Lista de Exercícios 1 - Sistemas Operacionais 1
Gabarito
Professor: Leandro Marzulo
2012/1

1. Nos primeiros computadores, cada byte em um dispositivo de E/S era lido ou escrito diretamente pelo processador, isto é, não havia nenhum mecanismo para realizar a transferência de bytes diretamente entre o dispositivo e a memória. Que implicações esse arranjo tem para a multiprogramação?

   Resp.: Quando não existir uma DMA no hardware, o processador será o responsável pelas transferências de dados entre os dispositivos físicos e a memória principal do computador. Com isso, o processador tenderá a estar ocupado na maior parte do tempo, mesmo que os processos executem operações de E/S com frequência. Isso ocorrerá porque para a maior parte dos dispositivos (com exceção dos mais lentos, como, por exemplo, um modem ligado a uma linha telefônica) o tempo da transferência dos dados entre a memória e o dispositivo dominar ao tempo total da operação de E/S. Logo, o principal ganho da multiprogramação, que é o de evitar que o processador fique ocioso quando operações de E/S são executadas, será reduzido, e a multiprogramação essencialmente permitirá a execução de vários processos no processador.

2. Suponha que um programa A leve 18s para executar no processador e que, para executar a sua tarefa, ele precise fazer E/S por 4s. Se este programa fosse executado em um sistema anterior ao da terceira geração, qual seria a fração de tempo do processador desperdiçada com operações de E/S? Justifique. Este desperdício ainda ocorreria nos sistemas posteriores ao da segunda geração? Justifique.

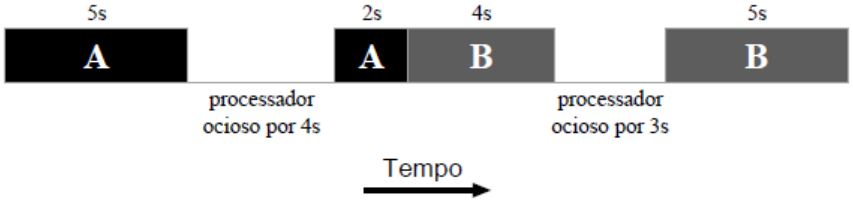
   Resp.: -Pela questão, vemos que o programa executou em um sistema da terceira ou da quarta geração. Logo, o sistema possui o conceito de multiprogramação e, com isso, o tempo de 4s gasto com E/S não está incluído nos 18s do tempo de execução do programa. Note que o programa somente pode ser executado em um sistema da segunda geração, pois na primeira geração o programador manipulava diretamente o hardware do computador, e não existia um processador que executava os programas. Como na segunda geração não existe o conceito de multiprogramação, agora o tempo de 4s de E/S fará parte do tempo de execução do programa no processador. Isso ocorrerá porque o processador, que está executando este programa, ficará ocioso esperando pelo término da operação de E/S, para depois continuar a executar o programa. Logo, o tempo de execução do programa será agora de 22s. Como o processador ficará ocioso por 4s destes 22s, então a fração de tempo desperdiçada do processador será de $4/22 = 2/11 \approx 0.18$, ou seja, aproximadamente 18% do tempo de execução do programa.
   -Não, pois devido ao conceito de multiprogramação que surgiu na terceira geração, o processador deixa de ficar ocioso quando o programa em execução faz operações de E/S.

3. Suponha que dois programas, A e B, estejam para serem executados no processador. O programa A executa por 6s, sendo que 20% deste tempo é gasto esperando pelo término de uma operação de E/S. Já o programa B, que não faz operações de E/S, executa por 2s no processador. Se o sistema operacional não implementa o conceito de multiprogramação, o processador poderá ficar ocioso? Em caso afirmativo, qual será o tempo de ociosidade do processador? Justifique a sua resposta.

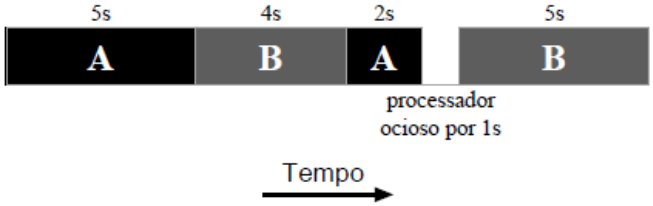
Resp.: -Sim, pois sem a multiprogramação o processador ficará ocioso quando o programa atualmente em execução fizer uma operação de E/S, até que esta operação termine e o programa possa continuar a sua execução.
-Pelo enunciado da questão, o programa A executa por 6s e 20% deste tempo é usado pela operação de E/S. Logo, o processador ficará ocioso por 0, 2 × 6 = 1, 2s. Note que o programa B não afeta o tempo de ociosidade do processador, pois o sistema não usa a multiprogramação.

4. Suponha que somente dois programas, A e B, estejam em execução no processador do computador. O programa A foi o primeiro a executar no processador: executou por 7s, tendo precisado fazer uma operação de E/S, com duração de 4s, após os primeiros 5s de execução. O programa B, que executou por 9s, também precisou fazer uma operação de E/S, com duração de 3s, após os primeiros 4s de execução. Se o sistema operacional não usar a multiprogramação, qual será o tempo de ociosidade do processador? Agora, se o sistema usar a multiprogramação, o processador ficará ocioso? Justifique a sua resposta.

   Resp.: -Se o sistema operacional não usar a multiprogramação, então não poderemos executar um outro programa quando o programa em execução fizer uma operação de E/S e, com isso, o processador ficará ocioso durante a execução desta operação de E/S. Logo, a ordem das execuções dos programas A e B no processador será dada pela figura a seguir. Como o processador ficará ocioso durante a execução de cada operação de E/S, então o tempo de ociosidade do processador, neste caso, será a soma dos tempos das operações de E/S executadas pelos programas A e B, isto é, o tempo será de 4s + 3s = 7s.




   -Agora, quando o sistema operacional usar a multiprogramação, o tempo durante o qual o processador ficará ocioso poderá ser usado para executar outros programas. Neste caso, poderemos usar o tempo de execução da operação de E/S do programa A para executar o programa B, e o tempo da operação de E/S do programa B para executar o programa A. Logo, a nova ordem de execução dos programas A e B no processador será a dada pela figura a seguir. Isto ocorrerá porque o tempo da operação de E/S feita pelo programa A é exatamente o tempo que o programa B executa antes de fazer a sua operação de E/S, e o tempo de execução da operação de E/S feita pelo programa B é suficiente para que o programa A termine a sua execução. Pela figura, vemos que o processador ainda ficará ocioso por 1s, mas isso somente ocorrerá porque não existem outros programas para serem executados no processador.

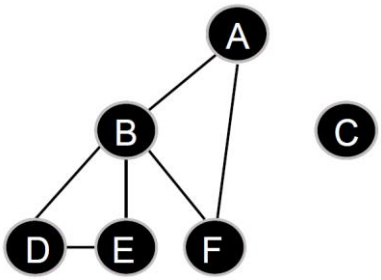



5. Suponha que um computador pode executar 1 bilhão de instruções por segundo, e que uma chamada ao sistema toma 1000 instruções, incluindo a de TRAP e todas as necessárias à troca de contexto. Quantas chamadas ao sistema o computador pode

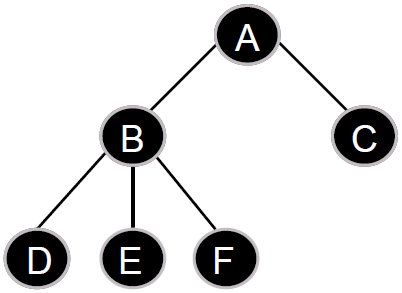
executar por segundo para ainda possuir metade da capacidade do processador para executar códigos de aplicação?

Resp.: Como metade da capacidade do processador deve estar disponível para executar código de aplicação, então poderemos usar até 500 milhões de instruções por segundo para executar chamadas ao sistema. Como cada chamada ao sistema toma 1000 instruções, então poderemos executar até 500 mil chamadas ao sistema por segundo.

6. Um aluno de sistemas operacionais alegou que a hierarquia dada a seguir relaciona os processos A, B, C, D, E e F em execução no sistema operacional. A alegação do aluno está correta? Justifique a sua resposta.




Resp.: Como vimos em aula, os processos no primeiro nível da hierarquia não possuem um pai. Além disso, para cada processo da hierarquia em qualquer outro nível diferente do primeiro, deve existir um processo, no nível imediatamente anterior, que seja o seu pai. Logo a alegação do aluno está incorreta, pois existem três erros na sua hierarquia: os dois ciclos da hierarquia e o processo C, que está no segundo nível da hierarquia e não tem um pai. O processo C ou deveria ser filho do processo A (se ele continuasse no segundo nível), ou deveria estar no primeiro nível. Em relação ao ciclo que envolve os processos A, B e F, o processo A não poderia ser pai de F, pois A está no primeiro nível e F está no terceiro nível. Finalmente, para o ciclo envolvendo os processos B, D e E, D não poderia ser pai de E (ou E não poderia ser pai de D), pois ambos estão no terceiro nível. Na figura a seguir mostramos uma possível hierarquia relacionando os processos A, B, C, D, E e F.




7. Suponha que um sistema operacional esteja executando sobre uma máquina virtual. Suponha ainda que o processador virtual possua um poder de processamento 25% menor do que o do processador real, e que o tempo de execução de uma chamada ao sistema aumente de 10ms para 12ms. Se um determinado programa, tendo feito 25 chamadas ao sistema, executou em 25s, qual seria o tempo de execução diretamente sobre o hardware do computador?

Resp.: Do tempo de 25s, ou seja, 25000ms, de execução do programa, 25×12 = 300ms foram gastos com as 25 chamadas ao sistema operacional feitas pelo programa. Logo, o programa executou no processador durante 25000-300 = 24700ms. Agora, como o uso da máquina virtual reduziu o poder de processamento em 25% então, quando o programa executar sobre o processador real, o seu tempo de execução será reduzido em

25%. Com isso, o programa executará agora no processador por 24700-0, 25×24700 = 24700-6175 = 18525ms. Além disso, como o tempo de execução de uma chamada ao sistema agora será de 10ms, então o programa gastará 25×10 = 250ms para executar as 25 chamadas ao sistema. Logo, o tempo de execução do programa, se ele fosse executado diretamente no processador, seria de 18525+250 = 18775ms, ou seja, 18, 775s.

8. Considerando o programa abaixo, responda:

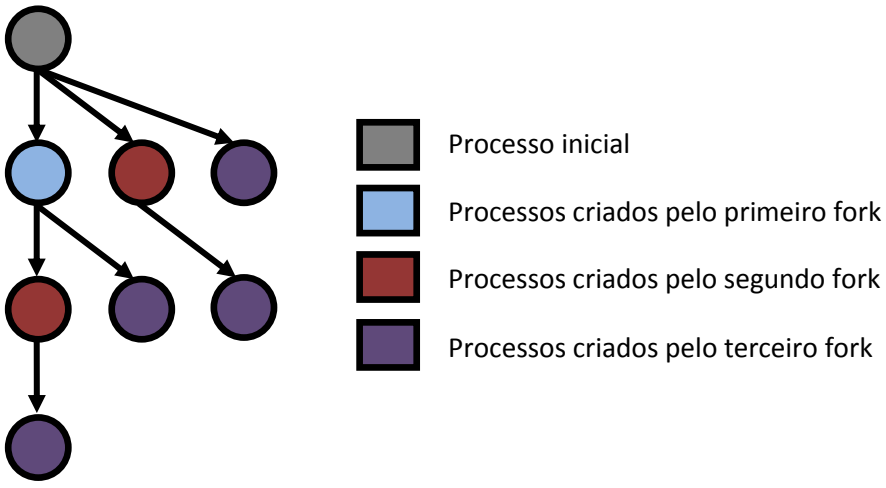
```
#include <stdio.h>
#include <stdlib.h>

int main() {
      fork();
      fork();
      fork();
      return 0;
}
```

a) Ao executar o programa, quantos processos são criados, incluindo o processo inicial, assumindo que não houve falha na execução de nenhuma chamada fork()?

Resp.: a cada chamada fork, cada processo existente cria um processo, ou seja, o número de processos dobra a cada chama fork. Temos 3 chamadas e, portanto, ao final, contando com o processo inicial, temos $2^3$=8 processos.

b) Como fica a sub-árvore de processos do processo incial, com todos os filhos criados (considerando que nenhum deles terminou e assumindo que não houve falha na execução de nenhuma chamada fork())?



9. Considerando o programa abaixo, responda:

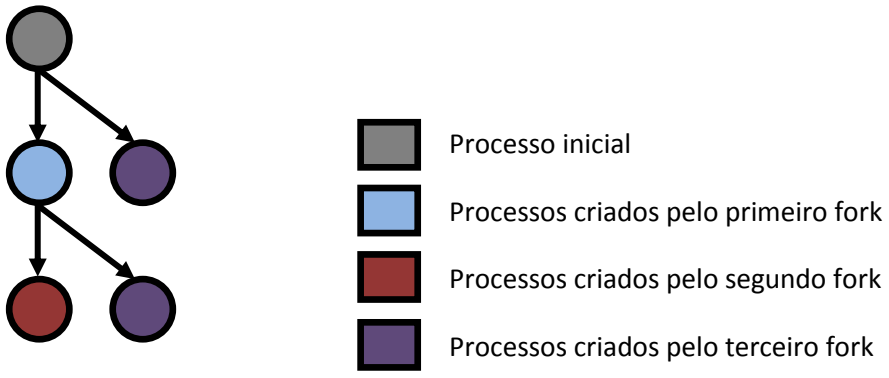
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

int main() {
      pid_t pid;
      pid = fork();
      if (pid == 0) pid = fork();
      if (pid > 0) fork();
      fork();
      return 0;
}
```

c) Ao executar o programa, quantos processos são criados, incluindo o processo inicial, assumindo que não houve falha na execução de nenhuma chamada fork()?

Resp.: Neste caso, temos a execução condicional do segundo e terceiro fork. O primeiro fork será executado e 1 processo será criado. O segundo fork será executado apenas pelo filho criado no fork anterior, resultando na criação de mais 1 processo. Já o terceiro fork será executado por todos os processos com variável pid>0, no caso, o processo inicial (pai no primeiro fork) e pelo pai no segundo fork, resultando na criação de mais 2 processos. Sendo assim, temos o processo inicial e mais 4 processos criados, totalizando 5 processos.

d) Como fica a sub-árvore de processos do processo incial, com todos os filhos criados (considerando que nenhum deles terminou e assumindo que não houve falha na execução de nenhuma chamada fork())?

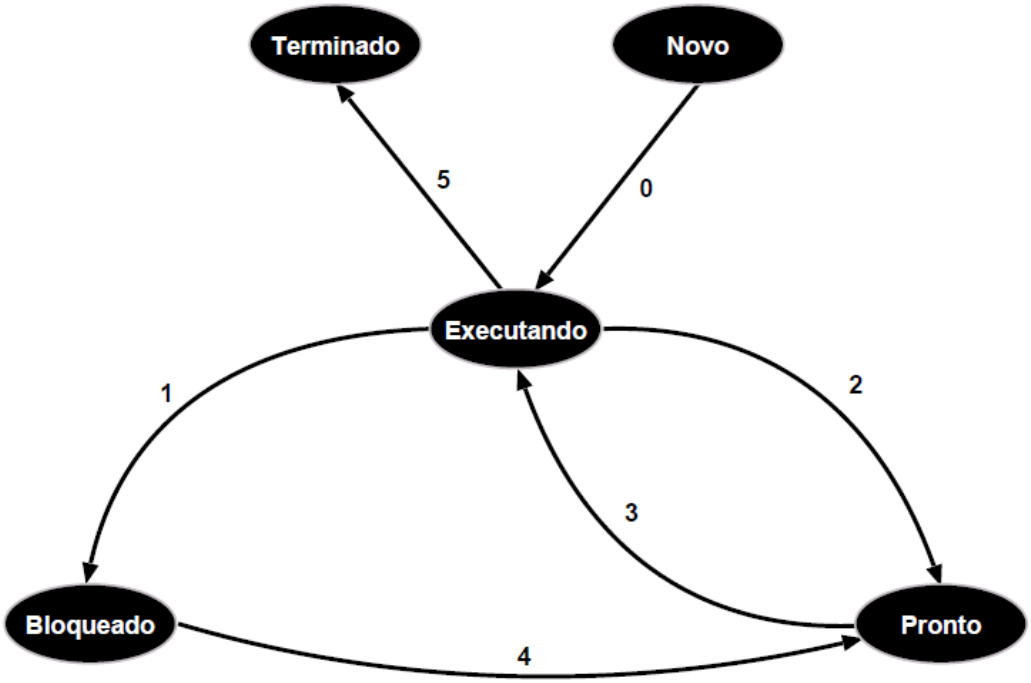



10. Suponha que um processo execute em 5s e que durante a sua execução sejam realizadas 500 operações de E/S. Suponha ainda que o sistema operacional esteja executando sobre o hardware real, e que uma operação de E/S execute em 2ms. Se o sistema operacional agora executar sobre uma máquina virtual que reduz a velocidade das operações de E/S em 25%, e cuja velocidade do processador virtual é 80% da velocidade do processador real, qual será o novo tempo de execução do processo?

Resp.: Pelo enunciado, o processo executa por 5s, ou seja, 5000ms. Como durante a execução do processo são executadas 500 operações de E/S, e como cada operação de

E/S demora 2ms para ser executada, então 1000ms deste tempo serão gastos pelas operações de E/S. Com isso, o processo executará no processador do hardware real por 4000ms. Quando o processo executar no sistema operacional sobre a máquina virtual, que reduz a velocidade das operações de E/S em 25%, cada operação de E/S gastará agora 2 + 0, 25 × 2 = 2, 5ms e, com isso, as 500 operações de E/S executarão agora em 2, 5×500 = 1250ms. Além disso, como o processador virtual possui 80% da velocidade do processador real, então o programa executará no processador virtual por 4000+0, 25×4000 = 5000ms (note que o tempo de execução será 25% maior do que no processador real). Logo, o tempo total de execução do processo no sistema operacional sobre a máquina virtual será de 5000 + 1250 = 6250ms, ou seja, 6,25s.

11. Na figura dada a seguir mostramos uma versão estendida do diagrama de transição clássico dos estados de um processo, com dois novos estados: o estado Novo, em que o processo é colocado quando é criado, e o estado Terminado, em que o processo é colocado quando termina a sua execução. Esse diagrama está correto? Justifique a sua resposta.




**Transições**

0: O novo processo inicia a sua execução.
1: O escalonador escolhe um outro processo para executar.
2: O processo bloqueia esperando por algum evento.
3: O processo é desbloqueado pois o evento já ocorreu.
4: O processo volta a executar no processador.
5: O processo termina a sua execução.

Resp.: Não, porque existem erros na figura. A transição 0 deveria ser do estado Novo para o estado Pronto, pois o processo recém-criado pode ser menos prioritário do que o processo em execução ou do que algum outro processo no estado Pronto. Note que se o novo processo for o mais prioritário e, adicionalmente, precisar ser executado imediatamente, bastará o sistema operacional chamar o escalonador para colocá-lo em execução. Logo, a descrição da transição 0 está incorreta, devendo ser "O novo processo torna-se pronto". Além disso, com base no diagrama que vimos na Aula 4, vemos que as descrições das transições 1 e 2 foram trocadas entre si, o mesmo ocorrendo com as descrições das transições 3 e 4.

12. Como seria utilizar um computador sem um sistema operacional? Quais são suas duas principais funções?

    Resp.: Sem o sistema operacional, um usuário para interagir com o computador deveria conhecer profundamente diversos detalhes sobre hardware do equipamento, o que tornaria seu trabalho lento e com grandes possibilidades de erros. As duas principais funções são "facilidade de acesso aos recursos do sistema" e "compartilhamento de recursos de forma organizada e protegida".

13. Explique o conceito de máquina virtual. Qual a grande vantagem em utilizar este conceito?

    Resp.: O computador pode ser visualizado como uma máquina de camadas, onde inicialmente existem duas camadas: hardware (nível 0) e sistema operacional (nível 1). Desta forma, o usuário pode enxergar a máquina como sendo apenas o sistema operacional, ou seja, como se o hardware não existisse. Esta visão modular e abstrata é chamada máquina virtual. A vantagem desse conceito é tornar a interação entre usuário e computador mais simples, confiável e eficiente.

14. Por que dizemos que existe uma subutilização de recursos em sistemas monoprogamáveis? Qual a grande diferença entre sistemas monoprogramáveis e sistemas multiprogramáveis?

    Resp.: Porque em sistemas monoprogramáveis somente é possível a execução de um programa por vez. Como um programa não utiliza todos os recursos do sistema totalmente ao longo da sua execução, existe ociosidade e, consequentemente, subutilização de alguns recursos.

15. Um sistema monousuário pode ser um sistema multiprogramável? Dê um exemplo.

    Resp.: Sim, somente um usuário interage com o sistema podento possuir diversas aplicações executando concorrentemente. O sistema Windows NT é um exemplo.

16. Como funcionam os sistemas de tempo compartilhado? Quais as vantagens em utilizá-los?

    Resp.: Os sistemas de tempo compartilhado (time-sharing) permitem que diversos programas sejam executados a partir da divisão do tempo do processador em pequenos intervalos, denomidados fatia de tempo (time-slice). A vantagem na sua utilização é possibilitar para cada usuário um ambiente de trabalho próprio, dando a impressão de que todo o sistema está dedicado, exclusivamente, a ele.

17. Qual a grande diferença entre sistemas de tempo compartilhado e tempo real? Quais aplicações são indicadas para sistemas de tempo real?

    Resp.: O fator tempo de resposta. Nos sistemas de tempo real, os tempos de resposta devem estar dentro de limites rígidos. Aplicações de controle de processos, como no monitoramento de refinarias de petróleo, controle de tráfego aéreo, de usinas termoelétricas e nucleares são executadas em sistemas de tempo real.

18. O que é concorrência e como este conceito está presente nos sistemas operacionais multiprogramáveis?

Resp.: Concorrência é o princípio básico para projeto e implementação dos sistemas operacionais multiprogramáveis onde é possível o processador executar instruções em paralelo com operações de E/S. Isso possibilita a utilização concorrente da UCP por diversos programas sendo implementada de maneira que, quando um programa perde o uso do processador e depois retorna para continuar o processamento, seu estado deve ser idêntico ao do momento em que foi interrompido. O programa deverá continuar sua execução exatamente na instrução seguinte àquela em que havia parado, aparentando ao usuário que nada aconteceu.

19. Por que o mecanismo de interrupção é fundamental para a implementação da multiprogramação?

    Resp.: Porque é em função desse mecanismo que o sistema operacional sincroniza a execução de todas as suas rotinas e dos programas dos usuários, além de controlar dispositivos.

20. Explique o mecanismo de funcionamento das interrupções.

    Resp.: Uma interrupção é sempre gerada por algum evento externo ao programa e, neste caso, independe da instrução que está sendo executada. Ao final da execução de cada instrução, a unidade de controle verifica a ocorrência de algum tipo de interrupção. Neste caso, o programa em execução é interrompido e o controle desviado para uma rotina responsável por tratar o evento ocorrido, denominada rotina de tratamento de interrupção. Para que o programa possa posteriormente voltar a ser executado, é necessário que, no momento da interrupção, um conjunto de informações sobre a sua execução seja preservado. Essas informações consistem no conteúdo de registradores, que deverão ser restaurados para a continuação do programa.

21. O que são eventos síncronos e assíncronos? Como estes eventos estão relacionados ao mecanismo de interrupção e exceção?

    Resp.: Evento síncronos são resultados direto da execução do programa corrente. Tais eventos são previsíveis e, por definição, só podem ocorrer um único de cada vez. Eventos assíncronos não são relacionados à instrução do programa corrente. Esses eventos, por serem imprevisíveis, podem ocorrer múltiplas vezes, como no caso de diversos dispositivos de E/S informarem ao processador que estão prontos para receber ou transmitir dados. Uma interrupção é um evento assíncrono enquanto uma exceção é um evento síncrono.

22. Qual a vantagem da E/S controlada por interrupção comparada com a técnica de polling (ou programada)?

    Resp.: Na E/S controlada por interrupção, as operações de E/S podem ser realizadas de uma forma mais eficiente. Em vez de o sistema periodicamente verificar o estado de uma operação pendente como na técnica de polling, o próprio controlador interrompe o processador para avisar do término da operação. Com esse mecanismo, o processador, após a execução de um comando de leitura ou gravação, permanece livre para o processamento de outras tarefas.

23. O que é DMA e qual a vantagem desta técnica?

    Resp.: A técnica de DMA permite que um bloco de dados seja transferido entre a memória principal e dispositivos de E/S, sem a intervenção do processador, exceto no início e no final da transferência. Quando o sistema deseja ler ou gravar um bloco de dados, o processador informa ao controlador sua localização, o dispositivo de E/S, a

posição inicial da memória de onde os dados serão lidos ou gravados e o tamanho do bloco. Com estas informações, o controlador realiza a transferência entre o periférico e a memória principal, e o processador é somente interrompido no final da operação.

24. Como a técnica de buffering permite aumentar a concorrência em um sistema computacional?

    Resp.: Como o buffering permite minimizar o problema da disparidade da velocidade de processamento existente entre o processador e os dispositivos de E/S, esta técnica permite manter, na maior parte do tempo, processador e dispositivos de E/S ocupados.

25. Explique o mecanismo de spooling de impressão.

    Resp.: No momento em que um comando de impressão é executado, as informações que serão impressas são gravadas antes em um arquivo em disco, conhecido como arquivo de spool, liberando imediatamente o programa para outras atividades. Posteriormente, o sistema operacional encarrega-se em direcionar o conteúdo do arquivo de spool para a impressora.

26. Em um sistema multiprogramável, seus usuários utilizam o mesmo editor de textos (200 Kb), compilador (300 Kb), software de correio eletrônico (200 Kb) e uma aplicação corporativa (500 Kb). Caso o sistema não implemente reentrância, qual o espaço de memória principal ocupado pelos programas quando 10 usuários estiverem utilizando todas as aplicações simultaneamente? Qual o espaço liberado quando o sistema implementa reentrância em todas as aplicações?

    Resp.: Sem reentrância, cada usuário teria sua cópia do código na memória totalizando 10 x (200 Kb + 300 Kb + 200 Kb + 500 Kb) = 12.000 Kb. Caso a reentrância seja implementada, apenas uma cópia do código seria necessária na memória principal (200 Kb + 300 Kb + 200 Kb + 500 Kb) totalizando 1.200 Kb. Um total de 10.800 Kb seriam liberados da memória principal.

27. Por que a questão da proteção torna-se fundamental em ambientes multiprogramáveis?

    Resp.: Se considerarmos que diversos usuários estão compartilhando os mesmos recursos como memória, processador e dispositivos de E/S, deve existir uma preocupação em garantir a confiabilidade e a integridade dos programas e dados dos usuários, além do próprio sistema operacional.

28. O que é o núcleo do sistema e quais são suas principais funções?

    Resp.: É o conjunto de rotinas que oferece serviços aos usuários, suas aplicações, além do próprio sistema operacional. As principais funções do núcleo encontradas na maioria dos sistemas comerciais são: tratamento de interrupções e exceções; criação e eliminação de processos e threads; sincronização e comunicação entre processos e threads; escalonamento e controle dos processos e threads; gerência de memória; gerência do sistema de arquivos; gerência de dispositivos de E/S; suporte à redes locais e distribuídas; contabilização do uso do sistema; auditoria e segurança do sistema.

29. O que é uma system call e qual sua importância para a segurança do sistema? Como as system calls são utilizadas por um programa?

    Resp.: As system calls podem ser entendidas como uma porta de entrada para o acesso ao núcleo do sistema operacional e a seus serviços. Sempre que um usuário ou

aplicação desejar algum serviço do sistema, é realizada uma chamada a uma de suas rotinas através de uma system call. Através dos parâmetros fornecidos na system call, a solicitação é processada e uma resposta é retornada a aplicação juntamente com um estado de conclusão indicando se houve algum erro. O mecanismo de ativação e comunicação entre o programa e o sistema operacional é semelhante ao mecanismo implementado quando um programa chama uma subrotina.

30. O que são instruções privilegiadas e não privilegiadas? Qual a relação dessas instruções com os modos de acesso?

    Resp.: Instruções privilegiadas são instruções que só devem ser executadas pelo sistema operacional ou sob sua supervisão, impedindo, assim, a ocorrência de problemas de segurança e integridade do sistema. As instruções não-privilegiadas não oferecem risco ao sistema. Quando o processador trabalha no modo usuário, uma aplicação só pode executar instruções não-privilegiadas, tendo acesso a um número reduzido de instruções, enquanto no modo kernel ou supervisor a aplicação pode ter acesso ao conjunto total de instruções do processador.

31. Explique como funciona a mudança de modos de acesso e dê um exemplo de como um programa faz uso desse mecanismo.

    Resp.: Sempre que um programa necessita executar uma instrução privilegiada, a solicitação deve ser realizada através de uma chamada a uma system call, que altera o modo de acesso do processador do modo usuário para o modo kernel. Ao término da execução da rotina do sistema, o modo de acesso retorna para o modo usuário.

32. Compare as arquiteturas monolítica e de camadas. Quais as vantagens e desvantagens de cada arquitetura?

    Resp.: A arquitetura monolítica pode ser comparada com uma aplicação formada por vários módulos que são compilados separadamente e depois linkados, formando um grande e único programa executável, onde os módulos podem interagir livremente. Na arquitetura de camadas, o sistema é dividido em níveis sobrepostos. Cada camada oferece um conjunto de funções que podem ser utilizadas apenas pelas camadas superiores. A vantagem da estruturação em camadas é isolar as funções do sistema operacional, facilitando sua manutenção e depuração, além de criar uma hierarquia de níveis de modos de acesso, protegendo as camadas mais internas. Uma desvantagem para o modelo de camadas é o desempenho. Cada nova camada implica em uma mudança no modo de acesso.

33. Quais as vantagens do modelo de máquina virtual?

    Resp.: Além de permitir a convivência de sistemas operacionais diferentes no mesmo computador, a vantagem desse modelo é criar um isolamento total entre cada VM, oferecendo grande segurança para cada máquina virtual.

34. Como funciona o modelo cliente-servidor na arquitetura microkernel? Quais suas vantagens e desvantagens dessa arquitetura?

    Resp.: Sempre que uma aplicação deseja algum serviço, é realizada uma solicitação ao processo responsável. Neste caso, a aplicação que solicita o serviço é chamada de cliente, enquanto o processo que responde à solicitação é chamado de servidor. Um cliente, que pode ser uma aplicação de um usuário ou um outro componente do sistema operacional, solicita um serviço enviando uma mensagem para o servidor. O servidor responde ao cliente através de uma outra mensagem. A utilização deste modelo permite

que os servidores executem em modo usuário, ou seja, não tenham acesso direto a certos componentes do sistema. Apenas o núcleo do sistema, responsável pela comunicação entre clientes e servidores, executa no modo kernel. Como conseqüência, se ocorrer um erro em um servidor, este poderá parar, mas o sistema não ficará inteiramente comprometido, aumentando assim a sua disponibilidade. Outra vantagem é que a arquitetura microkernel permite isolar as funções do sistema operacional por diversos processos servidores pequenos e dedicados a serviços específicos, tornado o núcleo menor, mais fácil de depurar e, conseqüentemente, aumentando sua confiabilidade. Na arquitetura microkernel, o sistema operacional passa a ser de mais fácil manutenção, flexível e de maior portabilidade. Apesar de todas as vantagens deste modelo, sua implementação, na prática, é muito difícil. Primeiro existe o problema de desempenho, devido a necessidade de mudança de modo de acesso a cada comunicação entre clientes e servidores. Outro problema é que certas funções do sistema operacional exigem acesso direto ao hardware, como operações de E/S.

35. Defina o conceito de processo.

    Resp.: Um processo pode ser definido como o ambiente onde um programa é executado. Este ambiente, além das informações sobre a execução, possui também o quanto de recursos do sistema cada programa pode utilizar, como o espaço de endereçamento, tempo de processador e área em disco.

36. Por que o conceito de processo é tão importante no projeto de sistemas multiprogramáveis?

    Resp.: Através de processos, um programa pode alocar recursos, compartilhar dados, trocar informações e sincronizar sua execução. Nos sistemas multiprogramáveis os processos são executados concorrentemente, compartilhando o uso do processador, memória principal, dispositivos de E/S dentre outros recursos.

37. É possível que um programa execute no contexto de um processo e não execute no contexto de um outro? Por que?

    Resp.: Sim, pois a execução de um programa pode necessitar de recursos do sistema que um processo pode possuir enquanto outro não.

38. Quais partes compõem um processo?

    Resp.: Um processo é formado por três partes, conhecidas como contexto de hardware, contexto de software e espaço de endereçamento, que juntos mantêm todas as informações necessárias à execução de um programa.

39. O que é uma troca de contexto? Como e quando ela ocorre?

    Resp.: O contexto de hardware armazena o conteúdo dos registradores gerais da UCP, além dos registradores de uso específico como program counter (PC), stack pointer (SP) e registrador de status. Quando um processo está em execução, o seu contexto de hardware está armazenado nos registradores do processador. No momento em que o processo perde a utilização da UCP, o sistema salva as informações no contexto de hardware do processo.

40. O que é o espaço de endereçamento de um processo?

    Resp.: O espaço de endereçamento é a área de memória pertencente ao processo onde as instruções e dados do programa são armazenados para execução. Cada processo possui

seu próprio espaço de endereçamento, que deve ser devidamente protegido do acesso dos demais processos

41. Como o sistema operacional implementa o conceito de processo? Qual a estrutura de dados indicada para organizar os diversos processos na memória principal?

    Resp.: O processo é implementado pelo sistema operacional através de uma estrutura de dados chamada bloco de controle do processo (Process Control Block — PCB). A partir do PCB, o sistema operacional mantém todas as informações sobre o contexto de hardware, contexto de software e espaço de endereçamento de cada processo.

42. Desenhe o diagrama clássico (de 3 estágios) de processos. Explique todas as transições. Como este diagrama é estendido para incluir a situação onde um processo não está residente em memória? Desenhe o novo diagrama e explique as novas transições.

43. Explique a diferença entre processos foreground e background.

    Resp.: Um processo foreground é aquele que permite a comunicação direta do usuário com o processo durante o seu processamento. Neste caso, tanto o canal de entrada quanto o de saída estão associados a um terminal com teclado, mouse e monitor, permitindo, assim, a interação com o usuário. Um processo background é aquele onde não existe a comunicação com o usuário durante o seu processamento. Neste caso, os canais de E/S não estão associados a nenhum dispositivo de E/S interativo, mas em geral a arquivos de E/S.

44. O que são processos CPU-bound e I/O-bound?

    Resp.: Processos CPU-bound usam a CPU com mais frequência e por mais tempo, enquanto processos I/O-bound fazem I/O com mais frequência e por mais tempo.

45. Como funciona a chamada de sistema fork() de sistemas do tipo UNIX? Dê um exemplo de uso com um programa em C.

    Resp.: A chama fork() cria um processo que é a cópia do pai, ou seja, seu contexto é igual ao do pai. No processo pai, a chamada fork retorna um número inteiro positivo que corresponde ao PID do processo filho criado, enquanto que no processo filho, a chamada fork retorna zero. Esta é a maneira de saber, dentro do programa, o que fazer para o pai e para o filho depois da chamada. Exemplos de uso se encontram no site.

46. Qual a função da chamada exec() de sistemas do tipo UNIX? Dê um exemplo de uso com um programa em C.

    Resp.: A chamada exec troca a área de texto do processo, ou seja, altera o programa a ser executado. Geralmente é usada em conjunto com a chamada fork quando se deseja chamar a execução de um programa à partir de outro. Exemplos de uso se encontram no site.

47. Qual a função das chamadas shmget(), shmat() e shmdt() de sistemas do tipo UNIX? Dê um exemplo de uso com um programa em C.

    As chamadas shmged, shmat e shmdt, respectivamente, cria/obtém uma região de memória compartilhada, anexa a região ao espaço de endereçamento do processo e desanexa a região do espaço de endereçamento do processo. Exemplos de uso se encontram no site.

48. O que são pipes? Dê um exemplo de uso da chamada pipe() dos sistemas do tipo UNIX com um programa em C.

    Resp.: Pipes permitem que dois processos comuns se comuniquem na forma de produtor consumidor. Exemplos do uso de pipes se encontram no site.