

## Q1a2

Q1.1 Comente a seguinte frase: "O sistema operativo é um computador virtual".

1,2 / 1,2

Esta afirmacao posso dizer que o sistema operativo e um computador virtual pois atua como uma camanda que abstrai e vai simplificar o hardware real do computador. Para mim da vista dum programador, isto significa que posso escrever codigo usando abstracoes simples sem me preocupar com os detalhes de baixo nivel do hardware subjacente, escondendo a complexidade, permitindo-me interacoes mais intuitivas.

Poderia mesmo dizer em minha opiniao, que o sistema operativo cria uma certa ilusao de um computador mais "friend" aonde ele transforma os recursos brutos do hardware em abstracoes de alto nivel, como ficheiros, processos e memoria virtual.

Em suma, com Sistema Operativo é construido um ambiente operacional que é muito mais facil de gerir e programar do que o hardware diretamente, sendo assim criado o tal "computador virtual" para o utilizador e as aplicacoes.

136 / 220 Word Limit

Q1.2 Um sistema operativo do tipo embutido encontra-se normalmente em que tipo de dispositivos?

0,5 / 1,2

Podemos encontrar um sistema operativo do tipo embutido em dispositivos desenhados para tarefas especificas que possuem recursos limitados ou requisitos de tempo real.

Podemos encontrar em varios dispositivos tais como:

- como telemoveis, consolas de jogo, nas nossas smart tvs
- na area em que exerco a minha profissao de enfermagem - tais como aparelhos para monitorisar a os doentes, ultrassom, equipamentos de ECG,,,
- nas coisas do dia a dia como por exemplo camaras de vigilancia, alarmes...nas caixas de multibanco (que gram tansacoes e interfaces com o utilizador)

Em suma, estes sistemas operativos embutidos encontram-se em praticamente todo o lugar, basta que estejemos atentos e podemos verificar que qualquer dispositivo eletrónico que não seja um computador de uso geral, mas que precise de software para funcionar de forma inteligente e autonoma, provavelmente usa este tipo de sistema operativo embutido.

## Q3a4

### Q1.3 Relacione e caracterize os conceitos de multiprogramação e de partilha no tempo (timesharing).

1,2 / 1,2

Do que pode ver, a Multiprogramacao é a capacidade de um sistema operativo manter multiplos processos na memoria principal - ram - ao mesmo tempo, de forma a poder maximizar a utilizacao da CPU.

No que diz respeito ao partilha do tempo (timesharing) podemos dizer que é uma extensao da multiprogramacao que permite que multiplos utilizadores interajam com o sistema em simultaneo recebendo uma pequena parte do tempo da CPU. Sendo o seu principal objectivo o de proporcionar uma interactividade e um tempo de resposta rapido para os varios utilizadores ou mesmo varias tarefas em vez de apenas maximizar somente a utilizacao da CPU.

A relacao que posso dizer presente neste dois conceitos é que a partilha de tempo é uma forma especifica e parece-me mais avancada de multiprogramacao, focada na interatividade dos varios utilizadores. Para mim a diferenca encontra-se em que a multiprogramacao procura a eficiencia e utilizacao de recursos, enquanto que a partilha de tempo tem o foco na capacidade de resposta e interatividade para varios utilizadores.

Posso dizer em sintese, que a multiprogramacao é como a base que permite ter multiplos programas em memoria ao passo que a partilha do tempo é a o mecanismos que usa essa base para fornecer uma experiencia para varios utilizadores ao mesmo tempo.

### Q1.4 Indique justificando as três razões principais da existência de tarefas.

0,8 / 1,2

As 3 principais razoes da existencia de tarefas quanto a mim sao:

1. Criacao e gestao de contexto de baixa sobrecarga - aqui criar, terminar e alternar entre threads é bastante mais leve e rapido do que fazer o mesmo com processos, pois os processos pedem a alocao de uma espaco de enderaco virtual completo, ao passo que a thread precisa apenas da pilha, registos e estado proprio. Um exemplo disto é um servidor web que precisa de lidar com muitos pedidos de clientes por segundo e neste caso a criacao de um novo processo para cada pedido nao seria sustentavel, por isso usar as threads neste caso permite gerir esta concorrência mais eficazmente.
2. Parelismo em Processadores Multicore - nas threads a unidade mais fina de execucao podem escalonar para executar em paralelo em diferentes nucleos (nao sei se consegui explicar correctamente). Um exemplo disto é quando uma aplicacao que processa imagens, aonde ela divide esta imagem em pequenos bocados e atribui o processamento de cada um destes bocados a uma thread que sera executada em paralelo e em nucleos diferentes. (analisar o conteudo duma imagem ).
3. Comunicacao e Partilha de Dados de forma eficiente - isto significa que a comunicacao e a troca de dados (aonde as threads sao utilizadas no mesmo processo de partilha automaticas).

## Q5a6

Q1.5 Explique em que consiste o método da barreira ( barrier ) de sincronização entre processos.

1/12

Pelo que compreendo este método barreira é como uma corrida de estafetas aonde todos os corredores tem de chegar a linha de passagem de testemunho para que a próxima fase de corrida comece, ou seja, o método barreira é um mecanismo de sincronização usado para garantir que todos os processos de um grupo possam atingir um determinado ponto comum no código antes que qualquer um deles possa continuar - poderia dizer um ponto comum (como o local de passagem de testemunho no exemplo que dei).

Posso dar um outro exemplo aonde é necessário efetuar um cálculo científico complexo e que vamos dividir em 3 fases:

1. fase 1 - cálculo inicial, aonde múltiplas threads calculam diferentes partes dos dados de forma independentes
2. fase 2 - barrier --> todas as threads para aqui, esperando que todas as outras estejam finalizadas
3. fase 3 - juntar os resultados --> só quando todos os cálculos da primeira fase estiverem finalizados é que uma nova thread poderia começar a juntar os resultados parciais para a próxima etapa.

Em suma, com este exemplo pretendia explicar que uma thread mais rápida poderia tentar usar dados incompletos da Fase 1, levando a resultados errados pois não esperou que todas tivessem terminado e esperado.

Q1.6 Observe o seguinte programa com dois processos:

```
/* pseudo-código */
semaforo recurso1, recurso2;

void processoA(void){
    down(&recurso1);
    down(&recurso2);
    utilizar_ambos_os_recurso1();
    up(&recurso2);
    up(&recurso1);
}

void processoB(void){
    down(&recurso2);
    down(&recurso1);
    utilizar_ambos_os_recurso1();
    up(&recurso1);
    up(&recurso2);
}
```

Este programa pode potencialmente levar a uma situação de impasse (deadlock)? Justifique.



Q1.7 No âmbito da memória virtual, a que corresponde o conceito de moldura de página (page frame) ou página física?

1/1,2

Penso que poderei dizer que uma moldura de página corresponde a um bloco de tamanho fixo de memória física - RAM. Ela é uma unidade de alocação de memória real, onde as páginas virtuais dos processos são carregadas quando precisam de ser consultados.

Eu tenho uma imagem de como explicar isto --> penso na RAM como um armário com várias gavetas aonde cada uma delas tem um tamanho fixo sendo cada uma delas uma moldura da página. Quando o sistema operativo precisa trazer uma página de um programa para a RAM, ele encontra a tal moldura de página e coloca-a lá dentro.

Para mim, do que me lembro, todas as molduras de página no sistema têm o mesmo tamanho, que são iguais ao tamanho de uma página virtual.

Q1.8 Considere um sistema de memória virtual com um esquema de Tabela de Páginas mononível. Supondo que não ocorre uma falha de página, descreva qualitativamente como é obtido o endereço físico a partir do endereço virtual. Considere páginas de igual dimensão.

0,9/1,2

Aqui está uma pergunta bastante difícil de explicar mas pelo que percebo num sistema de memória virtual com uma tabela de páginas mononível, a unidade de gestão de memória (MMU) converterá o endereço virtual em físico de seguinte forma (estando a página na RAM):

- A CPU começa por gerar um endereço virtual, depois a MMU divide esse endereço num número de página virtual;
- A MMU ao usar o número de página virtual como um índice para procurar na tabela de páginas do processo, vai levar a fornecer o número de quadro de página físico correspondente. (penso que é assim que se chama)
- Assim, com o número do quadro de página físico e com o deslocamento original (que indicava a posição dessa moldura) é criado o endereço físico.

Em suma, o que pretendo transmitir é que o número de página virtual diz - qual é a página virtual; e o offset diz - qual o byte dentro dessa página. A tabela de páginas vai traduzir este número de página virtual para - qual quadro físico de página na RAM; e o Offset permanecerá o mesmo para encontrar o byte dentro desse quadro físico.

Q1.9 Para um sistema de ficheiros baseado em i-nodes, explique no que consiste e como é implementado um "hard link".

1.2 / 1,2

Para um sistema de ficheiros baseado em i-nodes é como um nome alternativo para o mesmo ficheiro e sus dados - (atencao -nao é uma copia). Ele aponta diretamente para i-node, e o qual contem um contador de ligacoes que registra quantos hard links o referenciam.

Para mim, um sistema destes baseado em i-nodes - um hard link - é como uma entrada de directorio extra que aponta para o i-node de um ficheiro ja existente.

Passo agora a explicar o seu funcionamento: ao ser criado um hard link, uma nova entrada de directorio é feita para apontar para o i-node do ficheiro original, depois o contador de ligacoes no i-node do ficheiro vai ser incrementado; o ficheiro sera somente apagado quando o seu contador de ligacoes chegar a zero - ou seja todos os hard links forem todos removidos.

Em suma, ele permite qze o mesmo ficheiro seja acedido atraves de multiplos nomes de ficheiro qm diferentes localizacoes no sistema de ficheiros desde que seja num directorio diferente - sendo uma entrada de directorio adicional que aponta para um i-node existente.

Q1.10 Explique o conceito de I/O por mapeamento de memória.

1 / 1,2

O conceiot de I/O é uma tecnica onde os registos de controlo e buffers de dados dos dispositivos de entrada/saida sao tratados como se fossem locais de memoria RAM normais. Em vez de se usarm instrucoes de entrada/saida especiais, a CPU le ou escreve os enderecos de memorias especificos. Cabe ao hardware de direcionar estes acessos para o dispositvo entrada/saida correspondente.

E quais as vantagens deste conceito --> posso dizer que podemos ter uma programacao mais facil pois sao usadas instrucoes de memoria com load ou store; posso tambem acrescentar o facto de "unificacao de enderecos", nao havendo assim um espaco sepado para entradas/saidas. Tambem possum acrescentar a presenca duma maior eficiencia - pois em muitos sistemas, o acesso à memoria poderá ser mai rapido do que acessar a portas de entrada/saida separadas.

Esta conceito é utilizado em deiversas arquiteturas modernas que implementam uma abordagem hibrida combinada com buffers de dados de entrada/saida mapeados na memoria com portas de entradas/saidas separadas para registradores de controle.