Resoluções

1. (a) `mkdir X ; cp * X`
   (b) `rm -i X/*`
   (c) `rmdir X`
2. - `ls -l | tail +1 | wc -l`
   - `ls | wc -w`
3. - `head -25 fich | tail +10`
   - `tail +10 fich | head -16`
4. `(head -9 fich ; tail +26  ) > aux`
5. (a) `cut -d: -f5 /etc/passwd | sort`
   (b) `sort -d: +0d -1 /etc/passwd | cut -d: -f5`
   (c) `sort -d: +1n -2 /etc/passwd | cut -d: -f5`
6. `mail`
7. (a)
      i. `:%s/^[^@].*$//`
         `:global/^$/d`
      ii. `grep "^@" fich`

   (b)
      i. para cada linha do texto deve-se fazer
         A. posicionar no início da linha – `0`
         B. apagar o texto até à chaveta – `dt`
         C. inserir a palavra \item – `i\item`ESC
         D. posicionar no ltimo caracter (,) – `$`
         E. mudar o caracter para } – `r}`

         NOTA: o segundo e terceiro passos podiam ser substituidos por um nico
         `ct{\item`ESC
      ii. Para definir o comando adequado devemos usar o comando **map** associado à sequência de toques de tecla apropriada (ver caso anterior)
         `:map #1 0dt{i\item^[$r}`
         ou `:map #1 0ct{\item^[$r}`
         NOTA: a tecla que fica definida para executar esta acção é a tecla PF1.
         O caracter `^[` que significa ESC obtem-se fazendo CTRL-V seguido de ESC. Alias, sempre que pretendemos proteger a introdução de um caracter devemos proceder dessa forma.

iii. `:1,$s/^@.*\({.*\),/\\item\1}/`

8. devem aparecer no ficheiro *.exrc* as seguintes linhas:
   (a) `map ^E $`
   (b) `map ^R i^M^[`
   (c) `map #2 20j`
       `map #3 20k`
   (d) `map ^I i ^[`

9. Este processo passa pelo uso de "buffers". Imagine-se que pretendemos importar para o ficheiro *fich2* o bloco de texto do ficheiro *fich1* desde a linha 10 à linha 20. Os passos a dar serão:
   (a) `vi fich1`
   (b) `:10,20y a`
   (c) `:e fich2`
   (d) posicionar o cursor antes do local onde pretendemos inserir o bloco
   (e) `"ap`

10.
```
mail jbb
Este curso esta a decorrer muito bem.
.
```

11.

12. A resolução deste exercício e do próximo baseiam-se no conhecimento de que as mensagens recebidas se encontram no ficheiro `$MAIL` e que o início de cada mensagem é detectado por uma linha que começa com a palavra `From` e que tem a forma `From jbb Thu Mar 21 14:32 MST 1991` indicando quem enviou a mensagem e quando.

    Para sabermos quantas mensagens existem basta contar quantas linhas desta forma existem, isto é,
    `grep -c "^From" $MAIL` ou então
    `grep "^From" $MAIL | wc -l`

13. `grep "^From" $MAIL | cut -d" " -f2`

14. (a) `alias ddir 'ls -la \!* | grep "^d"'`
    (b) `alias ddir 'ls -la \!* | grep -v "^d"'`
    (c) `alias hora 'date | cut -d" " -f4'`

15.
```
#!/usr/ucb/csh
# mall : enviar o ficheiro parametro por mail a todos os
#        utilizadores fund*
if ($#argv != 1) then           # numero de parametros errado
   echo "$0 ficheiro : envia ficheiro por mail a fund*"
   exit 1
else if !(-e $1) then
        echo "ficheiro -- $1 -- inexistente"
        exit 1
endif
mail ‘ls /usr1/fund*‘ $1
```

16.
```
#!/usr/ucb/csh
# gasto : mostrar o espaco ocupado pelos utilizadores
#         sob a directoria \usr1 que concordam com um pattern
if ($#argv != 0) then           # numero de parametros errado
   echo "$0 deve ser invocado sem parametros"
   exit 1
endif
echo -n "Utilizadores : "
set pat=$<
foreach f ‘ls -d /usr1/fund*‘
    echo "Utilizador $f"
    if ($f =~ pat)    du /usr1/$f
end
```

17.
```
#!/usr/ucb/csh
# mkscript : construtor de scripts
if ($#argv == 1) then
   echo -n "Comentario ‘a script $1 : "
   set coment = $<
else if ($#argv > 1 && $argv[2] == "-d") then
     set coment = $argv[3-]
else
     echo "$0 nome [ -d texto ]"
     exit 1
endif
if (-e $1) then
   echo -n "Ficheiro $1 existente. Pretende reescreve-lo? (S/N) "
   switch ($<)
   case [Ss]:
        echo "conteudo do ficheiro $1 destruido \!\!\!"
        set temp = /tmp/bdscripts$$
```

```
            cp bdscripts $temp
            grep -v "^$1" $temp >! bdscripts
            rm $temp
            breaksw
       default:
            exit 1
       endsw
    endif
    echo "$1 : $coment" >>! bdscripts
    echo '#\!usr/ucb/csh' >! $1
    echo "# $1 : $coment" >> $1
    echo -n "# Criado em " >> $1
    date | cut -d" " -f0-3,5 >> $1
    echo -n "# pelo utilizador " >> $1
    logname >> $1
    vi +G $1
    chmod +x $1
```

18.

19.

20.

```
#!/usr/ucb/csh
if ($#argv != 1) then
   echo "$0 fich : inverte o ficheiro fich"
   exit 1
endif
nl -ba $1 | sort -nr | cut -f2-
```

21. Apresenta-se de seguida o texto da script. As linhas deste texto estão numeradas para que depois se possam fazer algumas considerações .

```
 1   #!/usr/ucb/csh
 2   unalias rm
 3   awk '{ print "#N    " $0 }' nomes > NOMES
 4   awk '{ print "#C    " $0 }' correccao > CORRECCAO
 5   set fich=/tmp/fich$$
 6   cat > $fich << end
 7   \$2 == "A" { print "#T ",\$1,\$3,\$4,\$5,\$6,\$7,\$8,\
 8 \$9,\$10,\$11,\$12 }
 9   \$2 == "B" { print "#T ",\$1,\$4,\$5,\$6,\$3,\$11,\
10 \$12,\$10,\$9,\$8,\$7 }
11   \$2 == "C" { print "#T ",\$1,\$9,\$11,\$12,\$10,\$8,\
```

```
12 \$4,\$5,\$6,\$7,\$3 }
13   end
14   awk -f $fich testes > TESTES
15   cat >! $fich << end
16   \$1 == "#C" { for (i=1; (i < 11); i++)
17           correccao [\$2 * 10 + i] = \$(i+3);
18           correccao [\$2 * 10] = \$3;
19         }
20   \$1 == "#T" { nota = 0.0;
21                  for (i=1; (i<11); i++)
22           nota = nota +(correccao [i*10] * \
23 correccao [i*10 + \$(i+2)]);
24           aluno [\$2] = nota;
25         }
26   \$1 == "#N" { printf ("%4s %4d ",\$2, \$4);
27           l=0;
28           for (i=5; (i<= NF); i++)
29               { l = l + 1 +length (\$i);
30 printf ("%s ", \$i); }
31           for (; (l<40);  l++) printf (".");
32           if (\$4 in aluno)
33            { printf (" %6d\n", aluno [\$4]);
34                     delete aluno [\$4] ; }
35           else printf (" FALTOU \n");
36         }
37   END { for (i in aluno)
38                        { printf ("???? %4d ", i);
39                  for (l=0; (l<40);  l++) printf (".");
40                  printf (" %6d\n", aluno [i]);
41                }
42      }
43   end
44   cat CORRECCAO TESTES NOMES | awk -f $fich
45   rm $fich CORRECCAO TESTES NOMES
```

O efeito pertendido é obtido usando dois programas “awk”: um para converter as versões B e C para a versão A - linhas 7 até 12 e o segundo para calcular as notas de cada aluno.

Foram usadas algumas facilidades da *C-shell* para colocar toda a informação necessária num nico ficheiro. Destas salienta-se a construção << que permite que o *stdin* de um comando seja descrito dentro da própria *script*. Vemos então que os textos que se encontram da linha 7 até à linha 12 e da linha 16 até à 42 deveriam estar num outro ficheiro - são os textos de um programa “awk”. Note-se ainda que dentro destas linhas aparecem os símbolos $ protegidos. Isto porque dentro de uma *script* o símbolo $ tem um significado especial (é um metacaracter).

As três primeiras chamadas ao awk (linhas 3, 4 e 14 servem para construir, a partir dos ficheiros **nomes**, **correccao** e **testes**, os ficheiros NOMES, CORRECCAO e TESTES nos quais cada linha é perfixada com as strings `#N`, `#C` e `#T` respectivamente.

Esta operação poderia ser evitada como se verá mais tarde.

A ltima invocação do awk (linha 44) serve para calcular as notas de cada aluno.

As três condições que existem são de teste do 1º campo (prefixo colocado na fase anterior). Este efeito poderia ser igualmente obtido pelo uso da variável FILENAME. Daí que a primeira parte desta *script* podia ser evitada.

Vemos então que quando se está a ler o ficheiro CORRECCAO (linhas 16 a 19) preenche-se um array correccao com a informação contida no ficheiro.

Depois, para cada linha do ficheiro TESTES (linhas 20 a 25) calcula-se a nota do aluno e guarda-se no array aluno (indexado pelo nmero do aluno).

Finalmente, à medida que se vão lendo os nomes dos alunos (linhas 26 a 36), imprime-se o seu nmero, curso, nome e a nota (caso tenha ido ao teste, isto é, pertença ao array aluno) ou a palavra FALTOU.

Existe ainda um ciclo (linha 31) cujo objectivo é aparecerem as notas alinhadas.

Para cada aluno cuja nota é escrita, elimina-se o seu nmero do array aluno (linha 34), para que depois (linhas 37 a 42) se possam escrever as notas dos alunos que, apesar de não constarem no ficheiro NOMES, existem no ficheiro TESTES.

22. A resolução do problema passa pela criação duma ”script”, em c-shell, que testa a correcta invocação, isto é, a chamada do comando com os dois parâmetros necessários, e invoca o programa ”*awk*” para proceder à primeira fase do processamento do texto base. Durante a execução desse programa ”*awk*”, cada vez que se encontra um identificador de pergunta (uma ”label”), invoca-se novamente o ”*awk*” (com um novo conjunto de comandos) de modo a extrair a questão identificada da base de dados. Apresentam-se de seguida o texto dos três programas: a ”*script*” e os dois *ficheiros de comandos ”awk”*.
O comando mkteste:

```
#!/usr/ucb/csh
#Comando: mkteste
```

```
#
if ($#argv != 2) then
   echo "Utilizacao: mkteste textobase bdquestoes"
else
   awk -f mktfase1.awk bd=$2 $1
end
```

Texto do programa awk correspondente à primeira fase do processamento:

```
#
#Programa: mktfase1.awk
#
BEGIN        {saida = FILENAME ".pre";}
/questoes:/ {for(i=2; i<=NF; i++)
             {system("awk -f mktfase2.awk padrao="$i" \
                     obj="saida " " bd)};
             next }
            {print >> saida }
```

Texto do programa awk correspondente à segunda fase do processamento:

```
#
#Programa: mktfase2.awk
#
BEGIN        {RS="@";FS="\n"}
$1 ~ padrao  {print "\\begin{Questao}" >>obj;
              for(i=2;i<=NF;i++) print $i >>obj;
              print "\\end{Questao}" >>obj }
```

23. Para resolver o problema foi necessário usar o "*lex*" —tendo-se para o efeito especificado o processamento a realizar no ficheiro *procmsg.lex*— e o "*awk*", o qual é invocado com os comandos escritos em *procmsg.awk*.
O primeiro procura no texto base um bloco começado em `"\message"` e terminado em "}", elimina-o e invoca o "*awk*" para ir buscar ao ficheiro *notas* a nota correspondente. Todos os restantes caracteres fora do referido contexto são copiados textualmente para a saída.
Ambos os processadores vão enviando a saida para "*stdout*", devendo, pois, o utilizador redireccioná-la para o ficheiro onde pretender guardar o texto final. De igual forma, deve redireccionar-se a *stdin* para que o programa gerado pelo "*lex*" leia o texto a processar do ficheiro conveniente.
Listam-se abaixo os programas usados:

```
%
%Programa: mprocmsg.lex
```

```
%
%{
#include <stdio.h>
int cntmsg = 0;
char cmd[256];
%}
%%
\\message\{[^\}]*\}     { auxiliar(++cntmsg); }
.|\n                   ECHO;
%%
main()
{
        yylex();
}

auxiliar(n)
int n;
{ fflush(stdout);
        sprintf(cmd,"awk -f procmsg.awk cnt=%d notas.tex\n",n);
system(cmd);
}
```

Terminada a escrita deste ficheiro é preciso invocar o "*lex*" e o compilador de C para obter um programa —*procmsg*— executável, devendo para isso executar-se os comandos seguintes:

```
lex procmsg.lex
cc -o procmsg lex.yy.c -ll
```

Segue-se o texto do programa "*awk*" invocado

```
#
#Programa: mprocmsg.awk
#
BEGIN  { RS=""; FS="\n" }
NR == cnt { print ;
           exit }
```

24. `% cat quadro`

```
    awk -f quadro.awk $*
```

`% cat quadro.awk`

```
BEGIN \{print "+--------------------------------------------------------------+"\}
      \{printf ("| %-60s |{\bs}n",$0)\}
  END \{print "+--------------------------------------------------------------+"\}
```

Ver manual de "printf" para pormenores de formatação.

- `:map q :.! quadro^m`
-

25. `awk -f ocorrenc.awk texto ...`

onde ocorrenc.awk seria

```
  BEGIN \{FS= "[^a-zA-Z_~`'^-]+";\}
        \{for (i=1; i<=NF; i++) Oco[$i]++ \}
  END   \{for (x in Oco)
             printf("%-15s - %s{\bs}n",x,Oco[x]) | "sort" \}
```

aplicando este programa a ele próprio daria:

```
BEGIN           - 1
Considera-se    - 1
Construir       - 1
END             - 1
Exerc'icio      - 1
FS              - 1
NF              - 1
Oco             - 3
^a-zA-Z_~`'^-   - 1
a               - 2
acentos         - 1
aplicando       - 1
arrays          - 1
as              - 1
associativos    - 1
awk             - 4
cada            - 1
caracteres      - 1
com             - 1
daria           - 1
de              - 3
determine       - 1
e               - 1
ele             - 1
```

26. O problema proposto pode ser resolvido, por exemplo, com a seguinte script:

```
#!/usr/ucb/csh
#
while (1)
   echo ""
   echo "____________________________________________________________"
   echo "              Selecionar um ficheiro"
   echo "____________________________________________________________"
   # Apresentar no ecran uma lista numerada dos ficheiros existentes
   ls | nl

   # Ler o numero de um ficheiro
   echo "____________________________________________________________"
   echo -n "Introduzir No. do Ficheiro: "
   set Nofich = $<

   # colocar na variavel X o respectivo nome
   set ficheiros = `ls `
   set X = $ficheiros[$Nofich]

   echo ""
   echo "____________________________________________________________"
   echo "E(ditar)    A(pagar)     S(air)     V(isualizar)"
   echo "____________________________________________________________"
   switch ($<)
      case "[Ee]":
        vi $X
        breaksw
      case "[Aa]":
        rm -i $X
        breaksw
      case "[Vv]":
        more $X
        breaksw
      case "[Ss]":
        exit 0
        breaksw
      endsw
end
```

27.

28. O efeito pretendido pode ser obtido com a script e programas ”awk” que a seguir se listam.

Texto da *script* “doc”

```
#!/usr/ucb/csh
# doc: cria documentacao sum'aria a partir de listagens de programas
#    escritos em C

# criacao do indice das funcoes:
ctags -x $* |awk -f doc1.awk >documentacao

# criacao do indice das inclusoes:
awk -f doc2.awk $* >> documentacao

# Obtensao dos comentarios selecionados (ie. contidos entre /*. e .*/)
awk -f doc3.awk $* >> documentacao
```

Texto do programa doc1.awk

```
BEGIN \{ print "Cap. 0     Indice de Funcoes ";
        risco();
        printf ("Nome        Ficheiro         Linha{\bs}n");risco()\}
      \{ printf ("%-12s%-18s%s{\bs}n",$1,$3,$2)\}
END   \{ risco()\}
function risco() \{print "-------------------------------------------"\}
```

Texto do programa doc2.awk

```
#********* tratamento das inclusoes ********************
(FNR==1)          \{risco();cabec()risco();\}
($1=="#include")  \{gsub(/["<>]/,"",$2);print "{\bs}t" $2\}

function cabec() \{print FILENAME, " inclui:"\}
function risco() \{print "-------------------------------------------"\}
```

Texto do programa doc3.awk

```
#********* tratamento de comentarios selecionados *******
(FNR==1)            \{ cabec()\}
$1=="/*.",$NF==".*/" \{ gsub(/{\bs}.{\bs}*{\bs}//,"",$NF);
                       gsub(/{\bs}/{\bs}*{\bs}./,"",$1);
                       print $0\}
function risco() \{print "-------------------------------------------"\}
function cabec() \{print "{\bs}n{\bs}nCap.", ++I , FILENAME ;
                  risco();\}
```