Exercícios Práticos

Sistemas Operativos

(Lic. em Matemática)

Nota: Para todos os exercícios de C deve usar apenas chamadas ao Sistema Unix, i.e. não usar funções como: sleep, fopen, etc..

1. Pretende-se fazer um menu para uso de alunos que estao a aprender a programar em C.

   Esse menu deve permitir as seguintes opcoes:

   (a) Mostrar os ficheiros com sufixo ".c" existentes na directoria, numerando-os;

   (b) Seleccionar um destes ficheiros usando o seu número;

   (c) Compilar o programa selecionado usando o *more(1)* para mostrar os erros que o compilador escreve no standard error (use uma pipe);

   (d) Executar o programa compilado;

   (e) Imprimir o programa selecionado depois de formatado pelo "embelezador" *cb(1)* e paginado com o *pr(1)* (use pipes);

   (f) No caso de o utilizador premir ⟨CTRL⟩ C o menu deve pedir para confirmar que o utilizador pretendo mesmo terminar.

   Escreva:

   (a) Um shellscript que implemente este menu.

   (b) A mesma coisa usando a linguagem C.

2. Escreva o código de 2 programas em que o primeiro gera 100000 números aleatórios e o segundo os separa em 2 ficheiros Pares e Impares, de acordo com cada um dos casos seguintes:

   (a) O segundo programa so é executado depois do primeiro terminar;

   (b) Os dois programas são executados simultaneamente na mesma máquina. O programa separador deve de 2 em 2 segundos dar a indicação de quantos elementos de cada tipo ja separou;

   Nota: Para cada uma das alíneas deve escolher o método mais adequado à situação justificando abreviadamente a razão da escolha;

3. Faça um programa em C que verifique a conjectura de Goldbach (qualquer numero par é a soma de 2 números primos).

   Para resolver este problema é necessário obter uma lista de números primos. Esta tarefa deve ser realizada por um processo separado.

   (a) O programa deve verificar a conjectura de Goldbach até um número n dado pelo primeiro pârametro. Depois de verificada a conjectura, deve dar a oportunidade ao utilizador de dar um novo (e maior) n e verificar a conjectura até esse novo número.

   (b) O processo que cria a lista de primos deve deixar uma cópia num ficheiro (em binário). Um segundo programa deve usar esse ficheiro para, dado um número, escrever uma de três mensagens:

      - n é primo
      - n não é primo
      - não sei se n é primo

      Este programa deve fazer uma pesquisa binária no ficheiro de primos.

4. Existe um conjunto de directórios todos dentro do mesmo directório que contém ficheiros possivelmente repetidos. Como exemplo no directório

   ```
   ~rps/aulas/so/94_95/exemplo_teste
   ```

   foram criados os directórios dir1, dir2, dir3 e treta que contém os seguintes ficheiros:

   ```
   dir1:
   chicken endianness hitler law

   dir2:
   chicken endianness

   dir3:
   hitler law

   treta:
   chicken hitler
   ```

   Pretende-se substituir eventuais repetições por links. Sabe-se que 2 ficheiros só são iguais se tiverem o mesmo nome, embora possam ter o mesmo nome e serem diferentes.

   (a) Escreva um shellscript que resolva este problema.

   (b) No fim do script deve ser escrita uma mensagem indicando o espaco de disco poupado.

5. Faça um programa em shell para gerir uma agenda. Esse programa deve ter um menu com (pelo menos) as seguintes opções.

   (a) listar as pessoas existentes, e respectivos números de telefone, por ordem alfabética.

   (b) juntar um novo nome, com o respectivo número de telefone.

   (c) procurar o telefone de uma pessoa.

   (d) procurar o nome do proprietário de um telefone.

   (e) retirar um nome.

6. Num sistema Unix cada utilizador tem um limite para o número de processos que pode lançar. Escreva um programa que determine experimentalmente esse número (assumindo que quando é executado o utilizador só tem a correr a shell), escrevendo-o no stdout.

   O programa não deve deixar "zombies".

7. Escreva um menu em shell que tenha as seguintes opções:

   (a) Seleccionar um ficheiro

   (b) Editar o ficheiro corrente

   (c) Mostrar o ficheiro no ecran

   (d) Seleccionar a impressora a usar

   (e) Imprimir o ficheiro

   (f) Terminar

8. Existe uma sub-árvore de directórios que contém apenas subdirectórios e ficheiros normais (ou seja, não existem devices, links simbólicos, named pipes, etc ...). Um directório considera-se vazio se não contiver ficheiros normais (pode ter subdirectórios).

   Faça um programa em shell que dado o nome do directório topo da sub-árvore escreva uma lista de directórios vazios ou se for chamado com a opção -r escreva a lista de directórios não vazios.

9. Em Unix existe sempre mais de uma maneira de resolver um problema. Como exemplo responda ao exercício anterior de uma maneira (quase) completamente diferente.

10. Pretende-se medir o tempo de mudança de contexto ("context switching time") de várias implementações de Unix. Para isso pretende-se um programa em C que:

    (a) lance um processo filho;

    (b) repita n vezes (em que n é um parâmetro do programa):

        i. um dos processos espera pelo outro;

ii. o outro processo assinala ao primeiro que pode continuar;
iii. os dois processos trocam de papeis.

Pode usar qualquer um dos meios de comunicação entre processos disponível em Unix.

11. Faça um programa que : dada uma lista de nomes, para cada um: se o nome for de um directório mostre o espaco ocupado (com du -s) caso contrário mostre toda a informação sobre ele (com ls -l)

    Implemente este programa em :

    (a) shell
    (b) C

    sugestão :use o ls para descobrir as propriedades do ficheiro e filtre o output do ls com o seu programa (que deve lançar o ls))

    do manual:

```
du - Display disk usage statistics
du [-a | -s] [-x] [file ...]
-s      Display only the grand total for the specified files.
```

12. Faça um shellscript que, dada a raiz de uma sub-árvore de directórios, se comporte de uma das seguintes maneiras, conforme a opção usada:

    -c : indique o número de ficheiros normais ("regular files", por oposição a sub-directórios) existentes em cada directório da sub-árvore.

    -e : forneça uma lista dos directórios vazios (que **não** contêem ficheiros, mas podendo ter sub-directórios).

    -a : oposto da opção anterior. Só mostra os directórios que contêem ficheiros regulares.

    Sugestões: pode utilizar mais de uma shellscript. Veja find(1), ls(1), wc(1)

13. Resolva o mesmo problema em C.

14. "The pipe notation using "⟨" and "⟩" survived only a couple of months; it was replaced by the present one that uses a unique operator to separate components of a pipeline. Altought the old notation had a certain charm and inner consistency, the new one is certainly superior."

    "Of course, it [the present pipeline notation] too has limitations. It is unabashedly linear, though there are situations in which multiple redirected inputs and outputs are called for. **For example, what is the best way to compare the outputs of two programs?** What is the appropriate notation for invoking a program with two parallel output streams?"

Dennis M. Ritchie, "The Evolution of the UNIX Time-sharing System", AT&T Bell Laboratories Technical Journal, Vol. 63, No. 8, October 1984

(a) i. Faça um shell script que dados:
   - um comando e opções (parâmetro 1)
   - 2 ou mais comandos (parâmetros 2 e seguintes)

   execute esses comandos com os seus outputs redirigidos para o comando dado como parâmetro 1.
   Este problema não tem uma solução geral. Portanto suponha que o comando que vai receber os outputs dos outros aceita como parâmetros uma série de nomes de ficheiros em que o nome "-" significa standard input. Suponha ainda que o programa funciona correctamente se os ficheiros forem "named pipes" (veja "mknod p" no manual). (Sugestão: veja "sh -c" no manual).

   ii. Suponha que tem dois programas (prog1 e prog2. O prog2 deve ser chamado com as opções "-v 2"). Qual é a resposta à 1ª pergunta do DMR usando o seu script ?

(b) Faça um programa em C que envie uma cópia do seu input a (2 ou mais) comandos dados como argumentos. Note que usar o "sh -c" trata automaticamente de expansões de nomes de ficheiros e de redireções.

   Nota: O código do programa deverá ser escrito em linguagem C.

(c) Faça um script, usando este programa em C, semelhante ao primeiro, mas que envie a cada um dos comandos executados uma cópia do input.

(d) use este script para, dado um ficheiro com 5 campos separados por "tabs", criar um ficheiro com os campos 3, 1 e 5 (por esta ordem) do ficheiro original.