Curso de Tecnologia em Sistemas de Computação
Disciplina de Sistemas Operacionais
Professores: Valmir C. Barbosa e Felipe M. G. França
Assistente: Alexandre H. L. Porto

Quarto Período
AD1 - Segundo Semestre de 2009

Atenção: Tem havido muita discussão sobre a importância de que cada aluno redija suas próprias respostas às questões da AD1. Os professores da disciplina, após refletirem sobre o assunto, decidiram o seguinte: Cada aluno é responsável por redigir suas próprias respostas. Provas iguais umas às outras terão suas notas diminuídas. As diminuições nas notas ocorrerão em proporção à similaridade entre as respostas. Exemplo: Três alunos que respondam identicamente a uma mesma questão terão, cada um, 1/3 dos pontos daquela questão.

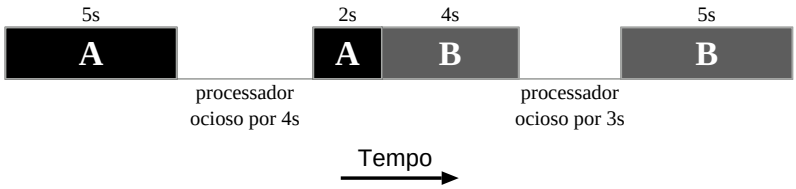
Nome -
Assinatura -

---

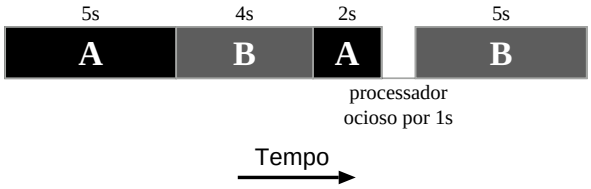
1. (1.5) Suponha que somente dois programas, A e B, estejam em execução no processador do computador. O programa A foi o primeiro a executar no processador: executou por 7s, tendo precisado fazer uma operação de E/S, com duração de 4s, após os primeiros 5s de execução. O programa B, que executou por 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.

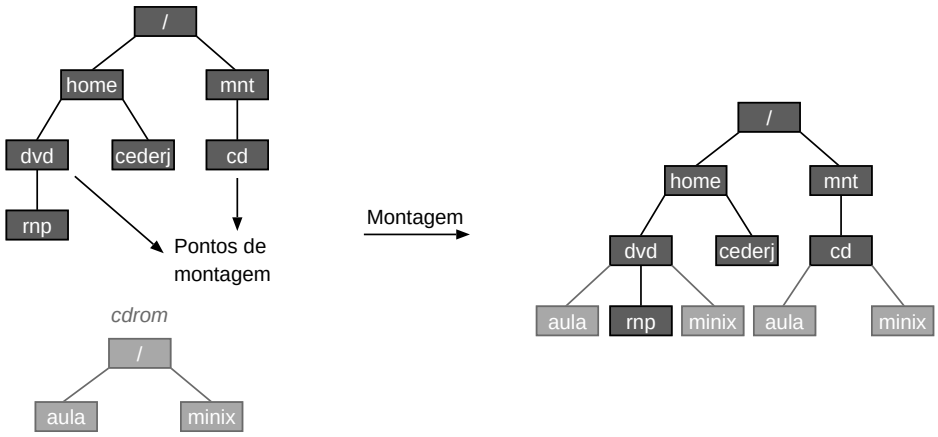
Resp.: -Se o sistema operacional não usar a multiprogramação, então não poderemos executar um outro programa quando o programa em execução fizer uma operação de E/S e, com isso, o processador ficará ocioso durante a execução desta operação de E/S. Logo, a ordem das execuções dos programas A e B no processador 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 $4s + 3s = 7s$.



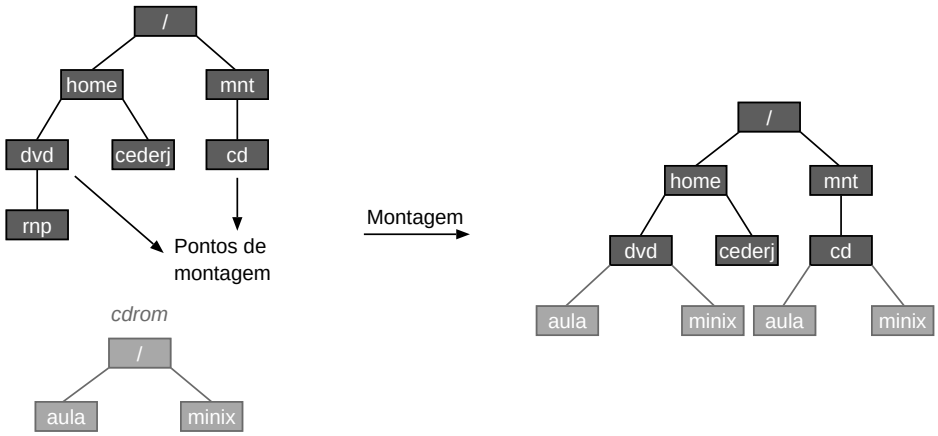
-Agora, quando o sistema operacional usar a multiprogramação, o tempo durante o qual o processador ficará ocioso poderá ser usado para executar outros programas. Neste caso, poderemos usar o tempo de execução da operação de E/S do programa A para executar o programa B, e o tempo da operação de E/S do 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.



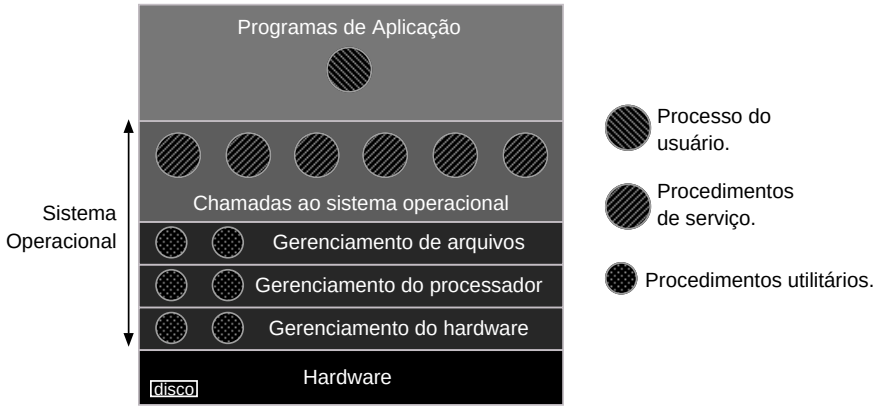
2. (1.5) Um aluno de sistemas operacionais alega que a figura a seguir está correta, pois representa um exemplo de montagem do sistema de arquivos de um *cdrom*. A afirmação do aluno é verdadeira? Justifique a sua resposta.



Resp.: A afirmação do aluno não é verdadeira. Ao montamos um sistema de arquivos em um diretório, todo o conteúdo deste diretório fica inacessível até que este sistema de arquivos seja desmontado. Isto ocorre porque, ao usarmos um diretório como o ponto de montagem, ele é usado temporariamente como o ponto inicial para acessarmos somente o sistema de arquivos que foi montado nele. Note que podemos montar um mesmo sistema de arquivos, como o de um *cdrom*, em diretórios diferentes, isto é, usando pontos de montagem diferentes, caso usemos o comando de montagem para cada um destes diretórios. A figura a seguir corretamente reproduz o sistema de arquivos após as montagens.

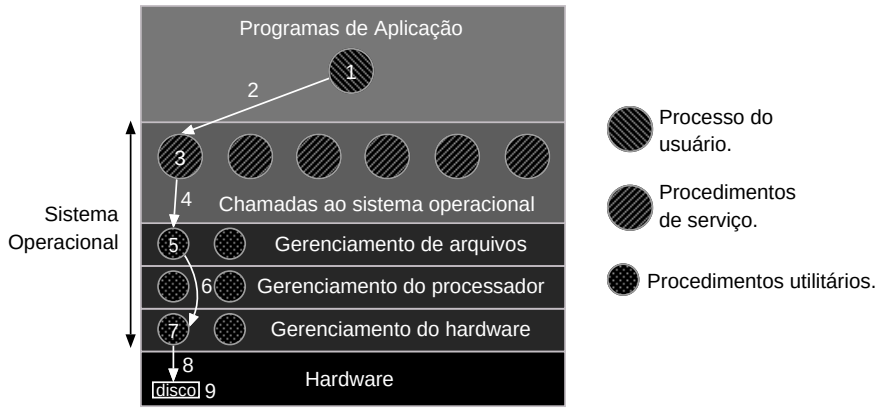


3. (1.5) Na figura a seguir mostramos um sistema operacional estruturado em camadas. Descreva os passos que serão executados quando um processo de usuário acessar um arquivo no disco.



Resp.: Na figura a seguir mostramos os passos que serão executados ao acessarmos um arquivo. No passo 1, o processo fará uma chamada à função da biblioteca responsável por abrir um arquivo. Esta função, por sua vez, executará a chamada ao sistema operacional responsável pela abertura dos arquivos. O passo 2 ocorrerá quando esta função da biblioteca executar a instrução TRAP, pois precisaremos mudar o processador para o modo supervisor ao executar o código do núcleo responsável pelo tratamento das chamadas do sistema operacional. No passo 3, este código do núcleo executará o procedimento responsável

pela chamada que abre um arquivo. Este procedimento precisará acessar o sistema de arquivos com as informações sobre todos os arquivos armazenados no disco, pois para abrir um arquivo precisaremos obter algumas informações sobre ele. Ao acessar o sistema de arquivos, o passo 4 será executado, pois precisaremos chamar um dos procedimentos da camada de gerenciamento de arquivos. Isto ocorrerá porque o sistema de arquivos é uma abstração que permite vermos o disco como um conjunto de arquivos e diretórios. No passo 5, este procedimento precisará acessar o disco para obter as informações do arquivo a ser aberto. Como o disco visto pela camada de gerenciamento de arquivos é um disco abstrato, o passo 6 então ocorrerá ao executar o procedimento da camada de gerenciamento de hardware responsável por fornecer uma versão abstrata do disco. No passo 7, este procedimento será chamado para determinar em que parte do disco real do hardware estarão as informações do arquivo que devem ser lidas do disco abstrato. O passo 8 ocorrerá quando o procedimento responsável pelo disco abstrato acessar o disco real e enviar os comandos à controladora deste disco necessários à leitura das informações do arquivo, o que inicializará uma operação de E/S para ler o disco. Finalmente, no passo 9, os circuitos do disco começarão a procurar as informações do arquivo nas superfícies físicas do disco. Agora, como o processo somente poderá continuar a executar após a leitura das informações do arquivo, este processo deverá ser bloqueado até o término da operação de E/S. Quando a operação de E/S terminar, o procedimento que gerencia o disco abstrato retornará as informações lidas ao procedimento que trata da abertura do arquivo. Este procedimento, por sua vez, repassará as informações necessárias à abertura do arquivo ao procedimento responsável por tratar a chamada que abre um arquivo. Finalmente, o processo será desbloqueado e o controle retornará à função da biblioteca que, por sua vez, abrirá o arquivo e retornará o identificador deste arquivo ao processo que chamou esta função, permitindo finalmente que o processo continue a sua execução.

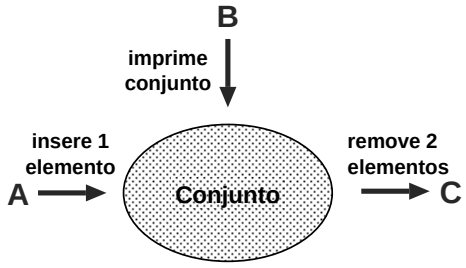


4. (1.5) Suponha que o computador possua somente um processador, e que os processos A, B, C e D sejam os únicos em execução no computador. Suponha ainda que cada processo precise executar no processador por 2 unidades de tempo. Se o escalonador alternar o processador entre estes processos a cada unidade de tempo, de tal modo que um processo somente volte a executar após todos os outros processos terem executado, quantas unidades de tempo decorrerão entre o início e o término da execução de cada processo? Quais serão os novos tempos decorridos se o computador possuir agora 2 processadores? Esses tempos serão os mesmos se aumentarmos o número de processadores para 4?

   Resp.: -Quando o computador tiver somente um processador, note que um dos processos, após executar por uma unidade de tempo, somente executará novamente no processador depois de três unidades de tempo. Isto ocorrerá porque, neste intervalo de tempo, os três outros processos executarão também por uma unidade de tempo no processador (o que sempre ocorrerá, pois todos os processos precisam executar por duas unidades de tempo no processador). Ao executar novamente no processador por mais uma unidade de tempo, o processo terminará a sua execução, pois já terá executado por duas unidades de tempo. Logo, o tempo decorrido entre o início e o término da execução de cada processo (A, B, C e D) será de cinco unidades de tempo, pois todos os processos precisam executar por duas unidades de tempo no processador.
   -Agora, se o computador tiver dois processadores, dois processos poderão executar em paralelo. Logo, após executar por uma unidade de

tempo em um dos processadores, um processo somente poderá executar novamente em um dos processadores após uma unidade de tempo. Isto ocorrerá porque os dois processadores precisarão executar, por uma unidade de tempo, os processos que não foram executados na unidade de tempo anterior (o que também sempre ocorrerá, pois cada processo ainda precisa executar por duas unidades de tempo). Logo, o tempo decorrido entre o início e o término da execução de cada um dos processos A, B, C e D será de três unidades de tempo, novamente porque todos os processos precisam executar por duas unidades de tempo.
-Neste último caso um processo, após executar por uma unidade de tempo em um dos processadores, poderá executar novamente por uma unidade de tempo (no mesmo processador ou em um outro processador). Isto ocorrerá porque, como o número de processadores do computador é igual ao número de processos, sempre existirão processadores disponíveis para executar os três outros processos. Com isso, o tempo decorrido entre o início e o término da execução de cada um dos processos A, B, C e D será o mesmo tempo que cada um deles precisa executar no processador, isto é, duas unidades de tempo.

5. (2.0) O conjunto dado na figura a seguir, que pode armazenar até n elementos, é compartilhado por três processos, A, B e C. O processo A sempre coloca um elemento no conjunto, o processo B sempre imprime o conteúdo atual do conjunto e o processo C sempre remove dois elementos do conjunto. Supondo que a estrutura de dados que representa o conjunto somente armazene os elementos do conjunto, como os semáforos podem ser usados para garantir o correto funcionamento dos processos A, B e C? Justifique a sua resposta.



Resp.: Para garantir o correto funcionamento dos processos A, B e C, isto é, a sua correta sincronização, vamos precisar de três semáforos,

sendo que dois deles, *vazio* e *cheio*, são de contagem, e o terceiro, *acesso*, é binário. O semáforo binário *acesso* é usado para garantir o acesso exclusivo ao conjunto, e é inicializado com o valor 1, pois nenhum processo ainda está executando. O semáforo *vazio* é usado para contar o espaço disponível no conjunto para a inserção de novos elementos, e é usado para bloquear o processo A se o conjunto estiver cheio, isto é, quando o conjunto não possuir espaço disponível para mais elementos. O semáforo *cheio* conta o número de elementos no conjunto, e é usado para bloquear o processo C caso o conjunto esteja vazio, isto é, se não possuir elementos. Os valores iniciais dos semáforos *vazio* e *cheio* dependem do número de elementos no conjunto antes da execução dos processos A e C. Se o número de elementos for m, então temos um espaço disponível de $n - m$ no conjunto e, portanto, o valor inicial do semáforo *cheio* é m e o valor inicial do semáforo *vazio* é $n - m$. O processo A deverá usar a função $InserirElemento(e)$ dada a seguir para inserir o elemento e no conjunto. Já o processo B deverá usar a função $ImprimirConjunto(\textbf{void})$ a seguir para imprimir os elementos do conjunto. Finalmente, o processo C deverá usar a função $RemoverElementos(e_1, e_2)$ dada a seguir para remover dois elementos e_1 e e_2 do conjunto. Como o processo C precisará de dois elementos do conjunto note, no código para a função *RemoverElementos*, que precisaremos executar duas vezes a operação **P** sobre *cheio*, pois removeremos dois elementos. Além disso, precisaremos executar a operação **V** também duas vezes sobre *vazio*, pois duas novas posições estarão disponíveis após a remoção destes elementos. Observe também que a função *ImprimeConjunto* somente precisará usar o semáforo *acesso*, pois somente imprime os elementos do conjunto.

```
void InserirElemento(e)
{
  P(vazio);
  P(acesso);
  // Código para inserir o elemento e no conjunto.
  V(acesso);
  V(cheio);
}
```

```
void ImprimirConjunto(void)
{
  P(acesso);
  // Código para imprimir todos os elementos no conjunto.
  V(acesso);
}

void RemoverElementos(e1, e2)
{
  P(cheio);
  P(cheio);
  P(acesso);
  // Código para remover dois elementos do conjunto e depois
  // colocá-los em e1 e e2.
  V(acesso);
  V(vazio);
  V(vazio);
}
```

6. (2.0) Suponha que dois processos, A e B, estejam acessando uma árvore compartilhada com tamanho ilimitado, e suponha que A e B executem por, respectivamente, 7 e 10 unidades de tempo. O processo A chama, em cada unidade de tempo de sua execução, duas vezes a função *RemoverNo* para remover dois nós da árvore. Esta função não fará nada se a árvore estiver vazia, isto é, A não será bloqueado. O processo B chama, também a cada unidade de tempo da sua execução, três vezes a função *InserirNo* para inserir três nós na árvore. Para cada um dos algoritmos de escalonamento dados a seguir, diga qual será o número de nós da árvore após a execução dos processos A e B, justificando a sua resposta:

   (a) (1.0) O algoritmo por *round robin*, supondo que um quantum é igual a duas unidades de tempo, e que inicialmente o escalonador escolhe o processo A para ser executado.

   Resp.: Na tabela dada a seguir mostramos a execução dos processos A e B no processador de acordo com o algoritmo de *round robin*. Na primeira linha da tabela mostramos, em cada coluna,

o tempo decorrido após a execução do processo desta coluna. Na segunda linha mostramos os processos executados para cada unidade de tempo dada na primeira linha. Finalmente, na terceira linha mostramos, em cada coluna, o número de nós na árvore após a execução do processo desta coluna. Como não foi informado o número de nós da árvore vamos supor, na terceira linha, que o número de nós antes do início da execução dos processos A e B era de n. Pelo enunciado, como o processo A executa por 7 unidades de tempo, 4 nós serão removidos da árvore nas três primeiras execuções de A no processador, e 2 nós serão removidos na última execução de A no processador. Já o processo B, que executa por 10 unidades de tempo, irá inserir 6 nós na árvore em cada uma das suas cinco execuções no processador. Pela tabela, vemos que o número de nós na árvore, após a execução dos processos A e B, será de $n + 16$.

0	2	4	6	8	10	12	13	15
A	B	A	B	A	B	A	B	B
$n-4$	$n+2$	$n-2$	$n+4$	n	$n+6$	$n+4$	$n+10$	$n+16$

(b) (1.0) O algoritmo por prioridades, sendo que a prioridade do processo em execução é reduzida a cada unidade de tempo, que este processo deixa de executar somente quando a sua prioridade não é mais a maior de todas, e que as prioridades dos processos A e B são, respectivamente, de 5 e 8.

Resp.: Nas tabelas dadas a seguir, mostramos a execução dos processos A e B no processador, agora de acordo com o algoritmo por prioridades. Na primeira linha de cada tabela mostramos, para cada coluna, a prioridade do processo desta coluna após a sua execução no processador. Na segunda linha de cada tabela mostramos os processos associados a cada uma das prioridades dadas na primeira linha. Finalmente, na terceira linha de cada tabela mostramos, em cada coluna, o número de nós da árvore após a execução do processo desta coluna. Assim como antes, vamos supor que a árvore tinha n nós antes da execução dos processos A e B. Como a prioridade do processo em execução é reduzida

a cada unidade de tempo, então A removerá dois nós da árvore quando a sua prioridade for reduzida, e B inserirá três nós na árvore quando a sua prioridade for reduzida. Pela última linha da última tabela, vemos que o número de nós na árvore, após a execução dos processos A e B, será também de $n + 16$.

7	6	5	4	4	3
B	B	B	B	A	A
$n+3$	$n+6$	$n+9$	$n+12$	$n+10$	$n+8$

3	2	2	1	1	0
B	B	A	A	B	B
$n+11$	$n+14$	$n+12$	$n+10$	$n+13$	$n+16$

0	-1	-1	-2	-2
A	A	B	B	A
$n+14$	$n+12$	$n+15$	$n+18$	$n+16$