da variável que escolheu. Consulte novamente as variáveis de ambiente e verifique se o seu valor ficou mudado.
 - Usando **set**
 - Usando **export**
- Após modificar o valor da variável que escolheu, invoque uma nova *shell* e consulte o valor dessa variável. A nova *shell* corresponde a um novo processo, e interessa ver se o novo valor da variável é passado para este novo processo.
 - Use o comando **bash** para invocar uma nova *shell*.
 - Utilize o comando **echo \$nome-da-variável** para consultar o nome da variável.
 - Use o comando **exit** para regressar à sua *shell* original. Nota: se usar *exit* na *shell* original, terminará a sessão, ou encerrará o terminal (conforme o caso que esteja a usar).
- Tome nota das diferenças entre usar o comando **set** e usar o comando **export** no que diz respeito à passagem dos valores das variáveis para o novo processo. Repita o exercício anterior, mude o valor da variável na nova Shell e veja se o novo valor é propagado para a Shell inicial.
- A partir da sua *shell* original, invoque uma nova *shell*. Nessa *shell* defina uma variável *abcd* com o valor 1234. Confirme que a variável existe (comando **echo \$abcd**) e regresse à *shell* original (comando **exit**). Na *shell* original confirme que a variável *abcd* é desconhecida (não tem valor nenhum): **echo \$abcd** não apresenta nada. O que pode concluir acerca da possibilidade de passar variáveis “para trás” (de um processo para outro que já existia)? Anote as suas conclusões.

Objectivos do exercício - Funcionamento das variáveis ambiente – linha de comandos

- Compreender como é feita a definição de variáveis de ambiente em linha de comandos.
- Compreender a passagem de variáveis de ambiente para um novo processo.
- Compreender a diferença entre **set** e **export** no que diz respeito à passagem de variáveis para o novo processo.
- Compreender que não é possível passar variáveis para o processo anterior (já existente).
- Uso dos comandos **set**, **export**, **echo**, e **exit**.

11. Variáveis de ambiente – consola -> programa lançado

Escreva um programa que apresente no ecrã, uma por cada linha, todas as variáveis de ambiente.

- a) Consulte as variáveis de ambiente na linha de comandos (utilize: **set** e **set / more**).
- b) Execute o programa. Este deverá apresentar as variáveis de ambiente
 - I. Usando o argumento **envp**.
 - II. Usando a variável externa **environ**.
- c) Refaça a alínea b.ii) sem usar a palavra **extern**. Ao executar o programa, muito provavelmente este rebenta. Porquê?
- d) Refaça a alínea b) a função **getenv**.

Objectivos do exercício - Consulta e manipulação de variáveis de ambiente em C

- Compreender como é feito o acesso às variáveis de ambiente em programas em C.
- Compreender o acesso e uso a variáveis criadas em módulos exteriores ao código do programador (variável **environ**, módulo **crt1.o**).
- Entender o papel do qualificador **extern** no uso de variáveis externas ao módulo do programador.
- Função C **getenv**.

12. Variáveis de ambiente – programa lançado -> consola que o lançou

Elabore um programa que i) peça o nome de uma variável de ambiente e um valor para esta, ii) atribua o valor à variável, e iii) consulte e apresente no ecrã o valor dessa variável. No contexto desse processo, a variável deverá ter o valor especificado. Averigue a possibilidade de usar **setenv** e **putenv** e as diferenças entre os dois (veja também o que **unsetenv** faz)

- a) No terminal, após a execução do programa, averigúe o valor da variável que indicou. Confirme que a variável não tem o valor que deu durante a execução do programa, correspondendo ao facto de não ser possível afectar variáveis no processo “anterior” já em execução.

Objectivos do exercício - Funcionamento das variáveis de ambiente

- Consolidar os conceitos de variáveis de ambientes.
- Compreender que não é possível passar variáveis para o processo anterior (já existente).
- Uso dos comandos **set**, **export** e **echo**.
- Funções **setenv** e **putenv**.