isec Engenharia

Sistemas Operativos

2021/22

DEIS – Engenharia Informática

Prova escrita
Parte I

7 / Jan / 2122

Duração **60 minutos**

Sem consulta

Parte 1

1. (10%) Assuma que encontra um sistema Unix em que as passwords dos utilizadores **estão, encriptadas, no ficheiro** `/etc/passwd`. Descreva detalhadamente os méritos e os problemas desta situação. Se achar que há problemas, exemplifique uma ação que possa por em causa o sistema com base na situação descrita. Se não houver problema nenhum justifique também.

2. (12.5%) Num computador + sistema operativo moderno será ***desejável*** e ***possível*** impedir que um programa execute qualquer instrução arbitrária e efetue qualquer tipo de operação? Se achar que sim, explique porquê, como funciona e quais são os mecanismos (hardware/software) envolvidos nessa característica, e indique um exemplo de algo que não deva ser permitido e porquê.

3. (12.5%) O endereçamento virtual resolve alguns problemas que existem no endereçamento real. Identifique 2 (dois) desses problemas e explique como é que o endereçamento virtual os resolve.

4. (15%) O programa ***scicalc*** (**sci**entific **calc**ulator) é um programa que já existe para Linux (se o instalar) que efetu cálculos aritméticos de números de virgula flutuante. Usa-se em linha de comandos da forma que os exemplo abaixo ilustram.

1) Obter resultado de *3.5 + 4.2*

```
aluno@aulas:~ $ scicalc 3.5 + 4.2
7.7
aluno@aulas:~ $
```

2) Obter resultado de *2 elevado a 4*

```
aluno@aulas:~ $ scicalc 2 pow 4
16
aluno@aulas:~ $
```

3) Exemplo de uma operação desconhecida

```
aluno@aulas:~ $ scicalc 3 etc 4
unknown operation etc
aluno@aulas:~ $
```

4) Exemplo de um multiplicação com operando errado

```
aluno@aulas:~ $ scicalc 3 mult abc
multiplication needs numerical operands
aluno@aulas:~ $
```

O programa consegue trabalhar com diversas operações, indicadas sempre no formato ***operando1 operaç operando2***, usando-se tal como nos exemplos apresentados, e os quais, à luz da matéria dada, são suficien para perceber como se interage com o programa. Se a operação pedida for reconhecida o programa apresent valor resultante e termina com o código 0; se a operação for inválida, o programa apresenta uma mensagem erro e termina com o código 1.

Nesta questão: usando obrigatoriamente o programa já existente ***scicalc***, faça um programa em C para Unix/Li que peça ao utilizador os dois operandos e a operação a realizar e apresente o resultado no ecrã no segui formato: se não tiver havido erro, a mensagem é: "o resultado foi: *x*" em que *x* é o valor calculado; se tiver hav erro, aparece simplesmente "enganou-se em qualquer coisa". O seu programa não é responsável por analis validade dos dados de entrada, mas tem que descobrir se a operação correu bem ou não e apanhar o resulta O seu programa deve permitir ao utilizador, numa única execução, efetuar vários cálculos, terminando quand primeiro operando for "fim".
Explique a lógica e significado das ações do programa. Não se preocupe com *#includes*.
Uma vez que esta disciplina não é "Introdução à programação", torna-se evidente que o que está em aqui ca são os mecanismos estudados nas aulas e o seu programa, definitivamente, não vai ele próprio fazer os cálcu

isec Engenharia

Sistemas Operativos

2021/22

DEIS – Engenharia Informática

Prova escrita
Parte II
Época Normal
7 / Fev / 2021

Duração **1h30 minutos**

Com consulta

Parte 2

Respostas

- **Pergunta 1 e 2 numa folha**
- **Pergunta 3 noutra folha**

1. Utilizando apenas uma linha de comandos de Unix e sem recorrer a ficheiros temporários (exceto se explicitamente indicado):

   a) (5%) Apresente no ecrã o *username* e *user id* do utilizador que tem o segundo *user id* mais elevado no sistema (exemplo do conteúdo do ficheiro: ***maria 1006***).
   A correta realização desta tarefa não deve depender da pasta atual em que se encontra a linha de comando.

   b) (5%) Guarde na sua pasta pessoal um ficheiro com o nome "ficheiros.txt" cujo conteúdo será com o nome e o tamanho (apenas estes dois campos – uma linha por cada ficheiro), dos cinco maiores ficheiros que se encontram na diretoria /etc. Se o ficheiro já existir, a informação é acrescentada ao que já lá estava.
   A correta realização desta tarefa não deve depender da pasta atual em que se encontra a linha de comando.

2. (20%) Escreva um programa que receba o nome de **N** ficheiros através da linha de comandos e lance **N** *threads*, indicando a cada uma o nome de um desses N ficheiros. Cada *thread* irá contar o número total de linhas do ficheiro que lhe foi indicado e incrementar um contador geral, comum a todas elas. O incremento é feito a cada nova linha lida do ficheiro. No final, a *thread* inicial do processo deverá apresentar o valor do contador geral. Se em alguma das *threads*, ocorrer um erro na abertura do ficheiro, ela deve terminar cada uma das *threads* que foram criadas incluindo ela própria, de forma a que o processo possa encerrar também logo de seguida.

   Não é necessário implementar o algoritmo que conta o número de linhas. Deve considerar a função seguinte:

   ```
   int leProximaLinha (char ficheiro[]);   // Já está feita - é só usar
   ```

   A função devolve 1 se leu mais uma linha, 0 se chegou ao fim e não leu nada, -1 se não conseguiu aceder a ficheiro

NOTA: deve declarar todas as variáveis e estruturas de dados que utilizar. Não é necessário especificar #include

(pergunta 3 no verso da folha)

3. (20%) Pretende-se um sistema informatizado para gerir o acesso às caixas de fila única de um supermercado. Este sistema é constituído por um processo distribuidor (distribui clientes pelas caixas), N processos caixas e vários processos clientes. O mecanismo de comunicação entre os vários processos é *named pipes*. As aplicações cliente e caixa já estão implementadas, e o foco da questão é o distribuidor.

Cada novo cliente que surge irá contactar o distribuidor, fornecendo-lhe apenas uma *string* que corresponde ao seu nome. De seguida aguarda que caixa o contacte através do *named pipe* com o seu nome. Existem dois tipos de clientes, os normais e os prioritários, existindo um *named pipe* diferente para cada um destes dois tipos de cliente.

O distribuidor é responsável por lançar uma nova caixa para cada novo cliente. Cada caixa atende um cliente, após o que termina. Existe um número máximo de caixas em simultâneo, especificado através da variável de ambiente NCAIXAS.

O funcionamento do distribuidor é: se ainda não tiver atingido esse número, o distribuidor averigua se tem clientes para atender, e caso, tenha, lança uma caixa para ele, passando-lhe (através de argumentos de linha de comandos) toda a informação necessária que permita ao caixa interagir com o cliente. Se existir um cliente prioritário, será atendido antes dos clientes normais que ainda não tenham sido atribuídos a nenhum caixa.

Depreende-se que as caixas informam o distribuidor quando terminam o atendimento de um cliente. Essa notificação deve também ser feita por *named pipe*.

Todos os *named pipes* aqui referidos pertencem ao distribuidor.

Implemente o código do distribuidor.

NOTA: deve declarar todas as variáveis e estruturas de dados que utilizar. Não é necessário especificar #*includes*.