Exercícios

1. (a) Criar uma directoria **X** e copiar para lá todos os ficheiros da directoria actual.

   (b) Apagar todos os ficheiros da directoria X pedindo confirmação .

   (c) Remover a directoria X.

2. Construir um comando que determine quantos ficheiros existem na directoria actual.

3. Construir um comando que mostre um ficheiro da linha **10** à linha **25**.

4. Construir um comando que coloque num ficheiro *aux* todas as linhas de outro ficheiro excepto as que vão da **10** à **25**.

5. Construir um comando que coloque num ficheiro *aux* os nomes de todos os utilizadores que têm acesso ao sistema.

   (a) por ordem alfabética do Nome

   (b) por ordem alfabética do nome de utilizador (*logname*)

   (c) por ordem crescente do nmero de utilizador

6. Ler as mensagens que existem na caixa do correio (seguindo as instruções nelas contidas).

7. Ao ficheiro onde guardou uma das mensagens aplique as seguintes transformações :

   (a) Apagar todas as linhas que não começam por **@**

      i. usando *vi*

      ii. usando *grep*

   (b) Mudar todas as linhas
   @....{label,
   para
   \item{label}

      i. uma a uma (para quem tiver paciência !!!)

      ii. definindo a acção a executar através de **map** e aplicando esse comando a cada uma das linhas

      iii. com um nico comando de substituição .

8. Editar o ficheiro *.exrc* de forma a ficarem definidas teclas para

   (a) colocar o cursor no fim da linha (associada a CTRL-E)

   (b) inserir um RETURN (associada a CTRL-R)

(c) deslocar o cursor **20** linhas para baixo ou para cima (associadas a PF2 e PF3)

(d) inserir um TAB (associada a CTRL-I)

9. Determinar um processo de importar um bloco de um texto para outro sem usar nenhum ficheiro intermédio.

10. Enviar uma mensagem ao utilizador *jbb* a comunicar o andamento deste curso.

11. Usando como base o ficheiro "**form**", construir (usando *vi*) uma descrição do grupo de trabalho e enviá-la ao utilizador "**jj**" por *mail*.

12. Construir um comando que determine quantas mensagens existem na caixa do correio.

13. Construir um comando que liste os *logname* dos utilizadores que nos enviaram mensagens.

14. Definir *alias* (abreviaturas) para

    (a) mostrar apenas as subdirectorias da actual

    (b) mostrar apenas os ficheiros da directoria actual

    (c) dizer (apenas) as horas

15. Programando em C-Shell, escreva um comando ("script") para enviar correio para todos os Utilizadores de *"UserName"* base **fund** (os quais estão directamente dependurados em */usr1*).
    O comando terá o nome **mall** e levará, obrigatoriamente, como argumento o nome do ficheiro que contém a mensagem a enviar a todos.

16. Desenvolva uma "script" para listar na consola o espaço gasto em disco por cada um dos Utilizadores alojados em */usr1* cujo *UserName* concorde com a palavra introduzida interactivamente (via consola) pelo utilizador dessa "script".
    Para os restantes Utilizadores deve apenas ser escrito o seu nome.

17. Crie um novo comando, usando as facilidades de programação da C-Shell, para guiar o utilizador na escrita de "scripts". O comando terá o nome **mkscript** e receberá como parametro obrigatório o nome da "script" a criar.
    Uma vez invocado valida a existência desse argumento e pede ao utilizador que introduza uma frase concisa que descreva o objectivo (possivelmemte a forma de usar) da "script" a criar. Então começa por acrescentar ao ficheiro *bdscripts* (possivelmente já existente) uma linha com o seguinte formato

nome_script ":" descrição

Depois, cria um novo ficheiro como o nome passado como parametro inicializando com os quatro seguintes comentários:

```
#!/usr/ucb/csh
#nome da script ":" descricao fornecida
#"criado em " data actual
#"pelo utilizador " nome do utilizador
```

De seguida, invoca o editor *vi* para o utilizador continuar a editar a "script" que pretende desenvolver e, no fim (após sair do editor), muda automaticamente o modo do novo ficheiro dando-lhe permissão de execução para todos.
Acrescente ao seu programa a possibilidade do texto descritivo ser passado, opcionalmente, como parametro do comando, devendo nesse caso aparecer depois do "switch" **-d**.

18. Crie um comando, em C-Shell, com a síntaxe:

    proc pattern+

    o qual deve procurar cada um dos padrões passados como parametros nos ficheiros cujos nomes estão contidos no ficheiro *'pgm.files'* (este terá de existir forçosamente). Ao encontrar deve ser indicado o nome do ficheiro onde ocorre e o nmero da respectiva linha.

19. Escreva um programa, em C-Shell, para editar comandos de acordo com a síntaxe do *sed*. Esse programa mostra a *história* (os ltimos comandos executados), pede que identifique um desses comandos (pelo seu nmero), pede que introduza a correcção a fazer (usando as facilidades do editor *sed*) e depois de alterar o comando seleccionado segundo o esquema proposto, procede à sua execução.

20. Desenvolva um comando em C-Shell:

    inv txt

    que inverta o texto cujo nome é dado como argumento, isto é, que escreva num outro ficheiro de nome igual ao primeiro mas com a extensão **inv** as linhas do primeiro pela ordem inversa (da ltima para a primeira).

21. Pretende-se fazer um programa para calcular as notas de um *teste americano*".

    Para isso existem 3 (**três**) ficheiros:

(a) **nomes** que contém uma linha para cada aluno com a seguinte informação :

curso (até quatro caracteres)

regime (até oito caracteres)

nmero (até 4 dígitos)

nome (o resto da linha)

(b) **testes** que contém uma linha para cada aluno que tenha ido ao teste com a seguinte informação :

nmero (até 4 dígitos)

versão (A B ou C)

respostas (os dez campos seguintes)

(c) **correccao** que contém uma linha para cada pergunta do teste com a seguinte informação :

nmero da pergunta (de 1 até 10)

peso da pergunta (real de 0 até 1)

cotação de cada resposta sequência de 10 nmeros de 0 a 100

As várias versões do teste são obtidas por permutação da ordem das perguntas. Na tabela seguinte mostra-se como é que se podem converter as várias versões para a versão de base (por convenção a versão A).

Pergunta na versão B	Equivalente na versão A	Pergunta na versão C	Equivalente na versão A
1	2	1	7
2	3	2	9
3	4	3	10
4	1	4	8
5	9	5	6
6	10	6	2
7	8	7	3
8	7	8	4
9	6	9	5
10	5	10	1

O programa deve escrever em *stdout* uma linha para cada aluno contendo

- curso
- nmero
- nome
- classificação (ou FALTOU se não tiver ido ao teste)

Apresenta-se de seguida um exemplo de utilização desta script.

```
$ cat nomes

LESI xxxxx 1234 Pancracio da Silva
LMCC xxxxx 2345 Manuel Joaquim
LFQ  xxxxx 3456 Antonio Aleixo
LEM  yyyyy 4567 Agostinho da Silva
LMCC zzzzz 5678 Joao Manuel da Silva Rocha
LESI ttttt 6789 Antonio Passos Dias Aguiar Mota

$ cat testes

1234 A 1 2 3 4 5 6 7 8 9 10
2345 C 2 3 4 5 6 7 8 9 10 10
3456 B 1 1 3 3 5 5 7 7 9 9
4567 A 1 5 4 3 7 9 8 7 7 10
7890 C 3 2 1 6 5 4 8 7 6 9

$ cat correccao

1  0.1  100   0   0   0   0   0   0   0   0   0
2  0.1    0 100   0   0   0   0   0   0   0   0
3  0.1    0   0 100   0   0   0   0   0   0   0
4  0.1    0   0   0 100   0   0   0   0   0   0
5  0.1    0   0   0   0 100   0   0   0   0   0
6  0.1    0   0   0   0   0 100   0   0   0   0
7  0.1    0   0   0   0   0   0 100   0   0   0
8  0.1    0   0   0   0   0   0   0 100   0   0
9  0.1    0   0   0   0   0   0   0   0 100   0
10 0.1    0   0   0   0   0   0   0   0   0 100

$ corr

LESI 1234 Pancracio da Silva ....................    100
LMCC 2345 Manuel Joaquim ........................      0
 LFQ 3456 Antonio Aleixo ........................     30
 LEM 4567 Agostinho da Silva ....................     20
LMCC 5678 Joao Manuel da Silva Rocha ............. FALTOU
LESI 6789 Antonio Passos Dias Aguiar Mota ........ FALTOU
???? 7890 .......................................      0
```

22. Desenvolva um comando **'mkteste'** que crie um ficheiro LaTeX a partir dum *esqueleto base do teste* (ficheiro cujo nome é passado como 1º

parâmetro) e duma *base de dados de perguntas* (o seu nome será o 2º parâmetro do comando). O resultado deve ficar num ficheiro de nome base igual ao 1º parametro mas com a extensão ".pre".
O texto base contém o cabeçalho, eventualmente outras linhas diversas, que devem ser copiadas integralmente para a saida e contém uma linha (ou mais do que uma) da forma:

```
\questoes: label1 label2 .... labeln
```

linha essa que não deve ser copiada para a saida, mas que têm os identificadores ("labels") das questões existentes na base de dados que devem ser inseridas no teste pela ordem implícita.
O texto de cada questão deve ser extraído integralmente da BD mas deve ser intercalado entre as linhas

```
\begin{Questao} e \end{Questao}
```

Quanto à base de dados, ela tem textos das perguntas, sendo cada texto iniciado por uma linha da forma:

```
@labeli
```

onde "labeli" é o identificador da pergunta, usado no texto base para identificar cada questão a inserir.

Para clarificar o objectivo deste exercício, listam-se a seguir: um texto base (que define a estrutura do teste a gerar), de nome *teste1*; uma base de dados de perguntas, de nome *bdquestoes*; e o texto do teste, gerado finalmente sob o nome *teste1.pre*.

```
%----------teste1
cabeca 1
cabeca 2
cabeca 3
cabeca 4
\questoes: q1 q3 q5 q2
fim

%----------bdquestoes
@q1
 linha1 q1
 linha2 q1
@q2
 linha1 q2
 linha2 q2
```

```
@q3
 linha1 q3
 linha2 q3
@q4
 linha1 q4
 linha2 q4
@q5
 linha1 q5
 linha2 q5
@q6
 linha1 q6
 linha2 q6
@q7
 linha1 q7
 linha2 q7

%----------teste1.pre
cabeca 1
cabeca 2
cabeca 3
cabeca 4
\begin{Questao}
 linha1 q1
 linha2 q1
\end{Questao}
\begin{Questao}
 linha1 q3
 linha2 q3
\end{Questao}
\begin{Questao}
 linha1 q5
 linha2 q5
\end{Questao}
\begin{Questao}
 linha1 q2
 linha2 q2
\end{Questao}
fim
```

23. O objectivo deste exercício é criar um comando para processar *documentos anotados*. Um texto anotado é um texto qualquer que tem exporadicamente comentários (que não devem ser imprimidos na versão final) intercalados entre "{" e "}", imediatamente precedidos da palavra reservada `"\message"`.

O processamento pretendido consiste, exactamente, em retirar todo o comentário (desde a citada palavra reservada até à ltima chaveta) substituindo-o pela nota (texto adicional) que se encontra num ficheiro auxiliar, numa posição igual ao nmero de ocorrência desse comentário, isto é, se fôr o primeiro deve ir buscar-se a primeira nota, se fôr o quarto comentário deverá extrair-se a quarta nota.
As notas estão escritas num ficheiro de nome *notas.tex*, podem conter várias linhas e separam-se umas das outras por uma linha em branco.

Segue-se um exemplo dum texto anotado (*base.tex*), um ficheiro de notas (*notas.tex*) e do resultado final (*novo.tex*), obtido com a invocação:

procmsg < base.tex > novo.tex

```
$ cat base.tex
Isto e uma experiencia \message{comentario 1}
do processador de mensagens, isto e, anotacoes
\message{ve la como escreves}
para fazer o upgrade do TeX.
Duvido \message{e preciso fe, esperanca e caridade} que isto
funcione a primeira.
Ate ja.

$ cat notas.tex
ESTA SERA A NOTA NUMERO 1 DA EXPERIENCIA.

E AGORA AQUI VAI A SEGUNDA NOTA

ESTE E O COMENTARIO A MENSAGEM 3.
SERA QUE RESULTOU ?

$ cat novo.tex
Isto e uma experiencia ESTA SERA A NOTA NUMERO 1 DA EXPERIENCIA.
do processador de mensagens, isto e, anotacoes
E AGORA AQUI VAI A SEGUNDA NOTA
para fazer o upgrade do TeX.
Duvido ESTE E O COMENTARIO A MENSAGEM 3.
SERA QUE RESULTOU ? que isto
funcione a primeira.
Ate ja.
```

24. **(awk: comando de enquadrar)**

Construir um comando para enquadrar um texto. Exemplo de utilização do referido comando (quadro) e respectivo resultado:

```
% cal | quadro

+-------------------------------------------------------------+
|    March 1991                                               |
|  S  M Tu  W Th  F  S                                        |
|                 1  2                                        |
|  3  4  5  6  7  8  9                                        |
| 10 11 12 13 14 15 16                                        |
| 17 18 19 20 21 22 23                                        |
| 24 25 26 27 28 29 30                                        |
| 31                                                          |
+-------------------------------------------------------------+
```

Modificações propostas:

(a) Definir um comando "vi" que enquadre a linha actual.

(b) redefinir o comando quadro de modo a fazer um quadro de largura "m" sendo "m" o primeiro parametro.

25. **(awk: usando arrays associativos)**

    Construir um programa que determine o nmero de ocorrências de cada palavra num (ou mais) textos, imprimindo-as ordenadamente.

    Considera-se pertencente a palavra as letras, acentos e ifen; qualquer sequência de outros caracteres funcionará com separador.

26. **(csh: menus de comandos)**
    Criar uma script para selecção interactiva de um ficheiro e escolha também interactiva de uma acção a realizar sobre ele.

    Exemplo de utilização dessa script:

```
--------------------------------------------------------------
               Selecionar um ficheiro
--------------------------------------------------------------
     1  antecedentes
     2  backup
     3  c
     4  carv
     5  comandos
     6  fundexec
     7  fundput
     8  jaa
```

```
     9  quadro.awk
------------------------------------------------------------
Introduzir No. do Ficheiro: 3

------------------------------------------------------------
E(ditar)     A(pagar)      S(air)      V(isualizar)
------------------------------------------------------------
A

------------------------------------------------------------
               Selecionar um ficheiro
------------------------------------------------------------
     1  antecedentes
     2  backup
     3  carv
     4  comandos
     5  fundexec
     6  fundput
     7  jaa
     8  quadro.awk
------------------------------------------------------------
Introduzir No. do Ficheiro: 3

------------------------------------------------------------
E(ditar)     A(pagar)      S(air)      V(isualizar)
------------------------------------------------------------
S
```

Modificação proposta:

- reescrever a forma de mostrar dos ficheiros (`ls | nl`) de modo a que os ficheiros aparecem escritos em 3 colunas (o que seria adequado quando se pretendesse seleccionar um entre muitos ficheiros) (sujestões : ver o comando "pr")

27. **(make: completar uma makefile)**

Acerca de um programa chamado SQL, construiu-se uma makefile incompleta que se mostra abaixo. Analise-a e complete-a segundo os comentários nela incluidos.

```
----------------------------------------------------------------------
#descricao que se achar necessaria
```

```
pergunta:
#    Por defeito esta makefile indicaria a lista de "coisas" que podem
# ser construidas com ela (sem fazer nenhuma)

DOC : sql
# ver tambem o exercicio seguinte como forma de melhorar esta entrada
        pr sql.y sql.l sqlf.c makefile > DOC

sql : y.tab.c sqlf.c
        cc -o sql y.tab.c -ly -ll

y.tab.c : sql.y lex.yy.c
        yacc sql.y

lex.yy.c : sql.l
        lex sql.l

instala :
#  definir uma macro BIN para programas executaveis
#  copiar o sql para a essa directoria

sql.tar:
# Armazenar os ficheiros indispensaveis numa directoria chamada SQL e
# colocar tambem um ficheiro README com a data actual e
# lista de ficheiros copiados e
# fazer o empacotamento dessa directoria em sql.tar
#
# depois apagar a directoria SQL

clean:
# os seguintes ficheiros s~ao lixo: lex.yy.c y.tab.c *.o
----------------------------------------------------------------------
```

28. **(awk: documentador automático de listagens)**

Pretende-se construir um criador automático de documentação para programas escritos em C. Esse "documentador" deverá:

- criar um índice de funções existentes nos ficheiros (ver ctags –x) com o formado "nomeFuncao nomeFicheiro Linha"
- criar uma lista de inclusões (para cada ficheiro, apresentar a lista de ficheiros que ele "manda incluir" isto é "#include ...")
- Para cada ficheiro criar um capítulo com os comentários "especiais"

(os comentários especiais são da forma `/*.      comantario .*/`, podendo ocupar mais que uma linha)

Exemplo do ficheiro resultado:

```
Cap. 0     Indice de Funcoes
------------------------------------------
Nome        Ficheiro         Linha
------------------------------------------
cons        sql.y            97
help        sqlf.c           22
itoa        sql.y            76
makelista   sql.y            108
makelistaauxsql.y            93
mkstr       sql.y            83
project     sqlf.c           32
prompt      sql.y            72
ren         sqlf.c           4
select      sqlf.c           45
show        sqlf.c           16
strsave     sql.y            112
------------------------------------------
------------------------------------------
sql.y  inclui:
------------------------------------------
        stdio.h
        string.h
        sqlf.c
        lex.yy.c
------------------------------------------
sqlf.c  inclui:
------------------------------------------
------------------------------------------
sql.l  inclui:
------------------------------------------


Cap. 1 sql.y
------------------------------------------
 1. Gramatica geral :

        SQL     :  expressao

        expressao : Exp  ';'  expressao
```

```
                  | Help  ';'  expressao
                  | END ';'

        Help    : HELP ID
                | HELP

        Exp     : rel '=' Exp
                | SELECT atributos FROM rel WHERE cond
                | SELECT atributos FROM rel

2. Funcoes internas
 cons : elemento , lista -> lista
 makelista : -> lista


Cap. 2 sqlf.c
------------------------------------------
 ren: relacao , relacao
         Muda o nome de uma relacao

 show: relacao
         Mostra o conteudo de uma tabela
         Obs: precisa de ser melhorado ...

 project: listaDeCampos, relacao -> relacao
         funcao projecao: seleciona algumas das colunas



Cap. 3 sql.l
------------------------------------------
 Aspectos Lexicos
    . As palavras chaves podem ser abreviadas
    . As condicoes tem de ser incluidas entre parentesis
```