

# Arquitetura de Computadores

LIC. EM ENG.<sup>a</sup> INFORMÁTICA  
FACULDADE DE CIÊNCIAS E TECNOLOGIA  
UNIVERSIDADE DE COIMBRA



## Lab 6 – Programando Num MIPS Real

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

### 1. Utilização da máquina “mips.deec.uc.pt”

O laboratório tem um servidor com base na arquitetura MIPS que corre uma distribuição adequada de **LINUX**.



Para se ligar ao servidor e transferir ficheiros precisa de utilizar ferramentas de acesso remoto como o “ssh” (secure shell) e o “sftp” (secure FTP). Se for utilizador Linux/Mac, pode ligar-se ao servidor MIPS através de um terminal correndo o seguinte comando:

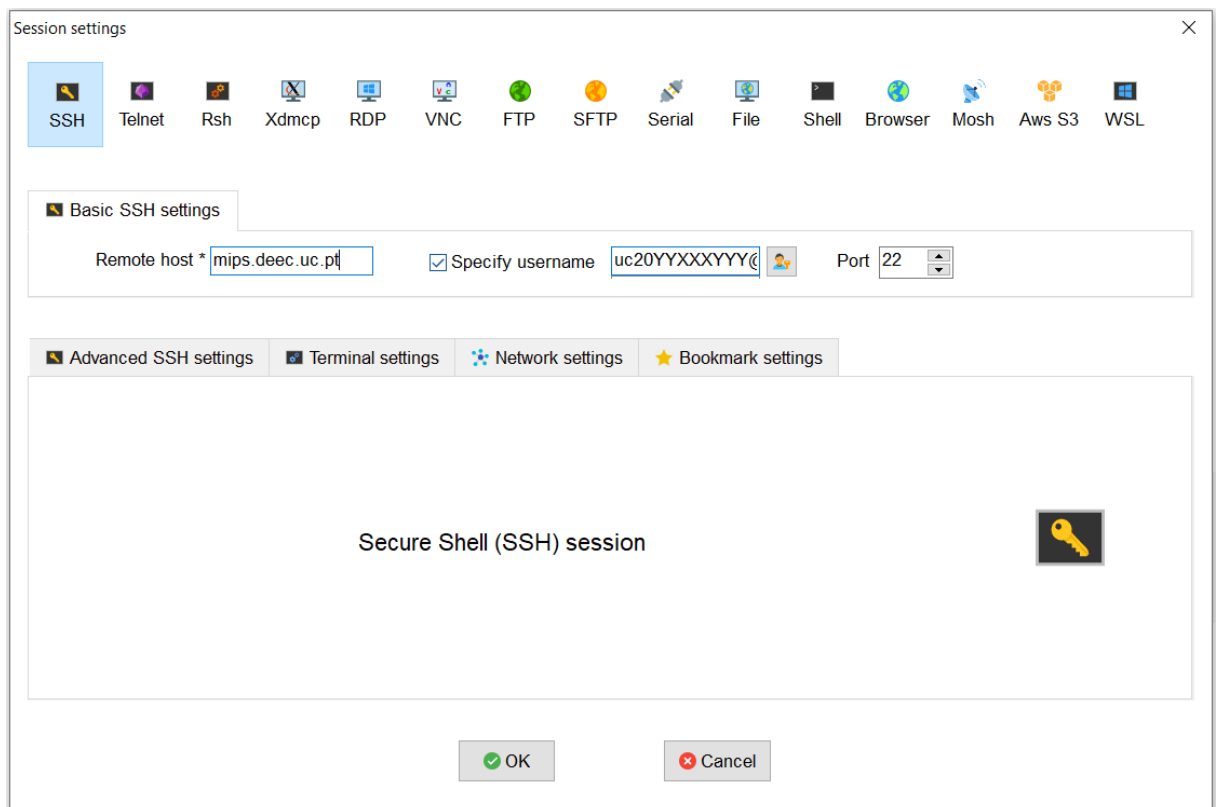
```
$ ssh 'uc20YYXXYY@student.uc.pt'@mips.deec.uc.pt
```

Deverá utilizar as credencias associadas à conta da Universidade (InforEstudante).

Se for utilizador **Windows**, deverá instalar clientes “ssh” e “sftp” gratuitos. Uma sugestão é utilizar o **MobaXterm** (<https://mobaxterm.mobatek.net/>) que permite ter ambos os clientes num só programa. Estes clientes fazem exatamente o mesmo que os comandos Linux/Mac referidos anteriormente, mas usam uma interface gráfica. Os utilizadores do MacOS podem utilizar a aplicação Cyberduck (<https://cyberduck.io/download/>) que é gratuita para a transferência de ficheiros. Devem utilizar a opção SFTP. Em alternativa podem experimentar as aplicações integradas (SFTP e SSH) **Termius** (<https://www.termius.com/>) ou **Royal TSX** (<https://www.royalapps.com/ts/mac/features>). Estas ferramentas podem não ser gratuitas ao fim de algum tempo.

Após instalar e executar o **MobaXterm**, para se ligar ao servidor deverá seguir os seguintes passos:

1. Carregar no botão *Session* no canto superior esquerdo  *Session*
2. Deverá abrir uma nova janela com o título *Session Settings* que sugere a escolha do tipo de sessão pretendido. Deverá carregar em *SSH*  *SSH*
3. Preencher os campos vazios com os seguintes dados:



No *Remote Host* deverá introduzir ***mips.deec.uc.pt***, deve seleccionar ***Specify username*** e preencher com as credencias associadas à conta da Universidade (Inforestudante) [uc20YYXXXYYY@student.uc.pt](mailto:uc20YYXXXYYY@student.uc.pt). Ao carregar *Ok* irá abrir na consola um comando para inserir a password associada à sua conta. **Ao introduzir a password não haverá nenhum feedback na aplicação. É mesmo assim!** Se preencheu bem os seus dados irá surgir uma opção de guardar credenciais que permite evitar ter de colocar password em acessos futuros. Salve apenas as suas credenciais se estiver a trabalhar no seu computador pessoal.

**Nota: De acordo com o Gabinete de Rede Informática, a primeira vez que tentar entrar no servidor irá receber a indicação de que ocorreu um erro. Efectivamente a primeira tentativa permite apenas criar a sua conta no servidor. Deverá cancelar a operação (CTRL+C) e tentar novamente. À segunda tentativa já deverá conseguir aceder ao servidor.**

4. Se tiver iniciado a sessão de forma correcta, terá neste momento acesso ao cliente *ssh* (consola) e *sftp* (explorador de ficheiros do lado esquerdo). Para enviar ficheiros para o servidor basta arrastar os mesmos para a zona do cliente *sftp*. Poderá também criar ficheiros carregando com o botão direito na zona do cliente *sftp* e seleccionando ***New empty file***.

5. Pode aceder remotamente a esta máquina a partir de sua casa. No entanto, e por questões de segurança, só o poderá fazê-lo se estiver ligado à VPN do DEEC. Para obterem indicações sobre a instalação/configuração da VPN do DEEC nos vossos computadores podem consultar a seguinte página:

<https://kb.deec.uc.pt/books/deec/chapter/vpn-jAV>

## 2. 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. **Analise o material de apoio para saber como compilar e executar ficheiros.**

- 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). Para executar o ficheiro deverá correr o comando `./'nome_do_ficheiro'`.
- 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 contém 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
```

- 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` para efectuar a compilação. O `make` só vai recompilar os componentes necessários, tendo em conta a data dos ficheiros (i.e., faz compilação incremental). Para o exemplo anterior, o *makefile* seria o seguinte:

```
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) Implemente agora uma função **adicional**, como fez para a soma, incluindo o ficheiro com o código-fonte e o 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.