 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Época Normal	Ano letivo 2016/2017	Data 27-06-2017
	Curso Licenciatura em Segurança Informática em Redes de Computadores	Hora 10:00	
	Unidade Curricular Sistemas Operativos	Duração 02h15	

Observações

O tempo previsto para responder a cada questão é apresentado entre parêntesis retos.
A cotação atribuída a cada pergunta é apresentada entre parêntesis curvos.
Os alunos que fazem apenas uma parte da prova têm direito a metade do tempo de duração da prova.

Parte I (10 valores)

1) Para cada uma das seguintes afirmações deverá indicar se as considera verdadeiras ou falsas. Caso considere alguma afirmação como falsa deverá rescreve-la, transformando-a numa afirmação verdadeira. À simples negação não será atribuída nenhuma cotação: **(2,0 valores)**

a) [2,5 min]

Em escalonamento preemptivo, após a atribuição do CPU a um processo, este é executado até ao fim.

b) [2,5 min]

Uma situação de bloqueio acontece sempre que surge uma espera circular com resolução, por recursos com várias instâncias.

c) [2,5 min]

O princípio de localidade de referência espacial refere-se aos acessos a uma zona de memória temporalmente próximos entre si.

d) [2,5 min]

A técnica de evitar *deadlocks* necessita de suporte para o *rollback* sobre recursos.

2) [10 min]

(1,25 valores)

Considere um computador com **512KB** de memória que utiliza um sistema operativo que faz a gestão de memória pelo algoritmo **Buddy**. Apresente uma representação de como a memória ficaria dividida após a seguinte lista de acontecimentos:

- Chegada de um novo processo (P1) com 129KB tamanho;
- Chegada de um novo processo (P2) com 65KB tamanho;
- Saída do processo P1;
- Chegada de um novo processo (P3) com 120KB tamanho;
- Saída do processo P2;
- Chegada de um novo processo (P4) com 125KB tamanho.

3) [10 min]

(1,25 valores)

Considere o seguinte conjunto de processos, e as suas necessidades em termos de recursos alocados, máximos, e ainda necessários satisfazer. Considere ainda o número de recursos livres que o sistema dispõe:

[h|76K] - [p1|18K] - [p3|82K] - [h|128K] - [p2|51K] - [h|200K]


Apresente a lista resultante da aplicação do **worst-fit** para a seguinte lista de eventos:

- Chegada de P4 (82K);
- Chegada de P5 (52K);
- Saída de P3;
- Chegada de P6 (10K).

4) [15 min]

(2,0 valores)

Considere o seguinte conjunto de processos. Assuma que os processos chegam no instante de tempo indicado na tabela seguinte.

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Época Normal	Ano letivo 2016/2017	Data 27-06-2017
	Curso Licenciatura em Segurança Informática em Redes de Computadores	Hora 10:00	
	Unidade Curricular Sistemas Operativos	Duração 02h15	

Processo	Instante de chegada	Duração
P1	0.3	1.0
P2	0.5	0.8
P3	0.7	0.4
P4	1.1	0.5
P5	1.2	0.7

Desenhe o diagrama de Gantt da execução dos processos, considerando que o algoritmo de escalonamento é o **SRTF**. Calcule ainda o tempo médio de espera para os processos.

5) [10 min]

(2,0 valores)

Considerando um sistema com **8K de memória RAM** e o extrato da representação de **16K de memória virtual** (*paging*) representada na tabela. Assuma que **cada página tem 1024 bytes** de tamanho. Recorrendo à técnica utilizada pela *MMU*, indique a que endereço físico corresponde o endereço virtual **5290**, e a que endereço lógico corresponde o endereço físico **3131**.

110	1
000	0
010	1
100	0
000	0
101	1
011	1
000	0

6) [5 min]

(0,5 valores)

Apresente um possível resultado da execução do programa seguinte. Assuma que o **semáforo s1** foi inicializado com **6 recursos**, e que o PID do processo pai é **2016**:

```
...
while ( getpid() < 2019 ) {
    printf("I am %d\n", getpid());


    fork();
    ocupa_recursos( s1);
}

print(" I succeeded %d\n", getpid());
...
```

7) [5 min]

(1,0 valores)

Considere um disco com 150 cilindros (0-149) sendo que a cabeça de leitura/gravação está atualmente no cilindro 15 (tendo atendido anteriormente o 50). A fila de requisições é mantida em FIFO. Para a seguinte lista de requisições: **2, 5, 39, 13, 100, 148, 145, 144, 146**, apresente as listas de atendimento de pedidos ordenadas de acordo com o algoritmo **C-SCAN**.

 <small>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</small>	Tipo de Prova Época Normal	Ano letivo 2016/2017	Data 27-06-2017
	Curso Licenciatura em Segurança Informática em Redes de Computadores	Hora 10:00	
	Unidade Curricular Sistemas Operativos	Duração 02h15	

Parte II (10 valores)

```

...
// ap_mem é um apontador para uma zona de memória partilhada
// id_canal[2] contém referências para dois pipes – diálogo passageiro e balcão, um usado para escrita e outro para leitura
void atende_passageiro( void *ap_mem, int id_canal[2] )
{
    Passageiro passageiro;

    // bloqueante, até haver dados para leitura, provenientes do passageiro
    le_dados_passageiro( &passageiro, sizeof(Passageiro), id_canal[0] );

    (...) // processo balcão faz alterações/atualizações na variável "passageiro"
    // guarda dados do passageiro em memória
    guarda_dados_passageiro( &passageiro, sizeof(Passageiro), ap_mem );

    // envia dados para processo passageiro
    escreve_dados_passageiro( &passageiro, sizeof(Passageiro), id_canal[1] );
}
...

```

1) [15 min]

(2,5 valores)

Considerando o excerto de código apresentado no quadro anterior, implemente a função "le_dados_passageiro()", que permite a um processo balcão receber os dados enviados por um processo passageiro, através de um *named pipe* (acessível através de id_canal[0]). Os dados são guardados na variável "passageiro".

2) [15 min]

(2,5 valores)

Considerando o excerto de código apresentado no quadro anterior, implemente a função "escreve_dados_passageiro()", que permite a um processo balcão enviar os dados guardados em "passageiro" ao processo passageiro, usando para isso um *named pipe* (acessível através de id_canal[1]). Os dados estão guardados na variável "passageiro".

3) [15 min]

(2,5 valores)

Considerando o excerto de código apresentado no quadro anterior, implemente a função "guarda_dados_passageiro()", que permite a um processo balcão armazenar os dados de um passageiro, tendo em atenção que a zona de memória partilhada, acessível através de "a_mem", pode conter até 100 passageiros. Nota: admita que a zona de memória partilhada foi inicializada de tal forma que passageiro.id == 0 para um espaço ainda não ocupado.

4) [20 min]

(2,5 valores)

Implemente um programa que crie 3 processos balcão. Um dado processo balcão deve invocar a função "atende_passageiro()" sempre que existam passageiros para atender (ou seja, deve haver um mecanismo que permita aos passageiros "acordar" balcões sempre que haja passageiros para atender). Note ainda que:

- um balcão atende um e um só passageiro de cada vez;
- quando um balcão lê dados de um passageiro, o campo passageiro.porta_embarque é atualizado com um número aleatório;
- deve garantir que apenas um balcão de cada vez acede à zona de memória partilhada para aí guardar os dados do passageiro;
- quando o balcão atende todos os seus passageiros fica suspenso até haver mais passageiros para atender;