

Lab 1 – Linguagem C e Compilação de Programas

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

1. Utilização das máquinas “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 respetivamente `ssh` (secure shell) e `sftp` (secure file transfer protocol).

Se for **utilizador Linux/Mac** abra uma janela de terminal e faça

```
ssh 'ucXXXXXXXX@student.uc.pt'@mips.deec.uc.pt
```

introduzindo depois as suas credenciais normais de utilizador do DEEC/UC¹ (username e password). Note que a primeira vez que acede ao servidor desta forma a conta é criada e, portanto, o acesso falha. É necessário voltar a aceder da mesma forma para que o acesso seja bem-sucedido.

Para transferir ficheiros entre a sua máquina local e o servidor mips deverá usar `sftp` (linha de comando) ou, em alternativa, usar o programa gratuito **Cyberduck** (<https://cyberduck.io/download/>). Pode encontrar em anexo as páginas de manual (man pages) destes serviços.


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 usando uma interface gráfico. Para utilizadores do MacOS podem experimentar as aplicações **Termius** (<https://www.termius.com/>), **Royal TSX** (<https://www.royalapps.com/ts/mac/features>), ou qualquer cliente **SSH** que encontrem.

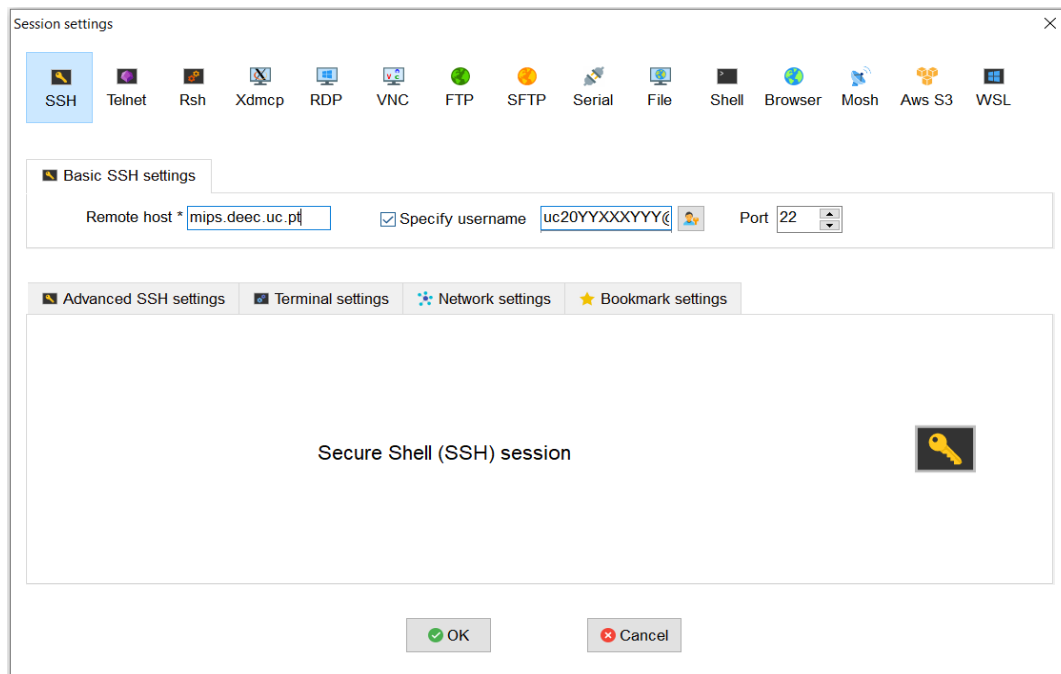
Após instalar e executar o **MobaXterm**, para se ligar ao servidor deverá fazer os seguintes passos:

1. Carregar no botão *Session* no canto superior esquerdo



¹ No caso de não ter conta no DEEC/UC deverá dirigir-se ao serviço de administração de rede no piso 2 antes da aula.

- Deverá abrir uma nova janela com o título *Session Settings* que sugere a escolha do tipo de sessão pretendido. Deverá carregar em *SSH* .
- Preencher os campos vazios com os seguintes dados:



No *Remote Host* deverá estar `mips.deec.uc.pt`, deve seleccionar *Specify username* e preencher com as credencias associadas à conta da Universidade (Inforestudante) uc20YYXXYY@student.uc.pt. Ao carregar *Ok* irá abrir na consola um comando para inserir a password associada à sua conta. Se preencheu bem os seus dados irá surgir uma opção de guardar credenciais que permite evitar ter de colocar password em acessos futuros.

Notas:

- Salve apenas as suas credenciais se estiver a trabalhar no seu computador pessoal.
- **De acordo com o Gabinete de Rede Informática, a primeira vez que tentar entrar no servidor irá falhar e a conta será criada. Deverá cancelar a operação (CTRL+C), tentar novamente e já conseguirá aceder ao servidor.**

2. Realização do trabalho

- 1) Para este trabalho deverá ter os ficheiros `sum_v1.c` e `sum_v2.c`, bem como todo o restante material de apoio. No seu computador local comece por abrir uma ligação `ssh` e `sftp` com o servidor de mips. Faça o download do material de apoio para a máquina local e transfira para o servidor mips usando `sftp`. Se fizer `ls` na linha de comando do mips deverá ver os ficheiros lá colocados
- 2) 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).

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

Corra o resultado fazendo `./sum_v1`. (`./` é essencial para que o programa corra!)

- 3) Agora utilize o compilador `gcc` com as flags `-E`, `-S` e `-c`. No primeiro caso, algo será escrito no ecrã, enquanto nos restantes os resultados serão guardados nos ficheiros `sum_v1.s` e `sum_v1.o`, respetivamente, na mesma diretoria 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.
- 4) Concentremo-nos agora no ficheiro `sum_v1.s`. Edite o ficheiro e localize a instrução `'addu'`. Troque `'addu'` por `'subu'` e guarde as alterações. Compile o ficheiro modificado até ao fim. Corra o executável e explique os resultados observados.
- 5) O código em `sum_v1.c` faz uso das funções `printf()` e `scanf()` das livrarias standard do C. Consulte as `man pages` (pesquise na internet por `'man pages'` ou, na janela de terminal digite `man printf -S3`) para saber mais sobre estas funções. Altere o programa de forma 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).
- 6) Analise, compile e teste `sum_v2.c`. Observe as diferenças que existem entre o código-fonte dos dois programas.
- 7) 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 ao 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 que ser disponibilizado.

- 8) 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
```

- 9) Para evitar ter que repetidamente escrever 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 fazer `make` na linha de comando. O `make` só vai re-compilar os componentes necessários, 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, por exemplo, <http://en.wikipedia.org/wiki/Makefile>.)

- 10) 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 hexadecimal. Modifique agora o código `sum_v3.c` (e o *makefile*) de forma a escrever no ecrã cada uma das parcelas e o resultado da soma em hexadecimal.
- 11) Da mesma forma, acrescente agora uma nova função (sempre com código-fonte em ficheiro separado) que implemente uma potência, $c=a^b$, recorrendo a um ciclo. Faça três versões diferentes dessa função, cada uma usando um tipo de ciclo diferente (`for`, `while` e `do-while`).

Microprocessors Systems 2021/2022

UNIX basic commands

Connect to DEEC's MIPS server using: ssh
 'ucxxxxxxxxx@student.uc.pt'@mips.deec.uc.pt

Command	Description	Example
pwd	print working directory	>> pwd /home/user/lab1
mkdir [directory]	create/make directory	# create 'lab1/' folder >> mkdir lab1
cd [directory]	change directory	>> pwd /home/user >> cd lab1 # enter lab1 folder >> pwd /home/user/lab1 >> cd .. # go back to prev folder >> pwd /home/user
	<tip> change to user's home directory	>> pwd /home/user/lab1/newfolder/stuff >> cd ~ >> pwd /home/user
ls [directory]	list the contents of the current directory	>> ls example.c example.o example >> ls lab1/ # list content of lab1 sum_v1.c sum_v2.c
mv [oldfile] [newfile]	change the name or the location of a file	>> ls file.c >> mv file.c renamefile.c >> ls renamedfile.c
rm [file]	remove/delete file	>> ls file1.c file2.txt >> rm file2.txt >> ls file1.c
rmdir [directory]	removes <i>empty</i> directories (use rm -rf directory for nonempty folders)	>> ls lab1 lab2 lab3 >> rmdir lab1 ls lab2 lab3
cp [src] [dst]	copy file	>> ls file.txt Folder >> cp file.txt Folder >> ls Folder file.txt
more [file]	print content of file in screen	>> more file.c # press [enter] key to continue # press 'q' key to quit
man [name]	query manual page for a given name	>> man strlen # check strlen manual # use 'q' key to quit
nano [file.xyz]	simple file editor	>> nano abc.txt # create or edit # file abc.txt

Other:

>> ./executable	run an executable program
Ctrl+C	interrupt/cancel program's execution
TAB+TAB	autocomplete names and commands