1.CONCEITOS INTRODUTORIOS

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

". Um sistema computacional pode ser encarado como um conjunto de recursos tais como processador, memória, ficheiros, dispositivo i/o, etc. Do ponto de vista bottom-up o s.o. não é mais do que um gestor de recursos do sistema, de forma a atender eficientemente as solicitações dos utilizadores. Do ponto de vista top-down o s.o transforma cada recurso físico num ou mais recursos virtuais, i.e transforma o sist. computacional numa máq. virtual que pode agora ser encarada de forma lógica. O s.o funciona deste modo como uma extensão do hardware, tornando-o mais fácil de manobrar por parte do utilizador .

- O que são chamadas ao sistema?

É uma função que faz o pedido do utilizador ao sistema operativo para efectuar uma operação de I/O. São interrupções geradas por software. Quando utilizador necessita de executar uma operação de I/O, gera uma system call, que vai “invocar” o sistema operativo, o sistema operativo vai verificar quem gerou a interrupção e qual a operação pertendida, se os dados estiverem correctos (operação válida e autorização para a efectuar), o sistema operativo realiza-a e no final devolve o controlo novamente para o utilizador.

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

Ambos permitem partilhar ficheiros e recursos entre sistemas computacionais diferentes. Num sistema SO de rede pretende-se criar a ilusão de que todos os recursos (impressoras) e ficheiros(NFS) da rede são locais, num SO distribuido Não se pretende criar essa ilusão, ou seja os recursos e ficheiros de outros SC são comsiderados remotos e só os seus próprios recursos e ficheiros é que são locais. O SO distribuido possui uma maior capacidade de processamento (maior velocidade computacional) e maior fiabilidade (tolerância a falhas), porque permite a execução de um programa noutro sistema computacional.

2.GESTAO DO PROCESSADOR

- 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 sejam garantidos. Quais são eles?

É necessário garantir que a execução dos processos não é afectada pelo instante, ou

local no código, onde ocorre a comutação; e que não são impostas quaisquer restrições relativamente aos tempos de execução, totais ou parciais, dos processos.

- O que é o scheduling do processador? Que critérios devem ser satisfeitos pelos algoritmos que o põem 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 acção que o scheduller exerce. O scheduller é o programa mais básico do SO, a sua função é controlar as transições de estado dos vários processos, de forma a q o sistema computacional seja eficientemente partilhado pelos vários processos. Os critérios a satisfazer numa politica de scheduling do processador são:

· Justiça: Qualquer processo ao longo de um intervalo de tempo considerado de referencia, deve ter direito à sua fracção do processador;

· Previsibilidade: O tempo de execução de um processo deve ser razoavelmente constante e independente da sobrecarga pontual a que o sistema computacional possa estar sujeito.

· Throughtput: Deve procurar maximizar-se o numero de processos terminados por unidade de tempo.

· Tempo de resposta: Deve procurar-se minimizar-se o tempo de resposta às solicitações feitas pelos processos interactivos;

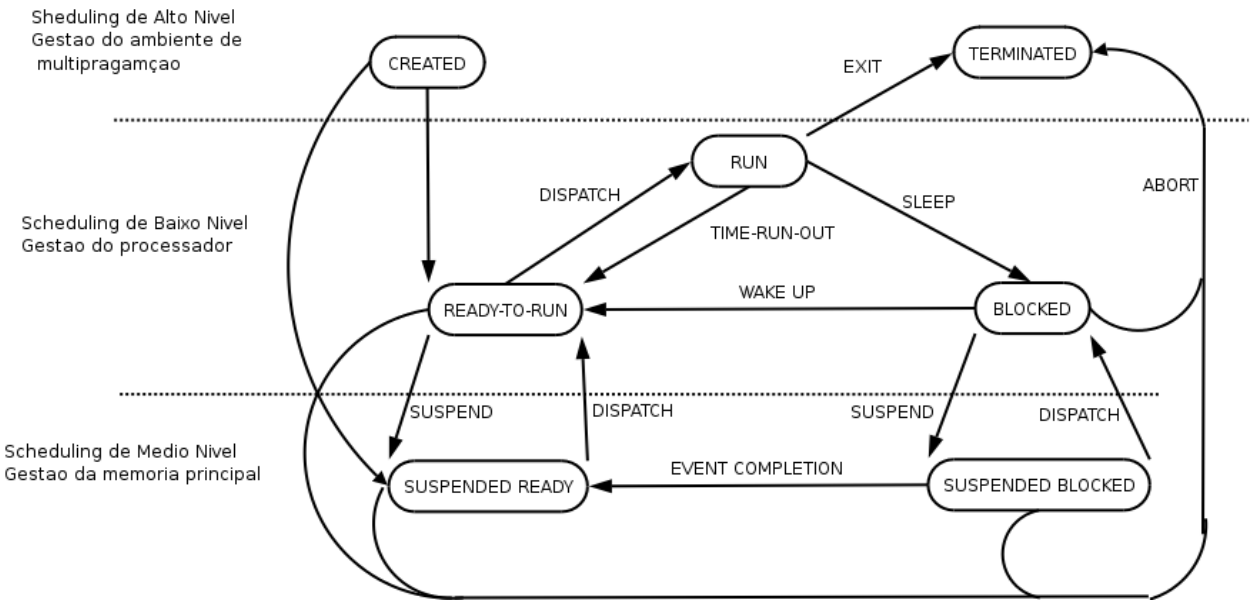
· Tempo de turnaround: Deve procurar minimizar-se o tempo de espera pelo completamento de um job no caso de utilizadores num sistema 'batch';

· Deadlines: Deve procurar garantir-se o máximo de cumprimento possível dos 'deadlines' impostos pelos processos em execução.

· Eficiência: Deve procurar manter-se o processador o mais possível ocupado com a execução dos processos dos utilizadores. Diminuição dos tempos mortos do processador.

Num sistema de multi-utilizador os mais importantes sao a Justiça, o Throughtput, o tempo de resposta, eficiência. Num sistema batch o tempo de turnaround é essencial tal o throughtput, e num sistema real a efeiciencia, e os deadlines são cruciais.

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



Baixo nível: Gestão dos processos em memória principal efectuada pelo processador. Neste nível os processos podem estar em três estados diferentes: RUN (Processo a ser executado), BLOCKED (processos á espera de eventos externos (I/O)) e READY-TO-RUN (processos que estão á espera de atribuição do processador para entrar em execução). Para que os processos transitem de estado existem quatro motivos: dispatch (um dos processos em espera é selecionado para execução), timer-run-out (o tempo atribuido ao processo em execução terminou), sleep (o processo decide que está impedido de proseguir ficando a aguardar a ocorrência de um acontecimento externo I/O), wake up (quando ocorre acontecimento externo que o colocou no estado BLOCKED).

Medio nível: Gestão dos processos entre memória principal e memória de 'swapping' efectuada pelo scheduling da memória principal. Neste nível os processos podem estar em 2 estados diferentes: SUSPENDED-READY (), SUSPENDED-BLOCK(). Para que os processos mudem de estado é necessário ocorrem uma das seguintes transições: suspend (suspensão o processo transferindo o seu espaço de endereçamento para a área de swapping), resume (retoma da execução de um processo, tranferindo o seu espaço de enderaçamento para a memória principal), event completion (ocorrência do acontecimento externo que o bloqueou).

Alto Nivel : A situação descrita até agora pressupõe que os processos são eternos, existindo permanentemente enquanto o sistema computacional estiver operacional. À parte de alguns processos de sistema, porém, este não é o caso mais frequente. Os processos são em geral criados, têm um tempo de vida mais ou menos longo e terminam. Um outro aspecto importante é controlar o grau de multiprogramação, que deve ser o maior possível, mas garantindo sempre um equilíbrio adequado entre o serviço aos processos que coexistem e a ocupação do processador.

Se houver uma grande necessidade de o utilizador da máquina usar muitos processos ao mesmo tempo e cujo "tempo médio de vida seja mais ou menos longo", pode haver a necessidade de existir a zona de swapping (scheduling de medio nível – Gestão da mem-principal.)

Assim, uma vez q a mem principal por muito grande q seja é necessariamente finita. Quando estamos n1 ambiente multiprogramado, em q o nº de processos q corrente/ existem, é mt gr/, ela acaba por ser um factor limitativo ao seu crescimento. Então para solucionar esta situação, costuma-se criar em mem massa uma extensão amem principal, geral/ designada por 'area de swapping', q funciona como ("arrecadação") área de aramzenamento secundária (dos processos q corrente/ e concurrente/ existem para ser processados). Liberta-se assim, espaço em mem principal para execução de processos que, de outro modo, não poderiam existir, e potencia-se portanto, um aumento da taxa de utilização do processador.

- 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.

O kernel é responsável pelo tratamento das interrupções e por agendar a atribuição do processador e de muitos outros recursos do sistema computacional.

Assim, para que o sistema de operação funcione no modo privilegiado (Kernel Running), com total acesso a toda a funcionalidade do processador, as chamadas ao sistema associadas, quando não despoletadas pelo próprio hardware, são implementadas a partir de instruções trap. Cria-se, portanto, um ambiente operacional uniforme, em que todo o processamento pode ser encarado como o serviço de excepções. Nesta perspectiva, a comutação de processos pode ser visualizada globalmente como uma vulgar rotina de serviço à excepção, apresentando, porém, uma característica peculiar que a distingue de todas as outras: normalmente, a instrução que vai ser executada, após o serviço da excepção, é diferente daquela cujo endereço foi salvaguardado ao dar-se início ao processamento da excepção.

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

perspectiva sistémica

– critérios orientados para o utilizador – estão relacionados com o comportamento do sistema de operação na perspectiva dos processos ou dos utilizadores;

– critérios orientados para o sistema – estão relacionados com o uso eficiente dos recursos do sistema de operação;

• perspectiva comportamental

- critérios orientados para o desempenho – são quantitativos e passíveis, portanto, de serem medidos;
- outro tipo de critérios – são qualitativos e difíceis de serem medidos de uma maneira directa.

- 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.

Non-preemptive scheduling – quando, após a atribuição do processador a um dado processo, este o mantém na sua posse até bloquear ou terminar.A transição timer-run-out não existe neste caso. É característico dos sistemas operativos de tipo batch.

Preemptive scheduling – quando o processador pode ser retirado ao processo que o detém; tipicamente, por esgotamento do intervalo de tempo de execução que lhe foi atribuído, ou por necessidade de execução de um processo de prioridade

mais elevada. É característico dos sistemas operativos de tipo interactivo.

Num sistema de tempo real a politica de schedulling deverá ser mista. O objectivo principal de um sistema de tempo real e conseguir atender certos elementos externos respeitando o deadline por eles imposto. Logo para estes processo e imperativo que a estratégia seja non-preemptive. No entanto se outro processo estiver em execuçao e este precisar de ser executado é necessário que a estratégia para o outro processo seja preemptive. Assim a solução ideal é associar prioridades estáticas aos diferentes processos e defenir que para os processos de maior prioridade a politica de shedulling e non-preemptive e para os processos com menor prioridade a politica de schedulling e preenptive.(???)

- O que é o aging dos processos?

Em qq sist onde se mantém processos à espera enquanto ele faz a alocacao de recursos e shuduling é possivel adiar indefinidamente o shudulig de um proc enquanto outros recebem toda a atencao do syst(adiamento indifinido). Isto pode acontecer qd os recursos sao organizados com base numa atribuicao dinamica de propriedades, já que é possivel que um proc tenha de esperar indefinidamente que outros processos de prioridade mais elevada sejam servidos. Sendo assim o Aging consiste em aumentar linearmente a priopridade de um processo que podera eventualmente exceder a prioridade de todos os outros processo exitentes no sist.

Usando prioridade dinamica no shedulling alguns processos correm o risco de adiamento indefinido, para evitar que isto aconteça usa-se o tempo que o preocesso está á espera na fila de espera de READY-RUN para calcular a prioridade.

2.COMUNICAÇÃO ENTRE PROCESSOS

- 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?

A região crtica de código é a execução por parte do processador do código que deseja

aceder uma região critica de dados

Uma região critica é código que manipula recursos ou dados partilhados por vários processos ao mesmo tempo.
As propriedades das primitivas de acesso são:

· Garantia efectiva de imposição da exclusão mútua. Só pode aceder um processo de cada vez á região crítica.

· Um processo fora da região critica não pode impedir outros de lá entrar.

· Os processos que aguardam acesso á região crítica não podem bloquear um processo que lhe acede no momento.

· Não pode ser adiado indefinidamente a possibilidade de acesso à região crítica a um qualquer processo.

· A solução não pode ser feita partindo do presuposto do tempo de execução de um processo. Independente da velocidade de execução.

· O processo que tem a posse da região crítica não pode lá ficar indefinidamente. Tempo de permanência na região crítica tem que ser finito.

- Distinga deadlock de adiamento indefinido.

deadlock'-quando dois ou mais processos ficam a aguardar eternamente a sua entrada nas respectivas regiões críticas, esperando acontecimentos, que se pode demonstrar, que nunca irão acontecer; o resultado é por isso um bloqueio das operações;

•adiamento indefinido-quando um ou mais processos competem pelo acesso às respectivas regiões críticas e, devido a uma conjunção de circunstâncias em que surgem continuamente processos novos (de prioridade mais elevada, por exemplo) que o(s) ultrapassam nesse desígnio, o seu acesso é sucessivamente adiado; está-se por isso perante um impedimento real à sua continuação.

- 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? Qual é a vantagem principal que boa parte das soluções 'hardware' apresenta sobre as soluções 'software'?

• soluções software – são soluções que, quer sejam implementadas num monoprocessador, quer num multiprocessador com memória partilhada, supõem o recurso em última instância ao conjunto de instruções básico do processador;

ou seja, as instruções de transferência de dados de e para a memória são de tipo standard: leitura e escrita de um valor; a única suposição adicional diz respeito ao caso do multiprocessador, em que a tentativa de acesso simultâneo a uma mesma posição de memória por parte de diferentes processadores é necessariamente serializada por intervenção de um árbitro;

• soluções hardware – são soluções que supõem o recurso a instruções especiais do

processador para garantir, a algum nível, a atomicidade na leitura e subsequente escrita de uma mesma posição de memória; são muitas vezes suportadas pelo próprio sistema de operação e podem mesmo estar integradas na linguagem de programação utilizada.

A principal vantagem das soluções de 'hardware' em relação ás de 'sofware' é o facto de os processos que aguardam entrada na região crítica ficam bloqueados, ou seja, não ficam em 'busy-waitng' (processos interveniente aguradam entrada na região crítica no estado activo) e por conseguinte não estão a consumir tempo de processador, isto faz com que as soluções de hardware aumentem a eficiência e eliminem os constrangimentos no estabelecimento no algoritmo de scheduling, porque um processo de maior prioridade não consegue bloquear um processo de menor prioridade que se encontre aceder á região crítica.

- 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.

Busy-Waiting é um problema comum às soluções software e ao uso de flags de locking, implementadas a partir de instruções de tipo read-modify-write, é que os processos intervenientes aguardam entrada na região crítica no estado activo

Tem como inconvenientes a perda de eficiência, o processo pode ser calendarizado para execução mas não executar nada ou pode mesmo provocar constrangimentos no processo de shedulling, bloqueando em execução e entrar numa sistuação de deadlock.Em sistemas computacionais multiprocessador com memória partilhada, e mais concretamente no caso de multiprocessamento simétrico, o problema de busy waiting não é tão crítico.

- 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 semaforo é um dispositivo que usa uma variavel tipo inteiro (exemplo: val) que não pode tomar valores negativos e sobre o qual só podem ser efectuadas duas operações atomicas, que são o sem_up (se existirem processos bloqueados são acordados , caso contrário a variavel val é incrementada) e o sem_down (se o valor de val for zero o processo que executou o sem_down é bloqueado, caso contrário a váriavel val é decrementada)

Para utilizar semaforos para aceder a uma região crítica, inicializa-se um semaforo (por exemplo: acesso) com o valor da váriavel val igual a 1. Se o processo P1 pretende aceder à RC executa a função sem_down(acesso) colocando a variavel val a zero sem ficar bloqueado executando o código que necessitar dentro da RC, se outro processo P2 quiser aceder à RC efectua a função sem_down(acesso), como o valor de val é zero o processo P2 fica bloqueado á espera que P1 liberte a região critica. Assim que o processo P1 já não necessitar da RC efectua a função sem_up(acesso), como o valor val é zero e existe o processo P2 em espera este é acordado e entra na RC, o valor de val continua a zero.

- 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.

Um monitor é um dispositivo de sincronização, proposto de uma forma independente por Hoare e Brinch Hansen, que pode ser concebido como um módulo especial, suportado pela linguagem de programação [concorrente] e constituído por uma estrutura de dados interna, por código de inicialização e por um conjunto de primitivas de acesso.

Quando as estruturas de dados são implementadas com monitores, a linguagem de programação garante que a execução de uma primitiva do monitor é feita em regime de exclusão mútua. Assim, o compilador, ao compilar um monitor, gera o

código necessário para impor esta situação. Um thread entra no monitor por invocação de uma das suas primitivas, o que

constitui a única forma de acesso à estrutura de dados interna. Como a execução das primitivas decorre em regime de exclusão mútua, quando um outro thread está no seu interior, o thread é bloqueado à entrada, aguardando a sua vez.

A sincronização entre threads é gerida pelas variáveis de condição. Existem duas operações que podem ser executadas sobre uma variável de condição

wait – o thread que invoca a operação é bloqueado na variável de condição argumento e é colocado fora do monitor para possibilitar que um outro thread que aguarda acesso, possa prosseguir;

signal – se houver threads bloqueados na variável de condição argumento, um deles é acordado; caso contrário, nada acontece.

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

Conceptualmente, o principal problema com o uso dos semáforos é que eles servem simultaneamente para garantir o acesso com exclusão mútua a uma região crítica e para sincronizar os processos intervenientes. Assim, e porque se trata de primitivas de muito baixo nível, a sua aplicação é feita segundo uma perspectiva bottom-up (os processos são bloqueados antes de entrarem na região crítica, se as condições à sua continuação não estiverem reunidas) e não top-down (os processos entram na região crítica e bloqueiam, se as condições à sua continuação não estiverem reunidas).

A primeira abordagem torna-se logicamente confusa e muito sujeita a erros, sobretudo em interacções de alguma complexidade, porque as primitivas de sincronização podem estar dispersas por todo o programa.

Uma solução é introduzir ao nível da própria linguagem de programação uma construção [concorrente] que trate separadamente o acesso com exclusão mútua a uma dada região de código e a sincronização dos processos.

- 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.

A politica de prevenção de deadlock no sentido estrito tem como finalidade a negação de uma das 4 condições que originam deadlock para deste modo eliminá-lo:

• negação da condição de exclusão mútua – só pode ser aplicada a recursos passíveis de partilha em simultâneo;

• negação da condição de espera com retenção – exige o conhecimento prévio de todos os recursos que vão ser necessários, considera sempre o pior caso possível (uso de todos os recursos em simultâneo);

• imposição da condição de não libertação – ao supor a libertação de todos os recursos anteriores quando o próximo não puder ser atribuído,atrasa a execução do processo de modo muitas vezes substancial;

• negação da condição de espera circular – desaproveita recursos eventualmente disponíveis que poderiam ser usados na continuação do processo.

Sendo a prevenção no sentido estrito muito radical, recorre-se ás politicas de prevenção no sentido lato que se baseiam numa politica de monitorização de atribuição de recursos, tendo como base o número de recursos que cada um pode vir a precisar e os recursos que se encontram disponíveis).(???)

- 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-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.

Convém notar o seguinte

• é 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 é sinónimo de deadlock – vai, contudo, considerar-se sempre o pior caso possível para garantir a sua não ocorrência.

4.Gestão de memória

- O que distingue fundamentalmente uma organização de memória virtual de uma organização de memória real? Qual é o papel desempenhado pela área de 'swapping' em cada uma delas?

Numa memória real o espaço de endereçamento de um processo em execução é necessário estar totalmente em memória principal, numa memória virtual não é necessário conter todo o espaço de endereçamento do processo em execução na memória principal. Por este motivo para uma memória principal com o mesmo tamanho é possivel colocar mais processos em memória principal do que no caso de uma organização real, ou seja, na memória virtual só se têm na memória principal apenas as partes necessárias á execução do processo e não a sua totalidade, enquanto que todo o espaço de endereçagem é mantido na memória secundária (área de swapping), as partes não presentes na memória principal são carregadas á medida que forem necessárias á execução do processo.
No caso da memória virtual não existe limitação do espaço de endereçamento fisico do processo, no caso real existe limitação do tamanho do espaço de endereçamento devido à memória principal ser menor que a de massa.

Para implementar uma memória virtual é necessário os seguintes requesitos:
Suporte de hardware para a gestão da memória (MMU);
Software para movimentação de partes do processo entre a memória principal e a memória secundária.

A vantagem do uso de memória real em relação a uma virtual é quando temos sistemas de tempo real e e é necessário que a execução de um determinado processo seja rápido, se tivermos todo o processo em memória principal é mais rápido, pois não é necessário estar a aceder á área de swapping caso ocorra uma page fault.
Um exemplo é o de monitorização da temperatura de uma caldeira que necessita de uma resposta rápida por parte do sistema.

A área de swapping no caso da memória real serve como arrecadação e existe uma correspondência de 1 para 1 do espaço de endereçamento do processo.

A área de swapping no caso da memória virtual serve para manter todo o espaço de endereçamento de um processo, esse espaço pode estar dividido em três tipos diferentes: paginada, segmentada e paginada segmentada.

- Qual é a diferença entre uma organização de memória virtual segmentada de uma paginada? Quais são as vantagens e inconvenientes de cada uma delas?

Numa organização de memória virtual paginada, a dimensão do bloco base de armazenamento em memória principal é fixa, enquanto que numa organização de memória virtual segmentada a dimensão dos vários blocos é variavel.
Na paginada, a partição do espaço de endereçamento reflecte apenas aspectos estruturais da própria memória física e resulta de uma divisão automática em blocos de comprimento fixo realizada na linkagem.

Na segmentada, a partição do espaço está directamente associada à organização do programa (código, stack, variáveis, etc...) estabelecida pelo programador, cada segmento corresponde a um módulo distinto.

A Segmentada tem a vantagem de cada segmento poder crescer ou diminuir separadamente, facilita a partilha de procedimentos e dados entre processos. A desvantagem é que tal como as partiçoes de mem real dinamicas sofre de fragmentaçao externa.

A virtual tem a vantagem de não ser necessario colocar todo o bloco de código do processo em memória principal, basta a página correspondente ao código que está a ser

necessário no momento. A desvantagem é que tal como as partiçoes de tamanho fixo sofre de fragm interna.

- 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 é necessário colocar todo o espaço de endereçamento do processo em memória principal, porque só são colocadas as partes estritamente necessárias de cada processo (aumenta a taxa de utilização do processador).
É necessário existir hardware que faça a tranferencia dos endereços virtuais para os endereços físicos (MMU) e software para movimentação de partes do processo entre a memória principal e secundária.

Podem estar mais processos em memória principal do que no caso de memória real porque ao contrário da real em que existe uma relação de 1 para 1 no espaço de endereçamento do processo neste caso essa relação é de muitos para 1, como só são colocados os pedaços necessários à execução do processo é possivel colocar lá mais processos, dai se tira que a ocupação em memória principal é mais bem aproveitada.

Esta vantagem é fundamentada pelo príncipio da localidade de referencia ,ou seja, as instruções de um processo e respectivos dados tendem a concentrar-se no tempo e no espaço.

Por exemplo, para a mesma memória principal, no caso em da real se tivermos dois processos cuja soma do espaço de endereçamento seja superior ao espaço físico da memória, sempre que quisermos efectuar a troca de um para o outro é necessário aceder á área de swapping, no caso da memória virtual já é possivel conter os pedaços estritamente necessários de cada processo carregados na memória principal não sendo necessário aceder á área de swapping para fazer a comutação de processos (só é necessário aceder á área de swapping quando ocorre uma page fault ou o conteudo da página em memória principal ter sido modificado e ser necessário salvaguarda-lo).

Concluindo, um sistema de memória virtual permite a multiprogramação de forma mais eficaz e liberta o programador da limitação da memória física.

5. Gestão de dispositivos de entrada-saída

- Porque é que num ambiente multiprogramado é fundamental desacoplar a comunicação dos processos utilizadores com os diferentes dispositivos de entrada-saída?

Para não ficarem em busy waiting.

Devido à diferença de velocidade entre os processos e os dispositivos de entrada/saida.

- Que papel desempenham os 'device-drivers' no estabelecimento de um interface uniforme de comunicação com os dispositivos de entrada-saída?

Os device drives (controlador) permitem criar uma interface abstracta com os dispositivos de I/O e a construção de software independente do dispositivo, que implementam operações comuns a todos os dispositivos de I/O. Obtem-se assim uma

plataforma uniforme para o software dos utilizadores. A função de um device driver consiste pois, em aceitar pedidos genéricos do software, por exemplo ler/escrever um bloco, e providenciar a sua execução pelo dispositivo.

Os device drives servem para independentemente do dispositivo de entrada/saida ele possa ser acedido através de funções pré-estabelecidas. Os device drives tiram o peso ao programador de como funciona o dispositivo e possa usar funções já conhecidas.

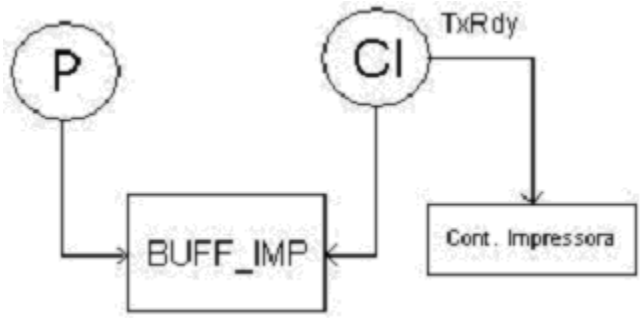
- O que distingue dispositivos de tipo caracter de dispositivos de tipo bloco? Descreva de uma maneira funcional como se desenvolve a comunicação entre um processo utilizador e um dispositivo de tipo caracter.

Dispositivos de Blocos: estes dispositivos armazenam informação em blocos de tamanho fixo, cada um numa localização bem definida. A principal propriedade destes dispositivos é a de permitir a leitura/escrita de cada bloco independentemente de todos os outros. Os discos e as tapes são dispositivos deste tipo.

Dispositivos de Caracteres: estes dispositivos processam a informação em grupos de caracteres, sem ter em atenção alguma estrutura de tipo bloco. Estes dispositivos não são endereçáveis e não permitem operações de pesquisa. As impressoras, os terminais de vídeo e os teclados são dispositivos deste tipo.

Os de caracter transmitem informação com mensagens com o tamanho fixo de um byte e ainformação não pode ser pesquisada, os dispositivos de bloco transmitem a informação com mensagens com tamanho de um bloco com vários bytes e é possivel aceder a um bloco independentemente dos outros, ou seja, premite a pesquisa indexada.

A figura abaixo descreve de uma maneira esquemática a situação proposta.



O processo P é o processo utilizador que em momentos particulares da sua execução pretende escrever na impressora linhas de texto. Para isso, socorre-se da chamada ao sistema

- void escreve_linha (char *linha)

que transfere para o buffer de comunicação BUFF_INP a linha em questão.

O processo CI é um processo de sistema activado pela interrupção TxRdy do registo

de transmissão do controlador da impressora. Quando é acordado, este processo procura ler um carácter, pertencente a uma linha armazenada em BUFF_IMP, e escrevê-lo no registo de transmissão da imperssora. Para isso socorre-se das duas chamadas ao sistema seguintes:

- char le_caracter ()

- void transm_caracter (char car)

Em termos gerais, a interacção estre dois processos pode ser descrita da seguinte forma: Sempre que tiver um alinha para imprimir, o processo P invoca o procedimento escreve_linha que transfere para o buffer de comunicação BUFF_IMP a linha em questão. O seu regresso é imediato, desde que haja espaço de armazenamento suficiente, caso contrário, o processo bloqueia. Na situação particular em que o buffer de comunicação esteja inicialmente vazio, o procedimento escreve_linha realiza a acção suplementar de transferir o primeiro carácter para o registo de transmissão do controlador da impressora, procedimento transm_caracter, despontando assim o mecanismo de interrupções que vai acordar sucessivamente o processo CI, até que toda a informação tenha sido efectivamente transferida.

Por sua vez o processo CI, quando é acordado pela interrupção, invoca a função le_caracter para recolher um carácter previamente armazenado em BUFF_IMP e , caso exista algum escreve-o no registo de transmissão do controlador da impressora, procedimento transm_caracter. Na situação particular em que o buffer de comunicação esteja inicialmente cheio a função le_caracter realiza a acção suplementar de tentar acordar o processo P eventualmente bloqueado a aguardar espaço de armazenamento disponível no buffer.

- Durante a operação normal de um sistema computacional, o acesso à memória de massa é quase constante. Como o seu tempo de acesso é tipicamente ordens de grandeza mais lento do que o acesso à memória principal, há o risco do desempenho global do sistema de operação ser fortemente afectado pelo 'engarrafamento' que daí resulta. Apresente sugestões de como minimizar este efeito.

Utilização da DMA.

Politicas de limpeza de páginas em memória principal

O uso de dois buffers, um para páginas modificadas e delas não modificadas.

6. Gestão da memória de massa

- Porque é que sobre a abstracção da memória de massa, concebida como um 'array' de blocos para armazenamento de dados, se torna fundamental introduzir o conceito de sistema de ficheiros? Quais são os elementos essenciais que definem a sua arquitectura?

O sistema de ficheiros permite abstrair o utilizador da organização da memória de massa, fornecendo-lhe apenas algumas funções de acesso aos ficheiros. (leitura, escrita, criação, eliminação, truncar, procura) Um sistema de ficheiros é composto por duas partes distintas: ficheiros e directórios (um directório é um ficheiro com atributos especiais).

Os ficheiros contém os dados, os programas e toda a informação que se pertende armazenar.

Os directórios servem para estruturar a organização dos ficheiros.

Um ficheiro é uma forma de guardar dados de uma forma organizada.

Um ficheiro é caracterizado pelos seguintes atributos:

· Nome: Nome simbólico para o utilizador reconhecer o ficheiro

· Tipo: Se o ficheiro é do tipo objecto, executável, código fonte, batch, texto, arquivos, etc.

· Localização: É um ponteiro para o dispositivo onde o ficheiro se encontra e a localização dentro desse dispositivo.

· Tamanho: O tamanho do ficheiro (em bytes, palavras ou blocos), e possivelmente o tamanho máximo permitido.

· Protecção: Informação de acesso ao ficheiro por parte dos diversos utilizadores.

· Tempo, data e dono: Data de criação, modificação e utilização.

- Em que consiste o problema da consistência da informação num sistema de ficheiros? Que razões levam à sua ocorrência? Como é que ele pode ser minimizado?

Tem haver com o problema resultante do facto de durante operações de criação, escrita, e outras operaçãos que alterem os ficheiros, o sistema falhe por alguma razão e a operação não seja totalmente realizada, por exemplo, pode ser sinalizado que um dado bloco de dados está a ser utilizado por um ficheiro mas como a operação foi interrompida, não foi possivel lá escrever a informação. Pode ser minizado fazendo regularmente uma verificação de consistência do sistema (scandisk), ou seja, percorrer todos os nós i e respectivos blocos de dados e verificar se estão correctamente. Uma solução utilizada também pode ser a realização de um diário, ou seja, sempre que se efectua uma operação que pode levar a uma situação de inconsistência do sistema, os vários passos dessa operação são guardados no diário e se no final forem bem executados, são sinalizados como tal, deste modo sempre que o sistema se desligue por uma razão anómala, basta ir ao diário e ver se no momento estava a ser efectuada alguma operação crítica e se foi bem realizada, se não foi finalizada o sistema realiza a operação inversa, ou seja, os passos que já tinham sido realizados são agora desfeitos.

- Descreva justificadamente os diversos papeis desempenhados pela memória de massa num sistema computacional. Dê razões que expliquem porque é que ela deve ser constituída por mais do que uma unidade de disco magnético.

Armazenamento de dados (ficheiros e directórios).

Area de swapping: extensão da memória principal, de forma a aumentar a taxa de utilização do processador e a libertar o programador da limitação da memória física, através da utilização da memória virtual.

Para não se perderem dados importantes quando um disco falhar, usando vários discos com cópias iguais. Quando é necessário armazenar muita informação.