Vou fazer exame de Sistemas Operativos!

(em construção)

Fsm, Fev 2004-02-12

Este documento contém informação relevante para quem pretende ter sucesso no exame da disciplina de Sistemas Operativos (SO) do 3º ano dos cursos de LESI e LMCC da UM. Discute-se em particular a atitude a tomar perante as questões do exame e o tipo de resposta a dar por forma a obter a cotação máxima a cada pergunta. São apenas apresentados exemplos de perguntas teóricas, mas a componente prática deve ser tratada da mesma maneira.

A disciplina de Sistemas Operativos tem como objectivo geral contribuir para uma sólida formação em informática, nomeadamente ajudando a perceber o funcionamento de um sistema como um todo: requisitos e expectativas dos utilizadores, carga associada aos diferentes tipos de aplicações, estratégias de gestão de recursos e respectivos custos e benefícios, recapitulando ainda questões como a compilação, separação de tarefas entre compilador, sistema operativo e hardware, interligação entre estes componentes, para que serve determinado tipo de hardware, etc.. A ideia é perceber bem como os computadores funcionam e, muito em particular, saber o que fazer quando se suspeita que algo não está a correr bem ou pode ser melhorado. É uma ideia compatível com a definição de "engenheiro": alguém que resolve o problema que lhe é colocado. Não precisa de mostrar que sabe muito, basta que resolva o que lhe é pedido, se possível com uma boa relação custo/benefício.

Lição nº 1: *A essência do saber, tendo-o, consiste em aplicá-lo [Confúcio]*

SO é uma disciplina de "mãos na massa" onde se valoriza a capacidade de aplicação de conhecimentos. Durante as aulas é repetida com frequência a ideia de que apesar das livrarias estarem cheias de livros de informática, as empresas continuam a contratar pessoas capazes de fazerem o diagnóstico correcto e sugerirem a terapia adequada. Compare com uma ida ao médico: se me doer a cabeça, quero que o médico me resolva esse problema e não que me dê remédios para o estômago só porque ele gosta ou estudou mais essa parte da matéria...

Fica aqui então a primeira sugestão para ter sucesso no exame de SO: não adianta "despejar o saco" da teoria, numa tentativa desesperada de que debitando tudo o que se aprendeu (decorou?) sobre SO eventualmente estar-se-á a responder ao que quer que o professor tenha perguntado... Além de fazer perder tempo (a ambos, aluno e professor), este tipo de resposta terá sempre cotação inferior a 50% da pergunta pois está longe de resolver o problema concreto que foi colocado. E quanto mais se escreve maior a probabilidade de dizer asneiras, razão pela qual o despejar do saco raramente ultrapassa os 20 a 30% da cotação. Pelo contrário, uma tentativa "honesta" de responder à pergunta vale perto de 50%, mesmo que a solução proposta esteja errada! Assim, a atitude a tomar perante o exame deverá ser começar por perceber a questão, identificar o que se pretende e de seguida responder de forma sucinta e objectiva apenas a essa questão.

Lição nº 2: *"It´s not what you do, it´s the way that you do it"*

Dentro de um espírito de valorização da contribuição individual e desvalorização das respostas decoradas dos livros sem se perceber o que se passa, as perguntas dos exames de SO contém quase sempre alguma indefinição ou mesmo ambiguidade. Para além de ajudar a dissuadir a tentativa de cópia pelo parceiro do lado, este mecanismo visa forçar cada um a dar a sua própria visão da questão. Quem assiste às aulas teóricas sabe que, por norma, a resposta correcta a uma pergunta típica de SO é "depende", e que é na identificação desse contexto e na justificação das opções é que reside a verdadeira cultura dos sistemas operativos. Isto quer dizer que duas respostas completamente diferentes podem ambas ter cotação muito elevada, desde que convenientemente justificadas.

Lição nº 3: *Decida-se...*

Perante uma dada questão, é frequente surgirem respostas do estilo "podia fazer assim, blá blá, podia fazer assado, blá blá, e também se podia fazer desta maneira, etc." Embora se perceba que tal comportamento resulta normalmente de uma tentativa de mostrar muitos conhecimentos, a menos que isso seja explicitamente solicitado deve procurar apresentar apenas uma solução, a que lhe parecer mais adequada/viável face aos compromissos em jogo. Se lhe pedirem uma e mencionar mais está inconscientemente a dizer que não sente confiança ou não tem conhecimentos práticos suficientes para escolher a melhor opção. Estará portanto a deitar fora (note bem, a deitar fora) pelo menos 30% da cotação da pergunta.

Recorde novamente a metáfora da ida ao médico. Que confiança teria se ele/ela se voltasse para si e lhe dissesse que podia fazer o tratamento de três maneiras, não desse nenhuma pista acerca da melhor opção, e lhe pedisse a si que escolhesse?

Lição nº 4: *Não responda com o enunciado*

É também frequente um estilo de resposta tipo "pescadinha de rabo na boca", em que ao longo de um ou mais parágrafos muito vagos e cheios de frases feitas surge finalmente uma justificação construída à custa do próprio enunciado! Por exemplo, se numa pergunta se disser que é possível tirar partido da dispersão de referências carregando para memória central apenas os blocos que são necessários num determinado momento, está-se a fazer uma afirmação e não uma pergunta. Essa virá a seguir, mas qualquer que seja essa pergunta ela não pode ser respondida dizendo que... a dispersão de referências é benéfica pois permite carregar para memória central apenas os blocos que são necessários. Percebeu a ideia? Não vale a pena perder tempo a evitar responder às perguntas.

Lição nº 5*: "Don´t panic!"*

Quando tudo parecer perdido, procure colocar-se na posição inversa e pensar no tipo de resposta que o professor gostaria de ver para lhe dar a cotação máxima. Não tente deitar-lhe areia para os olhos, ajude-o apenas a dar-lhe boa nota. Pode crer que já houve vários casos de 20 valores a Sistemas Operativos, e que esses foram dias de festa no GSD!

Casos de Estudo:

Para ajudar a compreender a atitude desejável perante um exame de SO, apresentam-se de seguida algumas questões reais, ou seja, perguntas que saíram de facto em exames (concretamente na 1ª e 2ª chamadas dos exames de SOI do 1º semestre de 2003/2004). Embora se discuta o tipo de resposta a dar a cada uma das questões, chama-se a atenção para o facto disso dever ser encarado apenas como um treino, e não como a resposta oficial para as perguntas surgidas no exame[1]. Não olhe para as respostas seguintes como uma cábula, concentre-se apenas na maneira de "atacar" as perguntas. Lembre-se sobretudo que em SO o que conta é "the way that you do it..."

Pergunta:
Compare a segmentação (pura) e a paginação do ponto de vista da alocação de espaço em memória central. Dê uma ideia aproximada dos algoritmos e estruturas de dados usados em cada caso. Que conclusões tira quanto à eficiência?

Comecemos por "traduzir" a pergunta, identificando o que está em causa. Trata-se da gestão de memória, mas qual delas? Será memória virtual, visto que até fala em segmentação e paginação? Será memória real? Mas então os apontamentos não dizem já que a "paginação procura resolver questões de eficiência e a segmentação questões de conveniência"? Algoritmos e estruturas de dados? Oh diabo, não é desta que vou fazer SO...

Analisemos então a pergunta. Em primeiro lugar, o que está em jogo é a **alocação de espaço** em memória central, a velha questão de saber onde colocar os segmentos ou páginas em RAM. Pede-se para fazer uma comparação entre segmentação e paginação **exclusivamente** em termos da "arrumação" dos blocos em memória central. Não são os objectivos da segmentação ou paginação, não é conveniência, não é protecção, nem sequer é partilha nem atribuição de endereços lógicos na altura da compilação ou após passagem pelo linker. Não é uma oportunidade de despejar o saco, está-se apenas a tentar saber se a paginação é mais ou menos eficiente do que a segmentação, olhando apenas para o problema da alocação de espaço. É-lhe pedido que diga se é mais fácil ou mais difícil arrumar em RAM blocos de dimensão variável (segmentos) ou blocos de dimensão fixa (páginas).

Em segundo lugar, é-lhe pedido que apresente provas das afirmações que faz, para ver se batem certo com o que leu nos livros: que a paginação é de facto mais eficiente. Neste caso, terá de fazer a prova olhando para a complexidade dos algoritmos e estruturas de dados necessários à gestão de espaço em RAM, e deduzir daí os respectivos overheads em tempo e espaço: tempo de CPU que o sistema operativo desperdiça (investe?) até decidir onde vai carregar um segmento ou página, e também quanto espaço desperdiça (investe?) nas estruturas de dados, espaço esse que poderia ser mais bem usado dando-o às aplicações. Se o algoritmo é complexo, é capaz de implicar muito tempo de CPU perdido na sua execução, tempo esse que não está a ser usado pelas aplicações.

Vejamos então, como é feita a alocação de blocos de dimensão variável? Que estruturas de dados são precisas? Ora bem, como é que se implementa o first-fit? E o best-fit? Buddy? Hum...uma lista ligada de pares contendo a localização+dimensão de blocos livres? Mas ordenada por localização ou por dimensão? Eu disse ordenada, quanto custa, quanto demora uma ordenação? E se tiver de compactar a memória? Será que dava jeito ter a lista duplamente ligada? Isto está a ficar complicado...

E como é feita a alocação de blocos de dimensão fixa? Como são todos iguais, para saber onde há espaços livres basta um bit para cada um, é indiferente se é no início ou no fim da memória

[1] Se bem que, caso respondesse da forma aqui descrita, seguramente teria nota positiva...

central pois as tabelas de páginas permitem o endereçamento directo em qualquer lugar. Portanto, com a paginação basta um bit map, de dimensão igual ao número de blocos disponíveis ((RAM_SIZE – KERNEL_SIZE) / PAGE_SIZE), e para encontrar um bloco livre basta pesquisar até encontrar um bit a 1 (neste caso 1 significa livre). Com CPUs de 32 bits ao ler uma palavra inteira "trazemos" logo a informação relativa a 32 blocos, e essa palavra for igual a Zero então não há espaço livre aqui e passamos de imediato para os 32 blocos seguintes. Muito mais fácil e rápido do que na segmentação, não acha? E não há necessidade de compactação... Fica portanto demonstrado que, em relação à alocação de espaço em memória central, a paginação é mais eficiente do que a segmentação (embora o bit map possa ocupar mais espaço do que as listas ligadas da segmentação).

Pergunta:
Normalmente um computador tem de executar simultaneamente várias actividades: tratamento de interrupções, actividades do sistema operativo, e aplicações. As aplicações podem ainda ser ou não "multi-threaded". Proponha uma estratégia de escalonamento capaz de lidar com esses quatro tipos de actividades, assegurando-se que justifica as suas opções.

Antes de mais, preste atenção à pergunta. Nota algo de estranho? Estamos a falar de escalonamento, de atribuição de tempo de CPU às diversas actividades, portanto é o mesmo que perguntar qual a maneira apropriada de atribuir tempo de CPU às interrupções, a processos do SO e a processos dos utilizadores. Já nota algo de estranho?

Coloquemos a questão de outra forma. Para se atribui tempo de CPU através do scheduler, ou seja, para se fazer escalonamento de processos, em primeiro lugar é preciso que existam processos prontos a executar... Processos do SO existem, e aplicações também, mas... as interrupções?

Pois é, as interrupções estão aqui a mais, a este nível as interrupções têm mais a ver com hardware e device drivers do que com escalonamento. Não há um processo de tratamento de interrupções, processo esse que seja candidato a CPU. Sempre que o hardware pretende interromper, ou as interrupções estão inibidas (porque será?) ou então elas surgem independentemente da política de escalonamento. São tratadas "interrompendo" momentaneamente a execução do processo que nesse momento estiver a executar, e regressando ao esse processo logo que possível. Se necessário, a rotina de tratamento de interrupções poderá assinalar a ocorrência de um evento, existirá porventura um processo bloqueado à espera desse evento, a seu tempo esse processo será escalonado e completará então as acções dependentes da interrupção. Mas o tratamento inicial da interrupção nada tem a ver com políticas de escalonamento. A inclusão das interrupções nesta pergunta destina-se a avaliar se o aluno percebe de facto o que está a dizer, ou se está apenas a responder por "pattern matching" de palavras da pergunta com o que leu nos livros e apontamentos.

Se tirarmos as interrupções da pergunta, dando obviamente uma justificação semelhante à dada acima, então o que fica? Fica quase a pergunta do costume. Há ainda o detalhe dos processos multi-threaded, há que decidir se um processo que tem vários "fios de execução" cada um a competir por CPU tem ou não direito a mais tempo de CPU do que um processo single-threaded. E a resposta é... depende.

Se for um utilizador normal que crie vários threads numa tentativa de dar a volta à estratégia de escalonamento (por exemplo round-robin) e beneficiar de mais tempo de CPU, se calhar um thread ao esgotar o quantum deve levar a que todo o processo liberte o CPU e dê a vez a outro processo de outro utilizador. Mas ao esgotar o quantum também podia passar para outro thread do mesmo utilizador. Depende, imagine que o processo multi-threaded era um servidor web, e que os threads surgem apenas porque uma questão de eficiência e não por oportunismo de um utilizador. Depende...