- 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.

- 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?

Esta tabela é fundamental num ambiente de multiprogramação, de modo a que exista continuidade na execução do processo quando lhe for novamente atribuido o processador, transmitindo assim ao utilizador a ilusão do processo nunca ter sido interrompido. A PCT permite efectuar uma melhor gestão dos processos devido ás informações nela contidas.
Na PCT é utilizada para armazenar todo o contexto referente a um processo interrompido, copiando-o da sua stack no momento da interrupção.

Os campos que existem na PCT são:

- Identificadores: Do processo, do pai do processo e do utilizador a que o processo pertence;
- Caracterização do espaço de endereçagem: Sua localização (em memória principal ou na área de 'swapping'), de acordo com o tipo de organização de memória estabelecido;
- Contexto do processador : valores de todos os registos internos do processador no momento em que se deu a comutação do processo. Salvaguarda dos registos internos;
- 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 outra informação associada.

Convém que a PCT esteja armazenada constantemente na memória principal, uma vez que possiblita uma troca mais rápida e eficiente entre os diversos processos em uso.

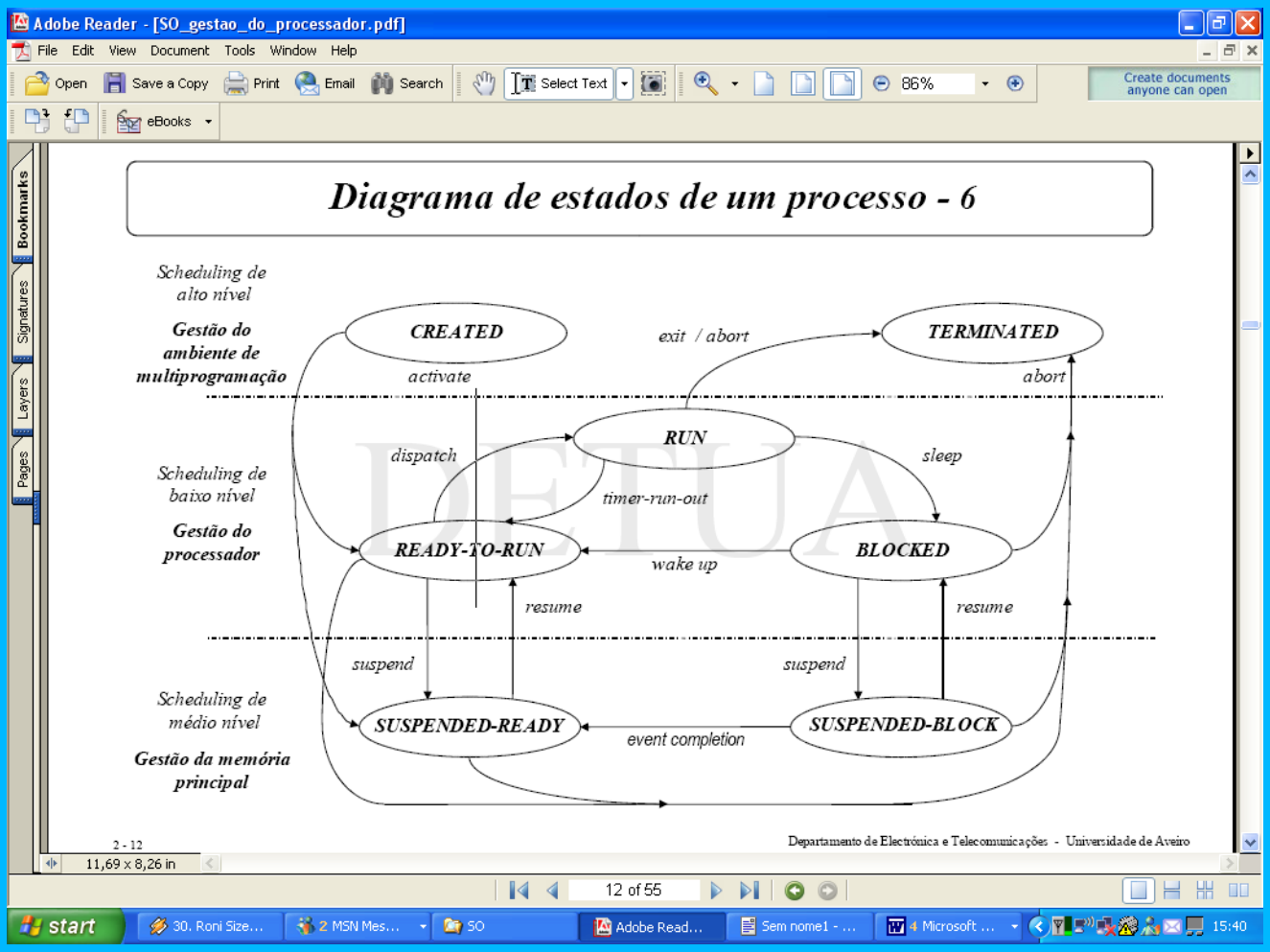
- O que é o *scheduling* do processador? Que critérios devem ser satisfeitos pelos algoritmos que o põem em prática? Quais são os mais importantes num sistema multiutilizador de uso geral, num sistema de tipo *batch* e num sistema de tempo real?

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 programas (ie, 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 (RAM) n1 SC, por muito grande q seja é necessariamente finita. Quando estamos n1 ambinete 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.

- 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?

(???)

- 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*.

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

a *comutação de contexto* é simulada pela activação e desactivação dos processadores virtuais e é controlada pelo seu *estado*;
(???)

- 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.

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

Prioridade estática: Os processos são agrupados em classes de prioridade fixa, de acordo com a sua importância relativa. Existe um elevado grau de probabilidade de ocorrência de adiamento indefinido. Para evitar o adiamento indefindo, quando o processo é criado é-lhe atribuida um dado nível de prioridade e sempre que esse processo é executado essa prioridade é decrementada, sendo reposta no valor inicial quando atinge o valor minimo. Disciplina utilizada em sistemas de tempo real.

Prioridade dinâmica: A prioridade dinâmica consiste na definição de classes de prioridade de carácter funcional, entre os quais os processos transitam de acordo com a ocupação da última ou últimas janelas de execução. Disciplina usada em sistemas de multi-utilizador.

- 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.

Existem varias classes de processos de forma a conseguir cumprir os critérios de schedulling, defenidos como mais importantes para um sistema mutiutilizador. Ou seja as diferentes classes tem como objectivo aumentar ao máximo a eficiência, reduzir o tempo de resposta, conseguir executar o maior numero de processos possivel no menor tempo possivel(trhoughtput), e aplicar o critério da justiça. (???)

- 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.(???)

- 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?

(???)

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

SCHED_OTHER – classe formada pelos processos restantes, o processador só é atribuído a processos desta classe se não houver outro tipo de processos prontos a serem executados;
• as classes *SCHED_FIFO* e *SCHED_RR* estão associadas a processamento de tempo real e a processos de sistema e o valor das suas prioridades é fixo; • a classe *SCHED_OTHER* está associada aos processos utilizador, a uma politica de schedulling dinamico, usando um algoritmo de créditos que calcula a prioridade a dar a cada processo.

- 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.

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.

- 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 admite dois conceitos, o conceito de pertença de recursos: armazenamento do espaço de endereçamento, e comunicação com os devices de I/O, e o conceito de fio de execução, que contem o program counter, os registos do processador, e a stack. Estes conceitos podem ser vistos separados e quando o fazemos o thread, é a parte do fio de execução dentro do contexto de um processo. (???)

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

Vantagens:
--- *maior simplicidade na decomposição da solução e maior modularidade na implementação*
--- melhor gestão de recursos do sistema computacional
--- *eficiência e velocidade de execução*

(O RESTO NÃO INTERESSA)