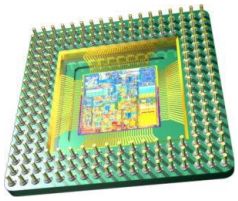

Lab 5 – Linguagem C e Compilação de Programas

Neste trabalho laboratorial pretende-se dar uma breve introdução à linguagem C, bem como uma visão do processo de conversão de um programa em código executável.

1. Utilização das máquinas "mips.deec.uc.pt"

O laboratório tem dois servidores com base na arquitetura MIPS que correm uma distribuição adequada de LINUX.

Para se ligar ao servidor e transferir ficheiros precisa de utilizar "ssh" (secure shell) e "sftp", respetivamente. Se for utilizador Linux/Mac, para se ligar ao servidor execute o comando:

```
ssh mips.deec.uc.pt
```

e terá acesso à maquina MIPS, quer esteja dentro ou fora da LAN do DEEC.

Se for utilizador Windows, deverá instalar os clientes "ssh" e "sftp" grátis PuTTY (http://www.putty.org/) e WinSCP (http://winscp.net), respetivamente. Estes clientes fazem exatamente o mesmo que os comandos Linux/Mac referidos anteriormente, mas usando uma interface com janelas.

Como vai estar a trabalhar num servidor remoto, só tem controlo por linha de comando de terminal (não há rato). Assim, e de forma a editar os seus ficheiros de código, deverá usar uma das seguintes estratégias:

1. Pode editar os ficheiros localmente com o seu editor preferido (e usando rato), e transferi-los de e para o MIPS usando "sftp". O WinSCP, inclusivamente, possui um editor de texto simples que atualiza remotamente o ficheiro em edição sempre que fizer "save" – basta fazer "double-click" no ficheiro em causa na janela do WinSCP para o começar a editar.
2. Ter simultaneamente abertos dois terminais. Um terminal é para executar comandos (gcc, correr executáveis, etc.) e o outro terminal é para editar ficheiros. Terá de usar um editor de texto localmente instalado. Recomendamos o "emacs" (ver comandos em material de apoio).

2. Para treinar em casa – utilizadores Windows

Se o seu PC tem sistema operativo Windows, e se este tiver pelo menos 1GB de memória, e não tiver oportunidade de se ligar aos servidores remotamente, pode instalar uma máquina Linux virtual.

Se for ao site http://www.virtualbox.org/ poderá criar um CD com a imagem do sistema Linux-Kubuntu, que irá ser instalado na caixa virtual que irá correr no seu sistema operativo. A partir daqui, poderá seguir as indicações dadas na secção que se segue.

3. Para treinar em casa – utilizadores de Linux/Mac

O compilador de C da GNU já vem com a maior parte das distribuições de Linux. Para verificar a instalação faça o download do ficheiro com código fonte `sum_v1.c` (no material da cadeira no Inforestudante) e coloque-o numa diretoria; abra uma janela de terminal e coloque-se na diretoria onde tem o ficheiro; crie um executável escrevendo o seguinte comando:

```
gcc sum_v1.c -o sum_v1
```

Corra o programa chamando `./sum_v1`. ("`./`" é essencial para que o programa corra!). Correu tudo bem? Então está pronto para começar os trabalhos laboratoriais da disciplina.

4. Realização do trabalho

Para este trabalho deverá ter em posse os ficheiros `sum_v1.c` e `sum_v2.c`, bem como todo o restante material de apoio.

1) Analise, compile e teste `sum_v1.c` no PC do laboratório utilizando a flag '-o' para definir o nome do ficheiro de saída (neste caso um executável). Por segurança deve guardar os seus trabalhos no servidor dos alunos no final da aula.

2) Agora utilize o compilador `gcc` com as flags '-E', '-S' e '-c'. No primeiro caso, algo será escrito no ecrã, enquanto que nos restantes os resultados serão guardados nos ficheiros `sum_v1.s` e `sum_v1.o`, respetivamente, na mesma directoria que o ficheiro original. Leia o que aparecer no ecrã e abra esses ficheiros com o editor de texto, verificando o seu conteúdo com atenção. Consulte os slides para conseguir explicar os resultados que obteve.

3) Concentremo-nos agora no ficheiro `sum_v1.s`. Edite-o e localize a instrução '`addu`'. Troque '`addu`' por '`subu`' e guarde as alterações. Compile o ficheiro `.s` modificado. Corra o executável entretanto criado e explique os resultados observados.

4) O código em `sum_v1.c` faz uso das funções `printf()` e `scanf()` das bibliotecas standard do C. Consulte `man pages` (https://linux.die.net/man/, que contêm o índice

geral, https://linux.die.net/man/3/printf e também https://linux.die.net/man/3/scanf) para saber mais sobre estas funções. Altere o programa de modo a que o resultado da soma ***seja escrito em hexadecimal***. Faça a soma de dois números negativos e explique o valor em hexadecimal que vê impresso no ecrã (representação em complementos de 2).

5) Analise, compile e teste `sum_v2.c`. Observe as diferenças que existem entre o código-fonte dos dois programas.

6) Imagine que pretende disponibilizar a função `soma()` para ser utilizada por múltiplos programas. Para tal deve criar um *header file* `soma.h` com a declaração da função, e soma.c com o código da função:

soma.h:

```
int soma(int  , int  );
```

soma.c:

```
int soma(int a, int b){
  return a+b;
};
```

O *header file* permite indicar a existência de uma função que recebe dois inteiros e devolve um inteiro, e deve ser invocado no início do programa com `#include "soma.h"`, para que o pré-processador junte essas linhas ao código passado ao compilador. Com essa informação, mesmo sem o código da função, podemos compilar o programa. Na fase de ligação ("linkage", ou em português "técnico", "linkagem") o código-objeto tem de ser disponibilizado.

7) Crie uma nova versão do programa, `sum_v3.c`, que recorra à função `soma()` indicada pelo *header file* `soma.h`. Compile separadamente `soma.o` e `sum_v3.o` a partir de `soma.c` e `sum_v3.c`, fazendo depois a ligação com:

```
gcc sum_v3.o soma.o –o sum_v3.exe
```

8) Para evitar escrever repetidamente comandos longos e ter em conta as dependências dos ficheiros, podemos recorrer à ferramenta Make. Num makefile são especificadas as dependências e linhas de comando para compilação, bastando executar o comando `make` na linha de comando. O `make` só vai re-compilar os componentes necessários (i.e., faz compilação incremental), tendo em conta a data dos ficheiros. Para o exemplo anterior seria:

```
sum_v3: sum_v3.o soma.o
      gcc -o sum_v3 sum_v3.o soma.o
sum_v3.o: sum_v3.c soma.h
      gcc -c sum_v3.c
soma.o: soma.c
      gcc -c soma.c
```

Crie o makefile com as linhas acima indicadas, grave com o nome `makefile`, e teste com o comando `make`. (Para mais informações sobre o uso da ferramenta Make, consulte http://mrbook.org/blog/tutorials/make/ e http://en.wikipedia.org/wiki/Makefile)

9) Faça agora uma função ***adicional***, como fez para a soma, incluindo ficheiro de código-fonte e respetivo *header file*. Esta função deverá receber um valor como argumento e escrever no ecrã a sua conversão para octal. Modifique também o código `sum_v3.c` (e o makefile) de forma a escrever no ecrã cada uma das parcelas e o resultado da soma em octal.

10) Da mesma forma, acrescente agora uma nova função (sempre com código-fonte em ficheiro separado) que implemente $c=(b-a)^b + (b+a)^a$ recorrendo a ciclos. Faça três versões diferentes dessa função, cada uma usando um tipo de ciclo diferente (`for`, `while` e `do-while`).