1 - Conceitos Introdutórios

1. Descreva as 2 perspectivas de definição de um sistema de operação. Mostre claramente em que circunstâncias cada uma delas é relevante.

Top-Down (programador) - Não é necessário conhecer o hardware, pois o sistema de operação fornece um sistema funcional do sistema computacional (máquina virtual). Interface + hardware originam um ambiente uniforme de programação, que é operacionalizado através de chamadas ao sistema (intermediário entre o software e os componentes físicos do computador).
Bottom-Up (construtor) - Sistema de operação é visto como o que gere o sistema computacional atribuindo controlada e ordeiramente os seus recursos aos programas. Tem a função de gerenciar o sistema e seus recursos garantido a utilização eficiente de cada.

2. O que são chamadas ao sistema? Dê exemplos validos para o Unix (recorde que o Linux não é mais do que uma implementação específica do Unix). Explique qual é a sua importância no estabelecimento de um interface de programação de aplicações (API).

Chamadas ao sistema é o mecanismo usado pelo programa para requisitar um serviço do sistema operacional, quando o utilizador pretendo usar um recurso I/O (on/out). Quando acontece, o sistema operativo vai verificar quem gerou a interrupção, verificar se é válida e se tem autorização para efectuar a mesma, devolvendo no final o controlo para o utilizador.
Exemplos: - Gestão de ficheiros – open(), create(), close(), write(), read(), stat(), link(), …
- Gestão do processador – exec(), fork(), getpid(), gtuid(), kill(), …
- Gestão de comunicação – pipe(), msgget(), shmget(), msgcll(), …
Com um API, englobasse todas essas chamadas ao sistema numa biblioteca de acesso mais simples, permitindo ao utilizador uma abstracção dessas chamadas, assim como de outras funções resultantes da combinação entre elas

3. Os sistemas de operação actuais apresentam um ambiente de interacção com o utilizador de características eminentemente gráficas. Contudo, quase todos eles fornecem em alternativa um ambiente de interacção baseado em linhas de comandos. Qual será a razão principal deste facto?

A principal razão é a existência de uma metalinguagem de programação que possibilita uma abordagem mais estruturada à construção de comandos complexos. Num ambiente de texto podemos correr instruções mais complexas de forma mais simples que o ambiente gráfico. Temos também a criação de BATCHS que podem executar essas instruções complexas e tratar dos dados de maneira mais simples e até serem agendadas.
É extremamente útil para o acesso a máquinas remotas como servidores.

4. Distinga multiprocessamento de multiprogramação. Será possível conceber-se multiprocessamento sem multiprogramação? Em que circunstâncias?

Multiprocessamento é a capacidade de um sistema operacional executar simultaneamente dois ou mais processos, pressupondo-se a existência de dois ou mais processos.
Multiprogramação é a ilusão criada por um sistema de aparentemente poder executar em simultâneo mais programas do que o número de processadores existentes, o que exige que a atribuição do(s) processador(es) seja multiplexada no tempo entre os diferentes programas presentes.
É possível conceber-se multiprocessamento sem multiprogramação em sistemas computacionais onde o número de programas a ser executado em simultâneo seja sempre igual ou inferior ao número de processadores disponíveis.

5. Um tipo de multiprocessamento particular, o chamado processamento simétrico tornou-se muito popular com o surgimento no mercado dos processadores dualcore. Explique em que consiste e qual a razão desta popularidade.

Processamento simétrico consiste na distribuição de tarefas pelos dois processadores, o que vai permitir que diferentes processadores partilhem o processamento de instruções requisitadas pelo sistema interno. Também faz com que todos os processadores partilhem a memória principal de modo a que o acesso à memoria seja nativo.
Este processo é muito popular pois oferece um aumento linear na capacidade de processamento a cada processador adicionado.

6. Considere um sistema de operação multiutilizador de uso geral. A que níveis é que nele se pode falar de multiprogramação?

Para vários utilizadores do sistema de operação vai ser invisível o processamento dos programas dos outros utilizadores, como se os recursos do computador fossem exclusivos para si próprio, mas na realidade, durante o tempo em que os programas de um utilizador não estiverem a ser processados, os de outro estarão obrigatoriamente a ser.

7. Embora sendo um sistema interactivo, um sistema de tempo real tem características próprias muito bem definidas. Descreva duas delas, justificando convenientemente a sua resposta.

Tem de reagir dentro de um prazo pré-definido, a um estímulo do meio. As tarefas têm de ser executadas no prazo máximo atribuído, se essa tarefa não for completa ocorre uma falha no sistema. (exemplo da máquina que monitoriza os batimentos cardíacos de um paciente no hospital)
Previsibilidade do sistema, considerado previsível quando podemos antecipar o seu comportamento independentemente de falhas, sobrecargas e variações de hardware.

8. Os sistemas de operação do tipo *batch* são característicos dos anos 50 e 60, quando o custo dos sistemas computacionais era muito elevado e era necessário rentabilizar a todo o custo o hardware. A partir daí, com a redução progressiva dos custos, os sistemas tornaram-se interactivos, visando criar um ambiente de interacção com o utilizador o mais confortável e eficiente possível. Será que hoje em dia ainda se justifica sistemas deste tipo? Em que circunstâncias?

Sim, quando é necessário executar vários processos sem ser necessário a constante intervenção do utilizador e obter resultados simples e precisos

9. Quais são as semelhanças e as diferenças principais entre um sistema de operação de rede e um sistema distribuído?

Em ambos é possível criar uma transferência face ao utilizador de todos os recursos possíveis, isto é, ambos permitem partilha de informação estre sistemas computacionais diferentes.
Num sistema computacional de rede é possível partilhar impressoras, ficheiros e internet dos vários computadores de modo a dar a percepção de um só sistema computacional.

10. Os sistemas de operação de uso geral actuais são tipicamente *sistemas de operação de rede*. Faça a sua caracterização.

Os sistemas de operação de rede são sistemas que tiram partido das facilidades atuais de ligação entre sistemas computacionais, estando ligados a um canal de comunicação comum, com o fim de estabelecer um conjunto de serviços comuns a toda a comunidade. Estes sistemas vão se encontrar numa rede mundial, estando portanto ligados a todos os outros sistemas computacionais ligados a mesma rede, podendo assim aceder a vários serviços como partilha de ficheiros, correio electrónico, acesso à internet, acesso a sistemas computacionais remotos, etc…

11. A partilha de ficheiros é uma característica marcante dos sistemas de operação de rede. Procure explicar como esta facilidade pode ser usada para permitir a um qualquer utilizador o acesso a sua área de trabalho a partir de um computador genérico de uma rede local, tal como se passa nos laboratórios da universidade.

A informação do computador encontra-se disponível não só no computador como também na rede. Assim, ao estarmos ligados ao canal de comunicação comum entre o nosso computador genérico e a nossa área de trabalho, vai ser possível aceder aos ficheiros armazenados na dita área de trabalho.

12. Os sistemas de operação dos palmtops ou personal digital assistants (PDA) têm características particulares face ao tipo de situações em que são usados. Descreva-as.

São sistemas com recursos inferiores faces aos computadores. Possuem menos capacidade de processamento, menos capacidade de armazenamento, mas são de fácil transporte, de porte reduzido e maior autonomia. Podem ser usados como simples agendas pessoais, simples agregadores de contactos.

13. O sistema de operação Linux resulta do trabalho cooperativo de muita gente localizada em muitas partes do mundo, que comunica entre si usando a internet. Mostre porque é que este facto é relevante para a arquitectura interna do sistema.

Como o Linux é constituído em comunidade, o kernel é desenvolvido sobre a arquitectura top-down, isto é, a comunidade de programador pode desenvolver funções sem ter de saber a programação de kernel.

14. Numa arquitectura do tipo microkernel, diferentes funcionalidades do sistema de operação são executadas em modo utilizador como processos de sistema, originando aplicação a um só sistema computacional de um modelo de processamento muito comum em sistemas distribuídos – o chamado modelo cliente-servidor. Quem são neste contexto os clientes e os servidores? Que vantagens é que esta concepção apresenta na construção do núcleo do sistema de operação?

Neste caso o servidor vai ser o microkernel e os clientes vão ser os processos do utilizador. Isto permite-nos ter 1 kernel minimalista simples que pode ser usado em qualquer computador.

2- Gestão do Processador

1. A modelação do ambiente de multiprogramação através da activação e desactivação de um conjunto de processadores virtuais, cada um deles associado a um processo particular, supõe que dois factos essenciais relativos ao comportamento dos processos envolvidos sejam garantidos. Quais são eles?

1 - Garantir que a execução dos processos não é afectada pelo instante, ou local no código onde ocorre a comutação.
2 - Garantir que não são impostas quaisquer restrições relativamente aos tempos de execução, totais ou parciais, dos processos.

2. Qual é a importância da tabela de controlo de processos (PCT) na operacionalização de um ambiente de multiprogramação? Que tipo de campos devem existir em cada entrada da tabela?

PCT é responsável por manter a ordem de processos e por guardar toda a informação relativa a cada processo. É usada intensivamente pelo scheduler para fazer gestão do processador e de outros recursos.
Entradas da tabela:
- Identificadores - do processo, do pai do processo e do utilizador a que o mesmo pertence;
- Caracterização do espaço de endereçamento - sua localização (em memória principal ou na área de swapping, RAM/SWAP), de acordo com o tipo de organização de memória estabelecido;
- Contexto do processo - valores de todos os registos internos do processador no momento em que se deu a comutação do processo;
- Contexto de I/O - informação sobre os canais de comunicação e buffers associados;
- Informação de estado e de “scheduling” - estado de execução do processo (de acordo com o diagrama de estados) e outras informações associadas.

3. O que é o scheduling do processador? Que critérios devem ser satisfeitos pelos algoritmos que o poem em prática? Quais são os mais importantes num sistema multiutilizador de uso geral, num sistema de tipo batch e num sistema de tempo real?

Scheduling é a parte integrante do núcleo central e é responsável pelo tratamento de interrupções e por agendar a atribuição do processador e de muitos outros recursos do sistema computacional.
Os algoritmos que põem em prática este sistema tem que satisfazer os seguintes critérios:
-**Justiça** na fracção de tempo dado a cada processo;
-**Previsível** pois cada processo deve ser razoavelmente constante e independente da sobrecarga pontual a que o sistema computacional possa estar sujeito;
-Maximizar o **Throughput** que consiste em maximizar o número de processos terminados por unidade de tempo;
-Deve procurar minimizar o **Tempo de Resposta** às solicitações feitas pelos processos interactivos;
-Deve procurar minimizar o **Tempo de Turnaround** (tempo de espera) pelo completamente de um *job* no caso de utilizadores num sistema batch;
-Deve procurar garantir o máximo de cumprimento possível das **Deadlines** (metas temporais) impostas pelos processos em execução;
-Deve procurar manter o processador o mais possível ocupado com a execução dos processos dos utilizadores (**Eficiencia**).
Os mais importantes num sistema:
Multiutilizador de uso geral: Justiça para todos os utilizadores terem direito ao mesmo tempo de processamento.
Tipo batch: Reduzir os tempos de Turnaround dos jobs que contituem a fila de processamento.
Tempo real: Previsibilidade pois pode ser necessário tratamento de processos com condições de alarme ou de acções cuja execução efectiva tem que ser mantida dentro de intervalos temporais bem definidos.

4. Descreva o diagrama de estados do scheduling do processador em três níveis. Qual é o papel desempenhado por cada nível? Num sistema do tipo batch multiprogramador fará sentido a existência de três níveis de scheduling?

O nível do processador (nível baixo) permite fazer uma melhor gestão do processador de modo a que todos os processos sejam atendidos ao mesmo tempo. O nível de memória principal (nível médio) permite fazer a gestão de memória para cada processo de modo a permitir mais processos em memória. O nível de ambiente de multiprogramação (nível alto) é responsável por criar os processos e terminá-los caso o batch seja muito complexo, isto é, se invocar muitos processos.
Num sistema do tipo batch multiprogramador faz de facto sentido a existência de três níveis de scheduling para garantir um equilíbrio adequado entre o serviço aos processos que coexistem e a ocupação do processador.

5. Os estados *READY-TO-RUN* e *BLOCKED*, entre outros, têm associadas filas de espera de processos que se encontram nesses estados. Conceptualmente, porém, existe apenas uma fila de espera associada ao estado *READY-TO-RUN*, mas filas de espera múltiplas associadas ao estado *BLOCKED.* Em princípio, uma por cada dispositivo ou recurso. Porque é que é assim?

Isto acontece porque um processo só abandona o estado BLOCKED quando ocorre um acontecimento externo (que pode ser o acesso a um recurso, complemento de uma operação de entrada/saída, etc). Assim, são necessárias varias filas de espera para se

saber qual foi o acontecimento externo que causou a transição do estado BLOCKED para o READY-TO-RUN.

6. Indique quais são as funções principais desempenhadas pelo *kernel* de um sistema de operação. Neste sentido, explique porque é que a sua operação pode ser considerada como um *serviço de excepções*.

As funções principais do kernel são: gestão do processador, gestão de memória e gestão do ambiente de multiprogramação. Podemos considerar a sua operação como um serviço a excepção pois toda a comutação de processos pode ser vista como uma rotina de serviço à excepção, com uma pequena alteração: normalmente a instrução que vai ser executada após o serviço da excepção, é diferente daquela cujo endereço foi salvaguardado antes de se iniciar o processamento da excepção.

7. O que é uma *comutação de contexto*? Descreva detalhadamente as operações mais importantes que são realizadas quando há uma comutação de contexto.

Uma comutação de contexto é no fundo uma comutação de processos. As operações mais importantes quando é realizada uma comutação de contexto são: salvaguarda do PC e do PSW na stack do sistema; carga no PC do endereço de rotina de serviço à excepção; salvaguarda do conteúdo dos registos internos do processador que vão ser usados; processamento da excepção; restauro do conteúdo dos registos internos do processador que foram usados; restauro do PSW e do PC a partir da stack do sistema.

8. Classifique os critérios que devem ser satisfeitos pelos algoritmos de *scheduling* segundo as perspectivas sistémica e comportamental, e respectivas subclasses. Justifique devidamente as suas opções.

Scheduling sob a perspectiva sistémica deve satisfazer critérios orientados aos utilizadores, pois estão relacionados com o comportamento do sistema de operação na perspectiva dos processos e dos utilizadores; deve também satisfazer critérios orientados ao sistema, pois estes estão relacionados com o uso eficiente dos recursos do sistema de computação. Já o scheduling sob a perspectiva comportamental deve satisfazer critérios orientados ao desempenho e mais alguns tipos de critérios.

9. Distinga disciplinas de prioridade estática das de prioridade dinâmica. Dê exemplos de cada uma delas.

Disciplinas de prioridade estática são aquelas que o seu método de definição é determinístico (como o utilizado em sistema de tempo real). Prioridades dinâmicas são aquelas que o seu método de definição depende da historia passada de execução do processo (como o usado em sistemas operacionais interactivos).

10. Num sistema de operação multiutilizador de uso geral, há razões diversas que conduzem ao estabelecimento de diferentes classes de processos com direitos de acesso ao processador diferenciados. Explique porquê.

Para evitar que haja um adiamento indefinido na calendarização de execução de um processo.

11. Entre as políticas de *scheduling preemptive* e *non-preemptive*, ou uma combinação das duas, qual delas escolheria para um sistema de tempo real? Justifique claramente as razões da sua opção.

Para um sistema de tempo real, uma politica de scheduling preemptive é mais adequada, porque num sistema de tempo real existe a necessidade de fazer constantes comutações de processos sempre que se executa um processo mais prioritário.

12. Foi referido nas aulas que os sistemas de operação de tipo *batch* usam principalmente uma política de *scheduling non-preemptive*. Será, porém, uma política pura, ou admite excepções?

É uma política pura pois este ordena os jobs por tempo (sendo o que irá ocupar menos tempo o primeiro a ser executado), atribuindo-lhes o processador sem nunca o retirar do processo que no momento o detém, tendo que aguardar que o processador seja libertado para o processo seguinte quando o anterior for executado até ao fim.

13. Justifique claramente se a disciplina de *scheduling* usada em Linux para a classe *SCHED_OTHER* é uma política de prioridade estática ou dinâmica?

A classe SCHED_OTHER usa uma disciplina de scheduling dinâmica, pois o algoritmo de scheduling combina a historia passada de execução do processo e a sua prioridade, e maximiza o tempo de resposta dos processos I/O - intensivos sem produzir adiantamento indefinido para os processos CPU – intensivos. O algoritmo usado por esta classe baseia-se em créditos.

14. O que é o *aging* dos processos? Dê exemplos de duas disciplinas de *scheduling* com esta característica, mostrando como ela é implementada em cada caso.

Aging de processos é a alteração da prioridade dos processos tendo em conta o seu "tempo de vida". O algoritmo de total fairness vai utilizar o processo de aging pois à medida que um processo espera o seu tempo de espera vai sendo incrementado e sempre que ele é calendarizado para execução o seu tempo de espera é decrementado sendo depois executado o processo com a maior diferença entre tempo de espera e tempo de execução virtual. Outro algoritmo que utiliza o processo de aging é o algoritmo de calculo de prioridade dinâmica que procura estimar a fracção da ocupação da janela de execução seguinte em termos de ocupação de janelas passadas, atribuindo se o processador para o processo com menor estimativa.

15. Distinga *threads* de processos. Assumindo que pretende desenvolver uma aplicação concorrente usando um dos paradigmas, descreva o modo como cada um afecta o desenho da arquitectura dos programas associados.

Um processo dedica-se a agrupar um conjunto de recursos enquanto as threads constituem as entidades executáveis independentes mesmo dentro do contexto do processo. Ao escolher uma aplicação usando o paradigma de threads, ao envolver menos recursos por parte do sistema de operação, possibilita que operações como a sua criação e destruição e mudança de contexto, se tornem menos pesadas e portanto menos eficientes; alem disso em multiprocessadores simétricos torna-se possível calendarizar para execução em paralelo múltiplos threads da mesma aplicação, aumentando assim a sua velocidade de execução.

16. Indique, justificadamente, em que situações, um ambiente *multithreaded,* pode ser vantajoso.

O uso de um ambiente multithreaded é vantajoso quando vamos ter programas que envolvem múltiplas actividades e atendimentos a múltiplas solicitações; quando existe uma partilha do espaço de endereçamento e do contexto de I/O entre as threads em que uma aplicação é dividida, torna-se mais simples a gestão de memória principal e o acesso aos dispositivos de entrada e saída de uma maneira eficaz.

17. Que tipo de alternativas pode o sistema de operação fornecer à implementação de um ambiente *multi-threaded*? Em que condições é que num multiprocessador simétrico os diferentes *threads* de uma mesma aplicação podem ser executados em paralelo?

Em alternativa à implementação de um ambiente multithreaded podemos apenas ter no nosso sistema computacional threads kernel visto que estas podem ser executadas paralelamente num sistema multiprocessador. Um multiprocessador simétrico possibilita a calendarização de várias threads da mesma aplicação para execução paralela o que é possível se o sistema computacional permitir multithreading.

18. Explique como é que as *threads* são implementadas em Linux.

Em Linux existe o comando fork que cria um novo processo a partir de um já existente por copia integral do seu contexto alargado, existe também o clone, que cria um novo processo a partir de um já existente por copia apenas do seu contexto restrito, partilhando o espaço de endereçamento e o contexto de I/O e iniciando a sua execução pela invocação de uma função que é passada como parâmetro. Desta forma, não há distinção efectiva entre processos e threads, que o Linux designa por tasks, e elas são tratadas pelo kernel da mesma maneira.

19. O principal problema da implementação de *threads*, a partir de uma biblioteca que fornece primitivas para a sua criação, gestão e *scheduling* no nível utilizador, é que quando uma *thread* particular executa uma *chamada ao sistema* bloqueante, todo o processo é bloqueado, mesmo que existam *threads* que estão prontas a serem executados. Será que este problema não pode ser minimizado?

Este problema pode ser minimizado implementando threads de kernel. Estas threads são implementadas directamente ao nível kernel que providencia as operações de criação, gestão e scheduling de threads. Esta implementação é menos eficiente que no caso das threads de utilizador, mas o bloqueio de uma thread particular não afecta a calendarização para a execução das restantes threads.

20. Num ambiente *multithreaded* em que as *threads* são implementadas no nível utilizador, vai haver um par de *stacks* (sistema / utilizador) por *thread*, ou um único par comum a todo o processo? E se os *threads* forem implementados ao nível do *kernel*? Justifique.

Num ambiente em que as threads são implementadas no nível do utilizador vai existir um par de stacks comum a todo o processo visto que threads ao nível utilizador não suportam vistas pelo kernel como uma única thread, tendo assim um só par porque são

tratadas como uma única thread. Já nas threads implementadas ao nível do kernel vamos ter uma stack por thread, pois como as threads são lançadas pelo kernel cada uma destas threads vai ter um par de stacks.

3 – Comunicação Entre Processos

1. Como caracteriza os processos em termos de interacção? Mostre como em cada categoria se coloca o problema da exclusão mútua.

Dividem-se entre processos independentes e cooperantes.
• processos independentes – quando são criados, têm o seu 'tempo de vida' e terminam sem interactuarem de um modo explícito; a interacção que ocorre é implícita e tem origem na sua competição pelos recursos do sistema computacional; trata-se tipicamente dos processos lançados pelos diferentes utilizadores num ambiente interactivo e/ou dos processos que resultam do processamento de jobs num ambiente de tipo batch;
• processos cooperantes – quando partilham informação ou comunicam entre si de um modo explícito; a partilha exige um espaço de endereçamento comum, enquanto que a comunicação pode ser feita tanto através da partilha de um espaço de endereçamento, como da existência de um canal de comunicação que interliga os processos intervenientes.
No processo independente eles querem aceder ao mesmo recurso mas só um é que pode. Isto é da responsabilidade do SO gerir.
No processo cooperante existe uma memoria partilhada onde eles partilham a informação mas só um e que pode aceder a esse espaço. É da responsabilidade dos próprios processos garantir a exclusão mútua neste processo.

2. O que é a competição por um recurso? Dê exemplos concretos de competição em, pelos menos, duas situações distintas.

A competição por um recurso acontece quando dois processos distintos necessitam de aceder ao mesmo recurso. Acesso a discos rígidos, acesso ao mesmo periférico, acesso a memoria partilhada e aos diversos dispositivos de I/O.

3. Quando se fala em região crítica, há por vezes alguma confusão em estabelecer-se se se trata de uma região crítica de código, ou de dados. Esclareça o conceito. Que tipo de propriedades devem apresentar as primitivas de acesso com exclusão mútua a uma região crítica?

Quando falamos em acesso a um recurso, ou a uma região partilhada, estamos a referir-nos a execução de código por parte do processador, assim sendo estamos a falar de uma região de código. Uma região critica é um pedaço de código que usa recursos partilhados e que a execução desse código exige que o recurso partilhado não possa ser utilizado por outro processo com risco de alteração de informação. Logo este código não pode ser interrompido até terminar a região critica.
As propriedades são: garantir a imposição de exclusão mútua, um processo fora da região critica nao pode impedir outro de lá entrar, nao pode ser adiada indefinidamente a

possibilidade de acesso a região critica a qualquer processo que o requeira, o tempo de permanência de um processo na região critica e necessariamente finito.

4. Não havendo exclusão mútua no acesso a uma região crítica, corre-se o risco de inconsistência de informação devido à existência de condições de corrida na manipulação dos dados internos a um recurso, ou a uma região partilhada. O que são condições de corrida? Exemplifique a sua ocorrência numa situação simples em que coexistem dois processos que cooperam entre si.

Condições de corrida ocorrem quando vários processos tentam aceder e manipular dados partilhados concorrentemente em que o valor final desse dados partilhados dependem do ultimo processo a lá terminar. Se estes processos cooperam quer dizer que estão a usar o acesso a região de memoria de dados partilhada e não ao recurso, logo se por exemplo um processo carregar em memoria partilhada os dados de um ficheiro e os alterar, um processo que aceda posteriormente a esses dados e os alterar, vai desfazer o que o processo anterior fez.

5. Distinga deadlock de adiamento indefinido. Tomando como exemplo o problema dos produtores / consumidores, descreva uma situação de cada tipo.

Deadlock acontece quando dois/+ processos ficam a espera eternamente de acesso a regiões criticas, e de acontecimentos que irão resultar de outros processos em espera;
Adiamento indefinido acontece quando um/+ processos tentam aceder a regiões criticas respectivas mas são ultrapassados por processos com prioridade maior sendo assim o seu acesso sucessivamente adiado.
No caso do deadlock existe problema de produtor/consumidor quando um produtor quer adicionar novos produtos na fila e a fila está cheia, como os produtores tem prioridade sobre os consumidores, consumidores não vão conseguir tirar produtos da fila criando um ciclo em que os produtores estão sempre a tentar adicionar mas não conseguem.
No caso do adiamento indefinido poderá ser causado por um aparecimento constante de acesso a região critica por parte dos produtores. Uma vez que os produtores tem sempre prioridade sobre os consumidores, estes ficam em adiamento indefinido.

6. A solução do problema de acesso com exclusão mútua a uma região crítica pode ser enquadrada em duas categorias: soluções ditas software e soluções ditas hardware. Quais são os pressupostos em que cada uma se baseia?

Solução software: esta pode funcionar em qualquer processador[(mono/multi)processador], com memoria partilhada nem que em ultimo recurso se use o conjunto básico de instruções. Quando temos operações em multiprocessador torna-se necessário o uso de um arbitro, de modo a controlar a tentativa de acesso simultâneo a mesma posição de memoria.
Solução hardware: esta solução não pode funcionar em qualquer processador pois usa instruções especiais do processador, logo tem de ser implementada para o mesmo, garantindo atomicidade na leitura e subsequente escrita de uma posição de memoria. Estas são a maior parte das vezes suportadas pelo SO.

7. Dijkstra propôs um algoritmo para estender a solução de Dekker a N processos. Em que consiste este algoritmo? Será que ele cumpre todas as propriedades desejáveis? Porquê?

O algoritmo de Dijkstra cria um array com N posições, em que cada pode ter 3 estados diferentes(NÃO, INTERESSADO, DECIDIDO). O processo que quer entrar é marcado como INTERESSADO e é feita a verificação da prioridade sobre os outros. Caso a prioridade seja sua este é marcado como DECIDIDO, caso contrario verifica se o processo com prioridade quer/não entrar passando DECIDIDO caso o processo com prioridade não queira entrar. Existe a possibilidade de este não resolver o adiamento indefinido, pois um processo pode perder sempre a disputa pela prioridade se um processo que já usou o processador desejar voltar a entrar.

8. O que é que distingue a solução de Dekker da solução de Peterson no acesso de dois processos com exclusão mútua a uma região crítica? Porque é que uma é passível de extensão a N processos e a outra não (não é conhecido, pelo menos até hoje, qualquer algoritmo que o faça).

Na solução de Dekker é usado um mecanismo de alternância para garantir que os processos acedam a região critica um a seguir ao outro e na solução de Dekker foi obrigar todos os processos a escrever a sua identificação na mesma variável de modo a que seja feita uma comparação de quem foi o ultimo que usou a região critica, colocando todos os processos em fila de espera por ordem de chegada. Esta permite uma implementação mais directa para para N processos que o Dekker.
Ambos são passiveis a extensão a N processos, excepto que a tentativa de expansão do de Dekker para vários processos (Dijkstra), não resolve a possibilidade de haver adiamento indefinido, não cumprindo assim todas as propriedades. Enquanto que a expansão de Dekker para N processos (““Lamport””) não tem o problema do adiamento indefinido.

9. O tipo de fila de espera que resulta da extensão do algoritmo de Peterson a N processos, é semelhante àquela materializada por nós quando aguardamos o acesso a um balcão para pedido de uma informação, aquisição de um produto, ou obtenção de um serviço. Há, contudo, uma diferença subtil. Qual é ela?

A diferença é que o ultimo a chegar coloca-se na ultima posição da fila e vai verificando se a posição seguinte esta livre, avançando caso esteja.

10. O que é o busy waiting? Porque é que a sua ocorrência se torna tão indesejável? Haverá algum caso, porém, em que não é assim? Explique detalhadamente.

O busy waiting é o problema causado pelas soluções de software ao problema de acesso a uma região critica com exclusão mútua e pelo uso da flag de locking que indica que aquela posição de memoria esta bloqueada aquela processo podendo mesmo não estar a ser acedida. Isto implica que os processos aguardam a entrada da região critica em estado activo. Esta solução é mais indesejada nos sistemas monoprocessador, pois atribuição do processador a um processo que pretende acesso a uma região critica que esta associada a um recurso ou uma região partilhada que esta a ser acedida por outro processo, fazendo que o tempo de atribuição do processador se esgote. Esta solução e interessante e desejável em sistemas multiprocessador, em que os processadores são dedicados, tornando o busy waiting vantajoso numa situação de bloqueio.

11. O que são flags de locking? Em que é que elas se baseiam? Mostre como é que elas podem ser usadas para resolver o problema de acesso a uma região crítica com exclusão mútua.

Servem para garantir atomicidade (a sua execução não pode ser interrompida) na leitura e escrita sucessivas duma variável. Como foi dito baseiam-se no uso duma variável. Os processadores tem a instrução test-and-set (tas) que permite duma maneira atómica a leitura duma posição memória, este tipo de operação usam variáveis flags de locking que após serem testados com sucesso, mudam o seu estado, qualquer processo q a seguir tentar entrar nessa região entrará em busy waiting até q o processo que fez o lock faça unlock da flag de locking.

12. O que são semáforos? Mostre como é que eles podem ser usados para resolver o problema de acesso a uma região crítica com exclusão mútua e para sincronizar processos entre si.

Um semáforo é um dispositivo de sincronização inventado por Dijkstra. Este é uma estrutura de dados composta por um valor unsigned int que funciona como flag de acesso e de posição na fila e um NO para uma fila. O sistema de sincronização é feito por rotinas de sleep e de wakeup. Este utiliza estas rotinas para controlar o acesso a região critica colocando a dormir quando estes precisam de informação que outros processos ainda não disponibilizaram e acordando-os quando a informação já esta disponível.

13. O que são monitores? Mostre como é que eles podem ser usados para resolver o problema de acesso a uma região crítica com exclusão mútua e para sincronizar processos entre si.

Monitores são dispositivos de sincronização que pode ser construído como sendo um modulo especial e pela linguagem de programação concorrente. Este é constituído por estruturas de dados interna, código de inicialização e um conjunto de primitivas de acesso.

14. Que tipos de monitores existem? Distinga-os.

monitor de Hoare – o thread que invoca a operação de signal é colocado fora do monitor para que o thread acordado possa prosseguir; é muito geral mas a sua implementação exige a existência de um stack onde são colocados os threads postos fora do monitor por invocação de signal;
monitor de Brinch Hansen – o thread que invoca a operação de signal liberta imediatamente o monitor (signal a ultima instrução executada) simples de implementar mas pode tornar-se bastante restritivo porque só há possibilidade de execução de um signal em cada invocação de uma primitiva de acesso;
monitor de Lampson /Redell – o thread que invoca a operação de signal pros- segue a sua execução, o thread acordado mantém-se fora do monitor e compete pelo acesso a ele; é simples de implementar mas pode originar situações em que alguns threads são colocados em adiamento indefinido.

15. Que vantagens e inconvenientes as soluções baseadas em monitores trazem sobre soluções baseadas em semáforos?

• vantagens
– suporte ao nível do sistema de operação – porque a sua implementação é feita pelo kernel, as operações sobre semáforos estão directamente disponíveis ao programador de aplicações, constituindo-se como uma biblioteca de chamadas ao sistema que podem ser usadas em qualquer linguagem de programação;
– universalidade – são construções de muito baixo nível e podem portanto devido sua versatilidade ser usadas no desenho de qualquer tipo de soluções;
• desvantagens
– conhecimento especializado – a sua manipulação directa exige ao programador um domínio completo dos princípios da programação concorrente, pois é muito fácil cometer erros que originam condições de corrida e, mesmo, deadlock.

16. Em que é que se baseia o paradigma da passagem de mensagens? Mostre como se coloca aí o problema do acesso com exclusão mútua a uma região crítica e como ele está implicitamente resolvido?

O paradigma da troca de mensagens é feito pelo uso de um canal de comunicação. Quando um processo A pretende enviar uma mensagem ao processo B coloca-a no canal de comunicação. O processo B para receber a mensagem apenas tem de ler o canal de comunicação. Como o acesso ao canal de comunicação é feito através do processo este não sabe se lá existe mais informação de outro processo. Quando um processo envia informação pelo canal de comunicação tem de indicar o processo a quem quer enviar assim como o processo que escuta o canal de comunicação tem de indicar o seu id.

17. O paradigma da passagem de mensagens adequa-se de imediato à comunicação entre processos. Pode, no entanto, ser também usado no acesso a uma região em que se partilha informação. Conceba uma arquitectura de interacção que torne isto possível.

Para implementar uma arquitectura que use este tipo de paradigma pode ser feito de duas formas, uma estrutura central (processa) controla o acesso informando os súbitos de se podem ou não entrar na região através de após o pedido enviar uma mensagem (ficando o processo que pediu acesso em bloqueio na recepção da resposta). Outra forma seria um sistema perguntar a toda a gente se está lá alguém a entrar. Quando receber resposta, quando sair avisa toda gente que sai, perde-se rentabilização na medida que tudo fica à espera que saia em sincronização bloqueante.

18. Que tipo de mecanismos de sincronização podem ser introduzidos nas primitivas de comunicação por passagem de mensagens? Caracterize os protótipos das funções em cada caso (suponha que são escritas em Linguagem C).

Existem dois níveis:
Sincronização não bloqueante: quando a sincronização é da responsabilidade dos processos intervenientes, isto é, a operação de envio envia a mensagem e regressa sem qualquer informação sobre se a mensagem foi efetivamente recebida, a operação de recepção regressa independentemente de ter sido ou não recebida uma mensagem;
* operação de envio */ void msg_send_nb (unsigned int destid, MESSAGE msg);
* operação de recepção */ void msg_receive_nb (unsigned int srcid, MESSAGE *msg, BOOLEAN *msg_arrival);
sincronização bloqueante - quando as operações de envio e de recepção contêm em si mesmas elementos de sincronização; a operação de envio envia a mensagem e bloqueia

até que esta seja efectivamente recebida a operação de recepção por seu lado só regressa quando uma mensagem tiver sido recebida;
* operação de envio */void msg_send (unsigned int destid, MESSAGE msg);
* operação de recepção */ void msg_receive (unsigned int srcid, MESSAGE *msg);

19. O que são sinais? O standard Posix estabelece que o significado de dois deles é mantido em aberto para que o programador de aplicações lhes possa atribuir um papel específico. Mostre como poderia garantir o acesso a uma região crítica com exclusão mútua por parte de dois processos usando um deles. Sugestão – Veja no manual on-line como se pode usar neste contexto a chamada ao sistema wait.

Sinal constitui uma interrupção produzida no contexto de um processo onde lhe é comunicada a ocorrência de um acontecimento especial. Este pode ser despoletado pelo kernel, pela ocurrencia de erros de hardware ou por condições especificas por software a partir do processo.
Tal como o processador no tratamento de excepções, o processo assume uma de três atitudes possíveis relativamente a um sinal
• ignorá-lo – não fazer nada face à sua ocorrência;
• bloqueá-lo – impedir que interrompa o processo durante intervalos de processamento bem definidos;
• executar uma acção associada – pode ser a acção estabelecida por defeito quando o processo é criado (conduz habitualmente à sua terminação ou suspensão de execução), ou uma acção específica que é introduzida (registada) pelo próprio processo em runtime.
Uma forma é de enviar cada sinal de entrada na região critica e saída, seria qualquer programa que tente aceder posta em wait.

20. Mostre como poderia criar um canal de comunicação bidireccional (full-duplex) entre dois processos parentes usando a chamada ao sistema pipe.

Na comunicação full-duplex é comum usarem-se 2 pipes. Um pipe redirecciona o stdout do P1 para o stdin do P2. O outro redirecciona o stdout do P2 para o stdin do P1. Podia-se utilizar só um pipe, mas seria mais ineficiente e complexo.

21. Como classifica os recursos em termos do tipo de apropriação que os processos fazem deles? Neste sentido, como classificaria o canal de comunicações, uma impressora e a memória de massa?

Um recurso é algo que um processo precisa para a sua execução. Os recursos tanto podem ser componentes físicos do sistema computacional (processadores, regiões de memória principal ou de memória de massa, dispositivos concretos de entrada / saída, etc.), como estruturas de dados comuns definidas ao nível do sistema de operação (tabela de controlo de processos, canais de comunicação, etc.), ou entre processos de uma mesma aplicação.
• recursos preemptable – quando podem ser retirados aos processos que os detêm sem que da resulte qualquer consequência irreparável à boa execução dos processos são por exemplo em ambientes multiprogramados, o processador, ou as regiões de memória principal onde o espaço de endereçamento de um processo está alojado;
• recursos non-preemptable – em caso contrário; são, por exemplo, a impressora, ou uma estrutura de dados partilhada que exige exclusão mútua para a sua manipulação.

Assim sendo o canal de comunicação e do tipo non-preemtable, as impressoras são do tipo non-preemtable e a memoria de massa pode ser do tipo preemtable ou non-preemtable.

22. Distinga as diferentes políticas de prevenção de deadlock no sentido estrito. Dê um exemplo ilustrativo de cada uma delas numa situação em que um grupo de processos usa um conjunto de blocos de disco para armazenamento temporário de informação.

Para evitar a ocorrência de deadlock basta que não ocorra uma das seguintes condições(visto que para acontecer deadlock tem de acontecer as 4 necessariamente):
• condição de exclusão mútua – cada recurso existente ou está livre ou foi atribuído a um e um só processo (a sua posse não pode ser partilhada);
• condição de espera com retenção – cada processo ao requerer um novo recurso mantém na sua posse todos os recursos anteriormente solicitados
• condição de não libertação - ninguém a não ser o próprio processo pode decidir da libertação de um recurso que lhe tenha sido previamente atribuído;
• condição de espera circular (ou ciclo vicioso) - formou-se uma cadeia circular de processos e recursos em que cada processo requer um recurso que está na posse do processo seguinte na cadeia.

23. Em que consiste o algoritmo dos banqueiros de Dijkstra? Dê um exemplo da sua aplicação na situação descrita na questão anterior.

O algoritmo do banqueiro é executado pelo sistema operacional quando um processo de computação requisita recursos.
O algoritmo impede o deadlock, ao negar ou adiar o pedido se ele determinar que aceitar o pedido pode colocar o sistema em um estado inseguro (onde um deadlock poderia ocorrer). Quando um novo processo entra em um sistema, ele deve declarar o número máximo de instâncias de cada tipo de recurso que não pode exceder o número total de recursos no sistema.
Exemplo: cada filósofo, quando pretende os garfos, continua a pegar primeiro no garfo da esquerda e só depois no da direita mas agora mesmo que o garfo da esquerda esteja sobre a mesa o filósofo nem sempre pode pegar nele;

24. As políticas de prevenção de deadlock no sentido lato baseiam-se na transição do sistema entre estados ditos seguros. O que é um estado seguro? Qual é o princípio que está subjacente a esta definição?

Define-se neste contexto estado seguro como uma qualquer distribuição dos recursos do sistema, livres ou atribuídos aos processos que coexistem, que possibilita a terminação de todos eles. Por oposição, um estado é inseguro se não for possível fazer-se uma tal afirmação sobre ele.
O principio subjacente e a diminuição da ocorrência de deadlock e para isso são necessárias as seguintes condições:
• necessário o conhecimento completo de todos os recursos do sistema e cada processo tem que indicar à cabeça a lista de todos os recursos que vai precisar – só assim se pode caracterizar um estado seguro;
• um estado inseguro não sinónimo de deadlock – vai contudo considerar-se sempre o pior caso possível para garantir a sua não ocorrência.

4 - Gestão de Memória

1 Descreva os diferentes níveis em que se estrutura a memória de um sistema computacional, caracterizando- os em termos de capacidade, tempo de acesso e custo. Explique, face a isso, que funções são atribuídas a cada nível.

Cache vai de KBs a poucos MBs, é rápida cara e volátil. Tem como função conter a cópia das posições de memória (instruções e operandos) mais frequentemente utilizados pelo processador num passado próximo.
Memória principal vai de centenas de MBs a alguns GBs, tem velocidade e preços médios e é volátil. Tem como função o armazenamento do espaço de endereçamento dos vários processos. Para garantir uma elevada taxa de utilização do processador este nº de espaços de endereçamento deve ser elevado.
Memória de massa tem capacidade de centenas de GBs, é lenta, barata e não volátil. Tem como função fornecer um sistema de ficheiros onde são salvaguardados mais ou menos permanentemente os dados e código. Tambe, pode ser usada como área de swapping, extensão da memória principal para que haja um maior nº de processos a coexistir num determinado instante.

2. Qual é o princípio que está subjacente à organização hierárquica de memória? Dê razões que mostrem porque é que a aplicação de tal princípio faz sentido.

Princípio da localidade de referencia; trata-se de uma constatação heurística sobre o comportamente de um programa em execução que estabelece que as referencias à memória durante a execução de um programa tendem a concentrar-se em fracções bem definidas do seu espaço de endereçamento durante intervalos mais ou menos longos.
Este tipo de organização baseia-se o pressuposto que quanto mais afastado uma instrução, ou um operando, está do processador, menos vezes será referenciado; nestas condições, o tempo médio de uma referência aproxima-se tendencialmente do valor mais baixo.

3. Assumindo que o papel desempenhado pela gestão de memória num ambiente de multiprogramação se centra, sobretudo, no controlo da transferência de dados entre a memória principal e a memória de massa, indique quais são as actividades principais que têm que ser consideradas.

transferência para a área de swapping do total ou parte do espaço de endereçamento de um processo quando o espaço em memória principal não é suficiente para conter todos os processos que coexistem.
- salvaguarda de um registo que indique as posições livres e ocupadas na memória principal.
- alocação de porções de memória principal para os processos que dela vão precisar, ou libertação qd não necessária.

4. Porque é que a imagem binária do espaço de endereçamento de um processo é necessariamente relocatável num ambiente de multiprogramação?

Uma vez que a situação de termos em memória principal processos bloqueados e na área de swapping processos prontos a executar, leva a uma menor eficiência na utilização do processador e a um desperdício do espaço de memória principal. Faz sentido, se a

memória principal não tiver espaço livre, transferir os processos bloqueados em memória principal para a área de swapping e transferir os processos prontos a executar na área de swapping para a memória principal. Por outro lado um processo pode precisar de muita memória para armazenar as suas variáveis e esgotar o seu espaço de endereçamento. Neste caso é necessário alocar mais memória e associá-la ao processo que a requisita.

5. Distinga linkagem estática de linkagem dinâmica. Qual é a mais exigente? Justifique a sua resposta.

Na linkagem estática todos os ficheiros objecto e bibliotecas do sistema são juntas num ficheiro único. Temos assim que todas as referências existentes no programa existem no espaço de endereçamento desde o início.
Na linkagem dinâmica as referências a funções de bibliotecas do sistema são substituídas por stubs, que determina a localização da função de sistema em memória principal ou a carrega em memória principal. Sendo em seguida a referência ao stub substituída pela referência da função, ou seja, em linkagem dinâmica não temos todas as referências satisfeitas ao início, elas vão-se satisfazendo ao longo do decorrer do programa, quando necessário.
A mais exigente é a linkagem estática, pois temos um maior espaço de endereçamento é quando há mudança de contexto e o processo passa ao estado blocked ou suspended-blocked em que vai haver transferência do espaço de endereçamento, como o espaço é grande e a memória de massa é lenta, demora-se um pouco.
A linkagem estática e mais existente em termos de quantidade de memoria para armazenar os programas mas o código e sequencial logo exige menos saltos entre posições e consequentemente reduz o acesso ao disco para resolver dependências. Assim sendo a dinâmica tornaria-se mais exigente.

6. Assuma que um conjunto de processos cooperam entre si partilhando dados residentes numa região de memória, comum aos diferentes espaços de endereçamento. Responda justificadamente às questões seguintes – em que região do espaço de endereçamento dos processos vai ser definida a área partilhada? – será que o endereço lógico do início da área partilhada é necessariamente o mesmo em todos os processos?
– que tipo de estrutura de dados em Linguagem C tem que ser usada para possibilitar o acesso às diferentes variáveis da área partilhada?

Uma vez que se trata de memória partilhada, temos que assumir que os dados partilhados podem tomar diferente tamanho ao longo da execução do programa. Como tal esses dados encontram-se na zona de definição dinâmica, região global;
- Não, pois o espaço ocupado pelo código e pela zona de definição estática pode ser diferente, o que resulta num endereço lógico inicial da zona de definição dinâmica diferente para os diversos processos. O tamanho das duas primeiras zonas (código e defenição estática) não é fixo, é definido pelo loader.
- Ponteiros. Na execução dinâmica o conteúdo do ponteiro é o endereço físico da variável para o qual aponta. Este endereço, e não o endereço lógico, é válido para todos os processos. Deste modo, o acesso às variaveis partilhadas deve ser feito usando ponteiros.

7. Distinga relativamente a um processo espaço de endereçamento lógico de espaço de endereçamento físico. Que problemas têm que ser resolvidos para

garantir que a gestão de memória num ambiente de multiprogramação é eficiente e segura?

Endereçamento físico refere-se à localização na memória principal do espaço de endereçamento do processo.
O endereçamento lógico refere-se à imagem binária do espaço de endereçamento do processo e é um endereçamento relativo relativo a um dado endereço, offset. Problemas que têm de ser resolvidos:
- conversão em runtime de endereçamento lógico para endereçamento físico;
- garantir que o endereçamento físico obtido na conversão anterior está dentro da gama de endereços do espaço de endereçamento do processo.

8. Caracterize a organização de memória designada de memória real. Quais são as consequências decorrentes deste tipo de organização?

(memória real)Neste tipo de organização existe uma correspondência biunívoca entre o espaço de endereçamento lógico e o espaço de endereçamento físico de um processo. Isto tem como consequências:
- limitação do espaço de endereçamento de um processo (não é possível em nenhum caso que o espaço de endereçamento de um processo seja maior que o tamanho da memória principal;
- contiguidade do espaço de endereçamento físico (é mais simples e eficiente supor que o espaço de endereçamento. é contíguo);
- área de swapping (o seu papel é assumir de arrecadação do espaço de endereçamento de todos os processos que não possam ser carregados em memória principal por falta de espaço.

9. Descreva detalhadamente o mecanismo de tradução de um endereço lógico num endereço físico numa organização de memória real.

```
if(end_logico>limite)
        gera_excepcao();    //erro
else
      end_fisico=end_logico+end_base;
```

- ocorre comutação de contexto
- registo base e limite são preenchidos de acordo com a entrada correspondente ao novo processo
- endereço base: endereço correspondente ao início do espaço de endereçamento físico do processo
- endereço limite: tamanho do espaço de endereçamento do processo

10. O que é que distingue a arquitectura de partições fixas da arquitectura de partições variáveis numa organização de memória real? Indique quais são as vantagens e desvantagens de cada uma delas.

Na arquitectura de partições fixas a memória principal é dividida num conjunto fixo de partições mutuamente exclusivas, mas não necessariamente iguais. Cada uma contém o espaço de endereçamento de um processo. Tem como vantagens o facto de ser simples de implementar (HW e estruturas de dados simples) e é eficiente (selecção é feita rapidamente).Tem como desvantagens a grande fragmentação interna e serve só para aplicações específicas.
Na arquitectura de partições variáveis toda a parte disponível de memória constitui à partida um bloco. Reserva-se sussessivamente regiões de tamanho suficiente para carregar o espaço de endereçamento dos processos que vão surgindo e liberta-se quando não é necessário. Para gerir a memória precisamos de 2 listas, lista das regiões ocupadas e das regiões livres. Para não existir risco de existirem regiões livres tão pequenas que não possam ser utilizadas e que tornem a pesquisa mais ineficiente, tipicamente a memória principal é dividida em blocos de tamanho fixo e a reserva é feita em entidades do tamanho desses blocos. Tem como vantagens uma menor fragmentação da memória, pois cada processo ocupa o espaço estritamente necessário. Tem como desvantagens o facto de uma pesquisa ser mais elaborada e inefeciente.
Nota: garbage collection – compactação do espaço livre agrupando as regiões num dos extremos da memória, exige a paragem de todo o processamento

11. A organização de memória real conduz a dois tipos distintos de fragmentação da memória principal.
Caracterize-os e indique a que tipo de arquitectura específica cada um está ligado.

Os 2 tipos de fragmentação da memória principal são partições fixas e partições variáveis.
Nas partições fixas a fragmentação é interna e a parte da partição interna que não contem espaço de endereçamento de processos é desperdiçada, não sendo usada para nada.
Nas partições variáveis a fragmentação é externa, as sucessivas reservas e libertações de espaço resultam em fragmentos de espaço livre tão pequenos que não podem ser utilizados. Esta fracção de memória principal desperdiçada pode em alguns casos atingir 1/3 do total de memória.

12. Entre os métodos mais comuns usados para reservar espaço em memória principal numa arquitectura de partições variáveis, destacam-se o next fit e o best fit. Compare o desempenho destes métodos em termos do grau e do tipo de fragmentação produzidos e da eficiência na reserva e libertação de espaço.

O next fit consiste em iniciar a pesquisa a partir do ponto de paragem da pesquisa anterior. É muito rápido a pesquisar e pode causar fragmentação interna pois se o espaço reservado for maior que o espaço de end do processo existe espaço desperdiçado. Mas este espaço desperdiçado faz com que na libertação do espaço reservado não existam fragmentos muito pequenos.
O best fit escolhe a região mais pequena disponível onde cabe todo o espaço de end. do processo. É mais lento a pesquisar e não há fragmentação interna, apenas externa.
Este último método resulta em mais memória desperdiçada, visto que tende a deixar fragmentos de memória livre demasiado pequenos para serem utilizados.

13. Caracterize a organização de memória designada de memória virtual. Quais são as consequências decorrentes deste tipo de organização?
Numa organização de memória virtual o espaço de endereçamento lógico e físico de um processo estão completamente dissociados. As consequências são as seguintes:
- pode ser estabelecido uma metodologia conducente à execução de processos em que o seu espaço de endereçamento pode ser maior que a memória principal;
- não contiguidade do espaço de endereçamento físico (o espaço de endereçamento de um processo dividido em blocos de tamanho fixo estão dispersos por toda a memória, procurando- se desta maneira obter uma ocupação mais eficiente do espaço disponível);
- área de swapping (é criada nesta uma imagem atualizada de todo o espaço de
endereçamento dos processos que coexistem correntemente, nomeadamente da sua parte variável).

14. Indique as características principais de uma organização de memória virtual. Explique porque é que ela é vantajosa relativamente a uma organização de memória real no que respeita ao número de processos que correntemente coexistem e a uma melhor ocupação do espaço em memória principal.
- não existe limitação para o espaço de endereçamento do processo (EEP) uma vez que a área de swapping funciona como extensão da memória principal, assim os processos que tenham parte do seu espaço de endereçamento na área de swapping, ou mesmo todo lá, não estão obrigatoriamente bloqueados, existindo assim um maior número de processos na memória principal;
- a parte mais inactiva de um EEP pode estar na área de swapping, deixando espaço na memória principal para outros processos.
Como o espaço de endereçamento físico de um processo não é necessariamente contíguo, não existem problemas de fragmentação.

15. O que é que distingue a arquitectura paginada da arquitectura segmentada / paginada numa organização de memória virtual? Indique quais são as vantagens e desvantagens de cada uma delas.
O que distingue uma arquitectura paginada da arquitectura segmentada/paginada é que na arquitectura seg/pag não se faz uma divisão do espaço de endereçamento lógico do processo (EEL) às cegas, pois 1º faz-se uma divisão do EEL em segmentos e depois esses segmentos são divididos em paginas.
Vantagens da arquitectura paginada:

- não conduz a fragmentação externa e a interna é desprezável (tem-se grande aproveitamento da memória principal);
- é geral, isto é, é independente do tipo dos processos que vão ser executados (nº e tamanho do seu EE);
- não exige requisitos especiais de hardware, pois a unidade de gestão de memória dos processadores actuais de uso geral já está preparado para a sua implementação.

Desvantagens da arquitectura paginada:
- acesso à memória mais longo;
- operacionalidade muito exigente, pois a sua implementação exige por parte do SO a
existência de um conjunto de operações de apoio e que são complexas e que têm de ser cuidadosamente estabelecidas para que não haja muita perda de eficiência).

Vantagens da arquitectura segmentada/paginada:
- iguais às 2 primeiras vantagens da arquitectura paginada;
- gestão mais eficiente da memória no que respeita às zonas de crescimento dinâmico
- minimização do nº de paginas que têm de estar residentes em memória em cada etapa de execução do processo.

Desvantagens da arquitectura segmentada/paginada:
- iguais a todas as desvantagens da arquitectura paginada;
- exige requisitos especiais de HW.

16. Descreva detalhadamente o mecanismo de tradução de um endereço lógico num endereço físico numa organização de memória virtual paginada.

O processador contem dois registos, o npag (numero de pagina) e o desloc(que nos dá o deslocamento dentro dessa pagina). O numero de pagina é enviado a unidade de gestão de memória que o vai somar ao endereço do registo base. O resultado é enviada para a memória principal, que vai à tabela de paginação, que vai informar a UGM, através do bit M/AS, se a pagina está ou não em memória principal, através do registo base, qual o numero de frame, e informar a ocupação ou não desta entrada (a não ocupação significa que ainda não foi reservado espaço na área de swapping para esta pagina), pelo bit O/L. O registo base irá fornecer o numero de frame e do processador vem o deslocamento, resultando no endereço físico, na memória principal. Há duas hipóteses de erro, uma caso não exista a pagina, e outra caso n exista o endereço. O primeiro é dado por subtrair AS ao bit M/AS, e o segundo é dado subtraindo L a O/L. Caso o bloco não esteja em memória, a UGM gera uma excepção por falta de bloco e a rotina de serviço à excepção deve por em marca acões que visem a transferência desse bloco da área de swapping para a memoria principal, e apos a sua conclusão a repetição da execução da instrução que produziu a referência. Todas estas operações são realizadas de uma forma completamente transparente ao utilizador que não tem consciência das interrupções introduzidas na execução do processo.

17. Explique porque é que hoje em dia o sistema de operação dos computadores pessoais supõe, quase invariavelmente, uma organização de memória de tipo memória virtual.

A implementação de uma organização de memória do tipo memória virtual não impõe limites ao tamanho do espaço de endereçamento do processo (EEP) e não dá quaisquer problemas em termos de fragmentação. Embora a sua implementação em HW seja mais complexa, o tempo de alocação e libertação de segmentos de memória seja maior, no global este tipo de implementação é mais vantajoso.

18. Porque é que a área de swapping desempenha papeis diferentes nas organizações de memória real e de memória virtual?

Porque ao contrário da organização de memória real, em que temos uma correspondência biunívoca entre espaço de end. lógico e espaço de end. físico, na organização de memória virtual estes 2 estão completamente dissociados, podendo assim um processo não estar bloqueado e ter parte do seu espaço de end. na área swapping.
Enquanto na organização de memória real o espaço ou está totalmente na área swapping ou na memória principal, por causa da correspondência biunívoca. Logo na organização de memória virtual convém ter uma imagem dos EEP que coexistem correntemente, para que qd seja necessária uma parte do EEP residente na área de swapping ela seja transferida para a memória principal pelo SO.

19. Considere uma organização de memória virtual implementando uma arquitectura segmentada /paginada. Explique para que servem as tabelas de segmentação e de paginação do processo. Quantas existem de cada tipo? Descreva detalhadamente o conteúdo das entradas correspondentes.

Numa arquitectura segmentada/paginada o espaço de end. lógico do processo é dividido em segmentos, segmentos estes que são divididos em páginas às quais corresponde um frame na memória principal. Logo temos uma tabela de segmentos e uma tabela de paginas, por processo.
Conteúdo da entrada da tabela de segmentos: endereço da tabela de paginação do segmento, bits de controlo.
Conteúdo da entrada da tabela de paginas: nº do frame em memória (as frames têm tamanho fixo), nº do bloco na área de swapping (memória de massa é dividida em blocos), informação de controlo.

20. Qual é a diferença principal existente entre a divisão do espaço de endereçamento de um processo em páginas e segmentos? Porque é que uma arquitectura segmentada pura tem pouco interesse prático?

A diferença reside no facto de que a divisão em segmentos não é uma divisão às cegas, pois tem em conta a estrutura modular na implementação de um programa. Ao contrário da paginada, que é feita às cegas, só coloca no início de uma nova página zonas funcionalmente distintas do espaçodend.d um process.
A arquitectura segmentada pura tem pouco interesse, pois ao tratar a memória principal como um espaço contínuo, leva ao uso de técnicas de reserva de espaço usadas na arquitectura de partições variáveis, o que origina grande fragmentação externa e desperdício de espaço. Os segmentos de crescimento contínuos tb dão problemas, pois os acréscimos podem não caber na localização actual, levando à sua transferência total para

outra região de memória ou se não houver espaço em memória principal é bloqueado o processo e o seu espaço de end. ou o segmento que não cabe são transferidos para a área swapping.

21. As bibliotecas de rotinas linkadas dinamicamente (DLLs) são ligadas ao espaço de endereçamento de diferentes processos em run time. Neste contexto, responda justificadamente às questões seguintes – que tipo de código tem que ser gerado pelo compilador para que esta ligação seja possível? – este mecanismo de ligação pode ser utilizado indiferentemente em organizações de memória real e de memória virtual?

– (compilador)Tem de ser gerado código de localização na memória principal das rotinas necessárias, ou que promova a sua carga em memória principal (stubs).
- Pode, pois quer numa ou noutra organização de memória o espaço de endereçamento da rotina linkada dinamicamente terá de estar ou ser carregada na memória principal, logo podendo haver correspondência biunívoca da organização real.

22. Quer numa arquitectura paginada, quer numa arquitectura segmentada / paginada, a memória principal é vista operacionalmente como dividida em frames, onde pode ser armazenado o conteúdo de uma página de um processo. Porque é que é conveniente impor que o tamanho de cada frame seja uma potência de dois?

Primeiro porque a memória principal tem um tamanho que é uma potência de 2, logo todos os frames têm o mesmo tamanho.
Segundo, pois sendo uma potência de 2, parte dos bits, por exemplo os mais significativos indicam-nos o nº do frame e os menos significativos a posição dentro do frame. O que simplifica a unidade de gestão de memória em termos de HW e SW.

23. Foi referido que nem todos os frames de memória principal estão disponíveis para substituição. Alguns estão locked. Incluem-se neste grupo aqueles que contêm as páginas do kernel do sistema de operação, do buffer cache do sistema de ficheiros e de um ficheiro mapeado em memória. Procure a aduzir razões que justifiquem esta decisão para cada um dos casos mencionados.

Locked significa que não estão disponíveis para substituição.
Como é lógico as páginas do kernel têm de estar locked, pois se não estivessem podiam ser substituidas (transferidas para a área swapping), logo se fossem precisas tinham de ser novamente transferidas para a memória principal e até talvez substituir uma página de outro processo, o que fazia perder algum tempo, podendo bloquear todo o sistema durante esse tempo.
Buffer cache e ficheiro mapeado em memória são ambos as técnicas usadas para uma maior rapidez de acesso, logo a sua substituição implicaria uma redução do tempo de acesso, o que vai contra o princípio com que eles foram implementados.

24. O que é o princípio da optimalidade? Qual é a sua importância no estabelecimento de algoritmos de substituição de páginas em memória principal?

O principio da optimalidade diz-nos que o frame que deve ser substituído é aquele que não vai ser mais usado, referenciado, ou se o for se-lo-á o mais tarde possível.
A sua importância é que ele é o princípio optimo de substituição, mas é não casual, pois tem de se conhecer o futuro para o usar. Logo tenta-se a partir do conhecimento passado prever o futuro para saber qual o frame que não vai ser usado ou vai ser referenciado tardiamente

25. O que é o working set de um processo? Conceba um método realizável que permita a sua determinação.

Carrega-se a 1ª e a última página do espaço de endereçamento do processo (correspondete ao início do código e ao topo da stack). Ao longo da execução do processo serão gerados pages-fault, sendo estes carregados, mas ao longo do tempo o número de pages-fault diminui até ser zero (devido ao princípio da localidade de referência). Aí temos o chamado working set do processo.

26. Dê razões que expliquem porque é que o algoritmo do relógio, numa qualquer das suas variantes, é tão popular. Descreva uma variante do algoritmo do relógio em que são consideradas as quatro classes de frames características do algoritmo NRU.

(algoritmo do relogio)É popular visto que as operações de fifo_in e fifo_out tornam-se num incremento de um ponteiro (módulo e nº de elementos da lista).

```
while (! frame_adequado)
{  if(Ref= =0)
        substituicao(pag_associada);
   else
        Ref=0;
        ponteiro + +    % nº elementos
}
```

27. Como distingue a estratégia demand paging da prepaging? Em que contexto é que elas são aplicadas?

Quando um processo é introduzido na fila dos processos Ready-to-run, mais tarde, em resultado de uma suspensão, é necessário decidir que páginas se vão carregar na memória principal.
Se não se carregar nenhuma página e se se esperar pelo mecanismo de page-fault para formar o working-set do processo temos demand paging.
Se se carregar a 1ª e a última página do processo da 1ª vez que passa a Ready-to-run ou as páginas residentes no momento da suspensão temos prepaging.

28. Assuma que a política de substituição de páginas numa dada organização de memória virtual é de âmbito global. Explique detalhadamente como procederia para detectar a ocorrência de thrashing e como procederia para a resolver.

Para detectar, o working set dos processos teria de que ser maior que o nº de frames unlocked disponíveis na memória principal. A solução seria ir suspendendo processos até que o problema, geração de page-fault, thrashing desapareça.