# Conjunto De Perguntas De Exames

1. **Diga o que entende por Sistema Operativo, quais os seus objectivos, como pode ser classificado, e em que partes pode ser decomposto.**

É o software que gere os recursos do computador e serve de base para as restantes aplicações. Apresenta a complexidade do Hardware com uma interface simples de entender e de programar. É conhecida como máquina virtual. O sistema operativo é aquela porção de software que corre em modo Kernel.

Uma possível resposta é o facto de o SO gerir recursos, isto é, ele é responsável por gerir a memória virtual, CPU, também processos (acho) e qualquer tipo de dispositivos. Outra das vertentes é estender a máquina, isto é, fazer com que se possa comunicar com a máquina com mais facilidade, isto em termos de programação, pois existem inúmeras coisas que teríamos que saber e controlar, por exemplo para simplesmente ligar o cabo da impressora ao PC.

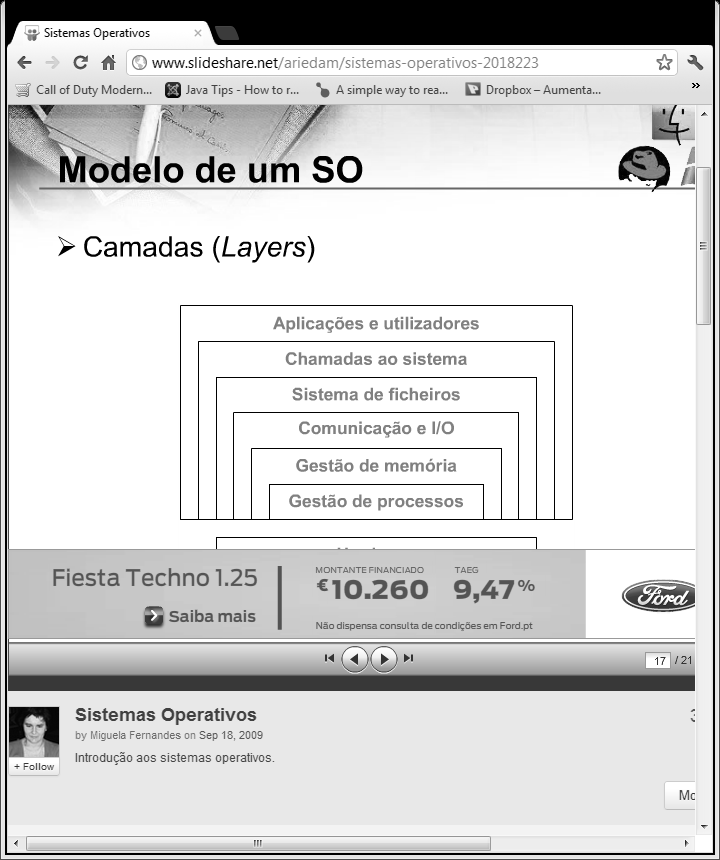
Os seus objectivos são:

* Executar programas do utilizador e tornar mais fácil a resolução de problemas.
* Tornar fácil o uso da máquina.
* Utilizar o hardware do computador duma forma eficiente

Podem ser classificados em:

* Multi-utilizador: o tempo de processamento do cpu de um computador pode ser partilhado por mais do que um utilizador de forma interactiva.
* Mono-utilizador: O CPU só pode estar dedicado de forma interactiva a um conjunto de processos do mesmo utilizador.
* Multi-programação: Capacidade de correr vários programas em simultâneo.
* Mono-programação: um programa a correr de cada vez.
* Dedicado: sistema operativo projectado para aplicações específicas.
* Uso geral: projectados para uma fácil utilização, permitem a execução de uma grande variedade de programas e reconhecem uma grande diversidade de periféricos.
* Centralizado: O SO cria uma máquina virtual sobre o único computador;
* Distribuído: o SO corre sobre um conjunto de computadores, dado a ilusão de que este conjunto é uma entidade única.

Pode ser decomposto em: gestão de processos, gestão de memória, comunicação e I/O, sistema de ficheiros, chamadas ao sistema, aplicações e utilizadores.



1. **Diga o que entende por:**
   1. **Programa;**
   2. **Linguagem de Programação;**
   3. **Processos;**
   4. **Multi-programação;**
   5. **Multi-Processamento;**
   6. **Tabela de Processos;**
   7. **Escalonador de SO;**
2. É uma sequência de instruções a serem seguidas e/ou executadas, na manipulação, redireccionamento ou modificação de um dado/informação ou acontecimento.
3. Uma linguagem de programação consiste em uma série de instruções para que o processador execute denominadas tarefas.
4. Um processo é basicamente um programa em execução, sendo este constituído pelo código executável e respectivos dados, pilha de execução, contador de programa, valor do apontador da pilha, valores dos registadores de hardware, alem do conjunto de outras informações necessárias à execução do programa.
5. Divide-se a memória em diversas partes com um Job alocado em cada uma delas, enquanto um Job espera a conclusão da sua operação I/O, um outro Job pode estar a utilizar o processador, a ideia é manter na memória simultaneamente uma quantidade de jobs suficiente para ocupar o processador a 100% e aproveitar ao máximo.
6. Capacidade de um sistema operacional executar simultaneamente dois ou mais processos. Pressupõe a existência de dois ou mais processadores. Difere da multitarefa, pois esta simula a simultaneidade, utilizando-se de vários recursos, sendo o principal o compartilhamento de tempo de uso do processador entre vários processos.
7. Tabela de processos contém informação sobre o estado dos processos. Cada processo tem um identificador PID, um *proprietário*, uma prioridade, memória atribuída, ficheiros abertos, o estado, etc... PS comando do UNIX que fornece informação sobre os processos em curso.
8. Parte do SO que faz a decisão de que processo irá utilizar os recursos do computador e durante quanto tempo.
9. **Enuncie as duas principais diferenças entre “Processo” e “Thread” no que diz respeito à gestão de processos e threads dum Sistema Operativo e no que diz respeito à gestão de memória.**

Cada processo tem o seu próprio program counter, stack, register set e espaço de endereçamento os processos não têm nada a haver uns com os outros.

Em muitos aspectos as threads são como mini processos, cada thread tem o seu próprio programa counter e stack para saber o que fazer. Threads partilham o CPU exactamente como os processos o fazem, só num multiprocessador é que eles correm em paralelo. Threads podem criar threads filho e bloquear à espera de um system call. Mas as threads não são tão independentes umas das outras como os processos são, todas as threads têm o mesmo espaço de endereçamento, o que significa que partilham as mesmas variáveis globais e podem limpar a stack de uma outra thread. Não existe protecção entre threads porque é impossível e não é necessária. Mas basicamente uma thread funciona exactamente como um processo.

A cada novo processo é necessário alocar um espaço de endereçamento, um contador do programa, variáveis. No caso de uma nova thread, não é preciso alocar espaço de endereçamento, sendo apenas necessário um contador e uma linha de controlo já que o escalonamento das threads é feito pelo processo que a cria.

1. **Quais os objectivos da comunicação entre Processos.**

O objectivo principal da comunicação entre processos é permitirem a transferência de dados entre si.

A comunicação entre processos tem várias fases: Condições de Concorrência, Secções Criticas, Troca de Mensagens, Escalonamento de processos.

1. **Explique para que diferentes fins podem servir as várias técnicas de “Multi-Programação” que estudou (semáforos,mutexes,monitores,etc..).**

**Semáforos:** E uma variável inteira, pode ter valor 0, indicando que não há nenhum sinal armazenado, ou pode ter valor positivo, indicando o número de sinais armazenados.

**Barreira:** Mecanismo de sincronização que é usado por grupos de processos. Algumas aplicações são divididas em fases e têm por regra que nenhum processo poderá prosseguir para a próxima fase ate que todos os processos estejam prontos para a próxima fase. Este procedimento e alcançado colocando uma barreira no final de cada fase.

**Pipes:** Um pipe permite a comunicação num só sentido entre dois processos, os dois processos e a pipe devem ter um ancestral em comum. Ambos os processos podem ler ou escrever do pipe. Cabe ao programador definir o sentido da comunicação. Comunicação bidireccional precisa de dois pipes. Leitura de uma mensagem de um pipe vazio bloqueia o processo até que lá seja colocada uma mensagem.

**Mutex:** Assegura que o consumidor e o produtor não acedem ao buffer em simultâneo. É incializado a 1.

**FIFOS:** Permite a comunicação entre processos sem qualquer relação de parentesco. Comunicação realizada por um canal de comunicação permanente e acessível a qualquer processo, suportado por um ficheiro especial.

1. **Exemplifique com o auxílio de pseudo-código, a utilização de semáforos para solucionar os problemas conhecidos como (Leitor-Escritor, Barbeiro Adormecido, Jantar De Filósofos), explicando o seu funcionamento.**

**Leitores e Escritores (Resumo):** Pode haver mais que um processo a ler da base de dados, mas se um processo estiver a escrever, mais nenhum pode estar a ler e/ou a escrever.

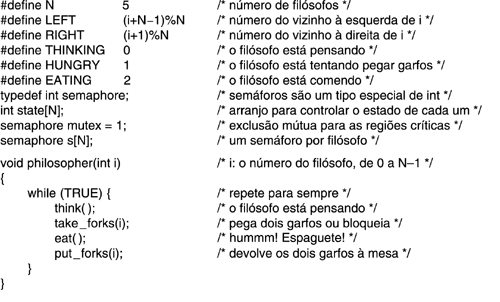
**Solução:** O primeiro leitor a ter acesso À BD executa um Down no semáforo db. Os leitores seguintes só incrementam um contador RC, antes de aceder À DB. Cada leitor que deixar a bd decrementa o RC, e o último leitor a sair executa um UP na BD (vendo que a variável db é igual a 0), permitindo que um escritor bloqueado (se houver algum), tenha acesso à BD.Os leitores têm prioridade sobre os escritores. Se um escritor aparecer quando houver leitores, o escritor espera.

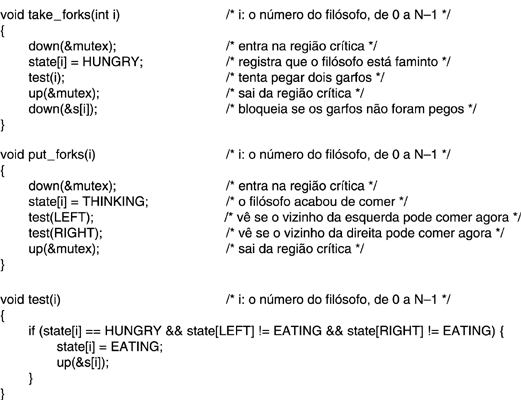


**Barbeiro Adormecido:** Usa-se três semáforos, costumers, que conta o número de clientes à espera, barbers, o número de barbeiro livres à espera de clientes e o mutex usado para exclusão mutua, e uma variável wating que também conta os clientes que estão à espera. Não há loop para o processo cliente, mas o barbeiro deve estar em loop tentando apanhar o próximo cliente.

****

**Jantar De Filósofos:** Apresenta cinco (5) filósofos e cinco (5) garfos. a ideia central deste problema é que dado os filósofos e os garfos em um jantar, cada filósofo necessita de dois (2) garfos para comer. Os filósofos podem estar em um destes três (3) estados: THINKING, HUNGRY ou EATING. se um filósofo está no estado THINKING e quer passar para o estado EATING, ele tenta pegar dois (2) garfos. Se ele não consegue pegar os dois (2) garfos, passa o estado HUNGRY. Se consegue pegar os dois garfos ele passa para o estado EATING. Enquanto no estado HUNGRY, o filósofo permanece tentando pegar os garfos, ou seja, fica esperando a libertação dos garfos.





1. **Defina deadlock ou impasse (interblocagem):**

Um deadlock é quando os processos bloqueiam e não conseguem desbloquear. Ocorre tipicamente quando dois ou mais processos que precisam de informação um do outro, e tal informação ainda não está disponível. Logo é criado um ciclo infinito, que faz com que os recursos em uso pelos processos em questão não sejam libertos.

1. **Considerando as várias estratégias possíveis para solucionar um impasse num SO, qual seria a estratégia a adoptar se tivesse o objectivo de reduzir o custo do dano causado pelo impasse. Exemplifique, explicando uma solução possível de acordo com a estratégia anteriormente adoptada.**

Considerando as várias estratégias possíveis para solucionar um impasse num SO, qual seria a estratégia a adoptar se tivesse o objectivo de reduzir o custo do dano causado pelo impasse. Exemplifique, explicando uma solução possível de acordo com a estratégia anteriormente adoptada. (Através de Rollback), para isto é necessário gravar periodicamente os estados dos processos. Assim quando se chega a um deadlock, é só voltar atrás usando a informação guardada, até que seja possível arranjar recursos para esse processo.

1. **Considere a seguinte lista de problemas relacionados com a utilização do computador:**
   1. **Perda de dados do disco;**
   2. **Privacidade dos dados (segurança);**
   3. **Robustez do SO;**
   4. **Rede de dados lenta;**
   5. **Tempo gasto com instalações e configurações de Software;**

**Solução para as seguintes estratégias:**

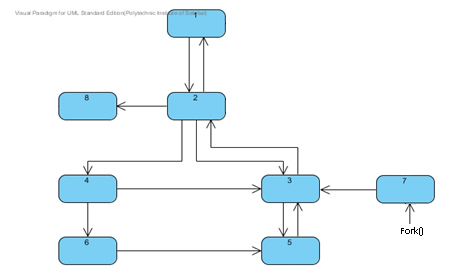
1. **Detectar e recuperar o problema;**

**ii. Evitar o problema;**

**iii. Prevenir o problema;**

**Robustez do SO:** Detecção e recuperação (Deixar deadlocks ocorrer, detectá-los e actuar), Evitar o problema (Desviar dinamicamente, através da cuidadosa alocação de recursos), Prevenir o problema (Negar 1 das 4 condições necessárias para criar 1 deadlock).

1. **Analisando o diagrama de estados dum processo, conforme apresentado, explique em que consiste cada um dos estados apresentados (8 no total):**



* 1. Execução em modo utilizador: Este tipo de execução é onde é utilizado o espaço de memória do utilizador, ou seja as threads aqui criadas, o kernel não sabe nada sobre elas.
  2. Execução em modo nuclear (Kernel): Este tipo de execução é feito no espaço de memória do núcleo.
  3. Pronto para executar: É um processo que está pronto a ser executado, esperando que o escalonador lhe dê tempo de CPU.
  4. Bloqueado: Processo que está à espera que o input esteja disponível.
  5. Pronto, em memória secundária: Se por memória secundária é a memória do disco, então é um processo que foi Swapped Out e está à espera de entrar para a memória principal para ser executado.
  6. Bloqueado, em memória secundária: Se por memória secundária é a memória do disco, então é um processo que está à espera que o input esteja disponível, para ser swapped in.
  7. Criado: Se for em UNIX é usado o comando fork() caso contrário, em Windows, é usado o comando CreateProcess.
  8. Extinção: Processo que tem estado como exit e está à espera de ser terminado pelo processo pai.

1. **Justifica as seguintes afirmações:**
   1. **Os segmentos têm de ser blocos de memória de tamanho variável;**
   2. **Através de paginação é possível obter um espaço de endereçamento (virtual) maior do que o espaço de endereçamento que a memória RAM instalada no computador permite.**
2. Segmentos são no fundo processos ou “buracos” entre processos, logo o tamanho deles supostamente deverá variar, visto que há processos maiores e menores.
3. A paginação de memória é o processo que consiste na subdivisão da memória física em pequenas partições, adicionando novos espaços de endereçamento à memória virtual. Sendo por isso possível ter mais memória virtual que física, embora com uma perda de performance. A paginação é implementada normalmente por unidades dedicadas de hardware integradas nos processadores.
4. **Um sistema operativo pode ser visto como um gestor de recursos num ambiente de concorrência, que recursos são esses e quais as entidades que se encontram em concorrência? Quais os objectivos principais do sistema Operativo nesta perspectiva?**

Como gestor de recursos creio que o que o SO tenta gerir são o CPU, espaço de memória e threads. As entidades em concorrência pensam que sejam os processos. Os objectivos do SO como gestor de recursos são fornecer de forma adequada e controlada a alocação do processador, das memórias e de dispositivos por todos os programas que competem por recursos.

1. **Explique qual o propósito de guardar a informação sobre os vários ponteiros para os segmentos de dados, texto e stack na tabela de processos.**

Guardar a informação é necessário pois, no caso do texto, se um texto está a ser escrito e perde a vez para um outro processo, é necessário guardar a informação de onde parou, para continuar a escrita no ponto correcto. Quando o processo é trocado do estado “Em execução” para o estado “Pronto a executar”, para que, quando retoma a execução, o processo possa continuar como se nada tivesse acontecido.

1. **Explique o funcionamento interno do SO quando ocorrem a sequência dos seguintes eventos:**
   1. **É criado o processo PID#52;**
   2. **O processo PID#31 fica bloqueado à espera do Input;**
   3. **O processo PID#43 termina normalmente a sua execução;**

O processo 52 é criado e fica em execução em modo utilizador e passa para modo nuclear, de seguida fica pronto a executar e de seguida fica bloqueado.

O processo 31 verifica se existe input, caso exista input passa de bloqueado para execução, caso não exista input continua bloqueado.

O processo 43 passa de bloqueado para pronto a executar, de seguida para execução nuclear e passa para extinção (zombie), que é o estado intermédio entre a execução e a destruição do processo.

1. **Ilustre, com o auxilio dum diagrama desenhado, quais os estados básicos possíveis para um processo e explica como é feita a transição entre esses estados.**



Ready: Processo disponível para execução, aguardando sua vez. (Ao ser selecionado, sofre transição para o estado Running, sendo que o sistema passa a interpretar o código).

Running: Processo actualmente em execução:

* Caso a fatia de tempo máximo de ocupação do processador seja atingida, o processo é interrompido, seus códigos/dados são mantidos na memória ou deslocados para a área de Swap, o processo retorna para a fila de Ready’s de acordo com as regras de escalonamento. A transição é feita para o estado Ready.
* Caso o processo realize uma chamada de sistema em que o tempo necessário para cumpri-la seja longo, o processo é interrompido, seus códigos/dados são mantidos na memória ou deslocados para a área de Swap e sofre transição para o estado Blocked. Caso a chamada de sistema seja imediatamente executada, o processo não sofre transição de estado.

Blocked: O processo está aguardando que uma chamada de sistema seja concluída (Uma chamada de sistema, ao ser concluída por um processo do sistema, gera uma sinalização (Evento) para o processo ser transferido para o estado de Ready)