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

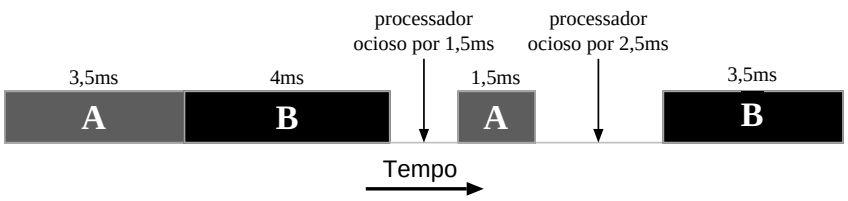
Quarto Período
Gabarito da AD1 - Primeiro Semestre de 2018

Atenção: 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 -

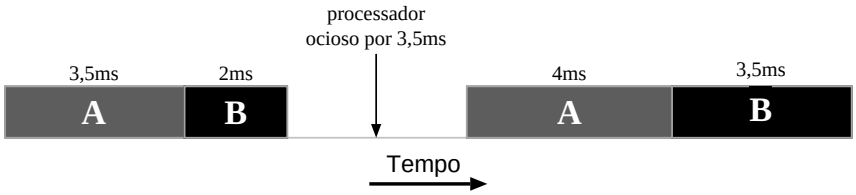
---

1. (1,5) Suponha que a multiprogramação seja usada somente para evitar a ociosidade do processador quando o programa em execução faz uma operação de E/S, e que os programas A e B tenham sido executados como na figura a seguir. Responda, justificando a sua resposta:



(a) (1,0) O que ocorrerá com o tempo de ociosidade se o programa A executar por mais 2,5ms após fazer a sua operação de E/S, o programa B executar por menos 2ms antes de fazer a sua operação de E/S, e A continuar executando antes de B?

Resp.: Teremos a execução dada na figura a seguir, onde o tempo de ociosidade é de 3,5ms.



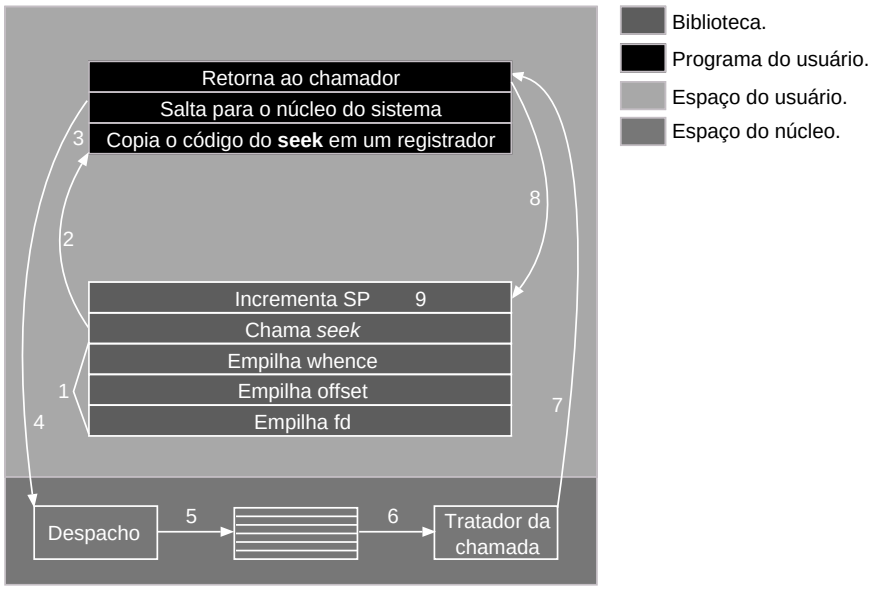
(b) (0,5) Em quanto o tempo de ociosidade aumentará se a multiprogramação deixar de ser usada? Esse tempo depende da ordem de execução dos programas no processador?

Resp.: Se a multiprogramação não for usada, então o tempo de ociosidade do processador será a soma dos tempos de todas as operações de E/S feitas. Como, pela figura, os tempos das operações de E/S de A e B são ambos iguais a 5,5ms, então o tempo de ociosidade do processador será de 11ms.

2. (1,5) Na aula 2 vimos os passos executados ao chamarmos a função da biblioteca *read*, a qual implementa a chamada ao sistema operacional **read**. Quais serão os passos executados se desejarmos fazer a chamada ao sistema operacional **seek**, usando a função da biblioteca *seek*, para a qual passamos o descritor do arquivo no sistema de arquivos, dado em *fd*, o deslocamento, dado em *offset*, e o modo como o deslocamento é usado, dado em *whence*?

Resp.: A seguir mostramos a figura obtida, similar à dada no último slide da aula 2, ao fazermos a chamada ao sistema operacional **seek**. No passo 1, o processo do usuário que executou a função *seek* empilha

os parâmetros *fd*, *offset* e *whence* passados a essa função. Após empilhar os parâmetros o processo, no passo 2, chama a função da biblioteca *seek*. Após ser chamada, esta função então coloca, no passo 3 e em um lugar pré-determinado pelo sistema operacional, o código que identifica a chamada ao sistema operacional **seek**. Depois disso, no passo 4, essa função executa a instrução TRAP do processador, o que mudará o processador do modo usuário para o modo supervisor e fará com que o controle seja transferido para o endereço do núcleo responsável pelo tratamento das chamadas ao sistema operacional. No passo 5, a parte do núcleo responsável por tratar as chamadas obtém, usando o código (passado pela biblioteca) como um índice em uma tabela com os endereços das funções que executam as chamadas, o endereço da função do núcleo que executa a chamada **seek**. Então, no passo 6, o sistema operacional executa esta função, denominada de **tratador da chamada seek**. Depois de esse tratador executar as tarefas necessárias para saltar para a posição *offset* do arquivo, identificado pelo descritor *fd*, segundo o critério dado em *whence* então, no passo 7, o processador será alternado do modo supervisor para o modo usuário, e o controle será passado à instrução, da função *seek* da biblioteca, posterior à instrução TRAP. Após fazer as finalizações necessárias depois de o salto ser feito no arquivo identificado pelo descritor *fd*, a função da biblioteca então passa, no passo 8, o controle novamente ao processo do usuário, na instrução seguinte à que chamou a função. Finalmente, no passo 9, o processo do usuário incrementa o ponteiro da pilha SP com o valor necessário para remover os parâmetros *fd*, *offset* e *whence* colocados na pilha antes de ser chamada a função *seek*.

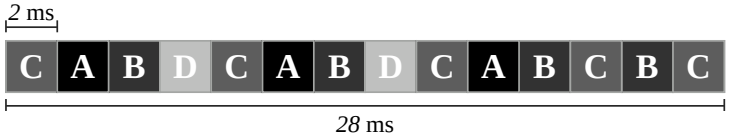


3. (2,0) Suponha que o sistema operacional esteja executando diretamente sobre o hardware de um computador cujas operações de E/S demoram 0,4ms. Suponha ainda que um processo tenha executado por 5000ms e que, durante a sua execução, tenha feito x operações de E/S. Se o sistema operacional agora executar sobre uma máquina virtual que reduza a velocidade do processador em 35% e a velocidade das operações de E/S em 60%, qual deverá ser o número x de operações de E/S feitas sobre o hardware do computador, supondo que o processo tenha executado $x+1000$ operações de E/S na máquina virtual, e que o tempo total de execução na máquina virtual tenha sido de 10000ms? Justifique a sua resposta.

Resp.: Como o tempo total de execução é de 5000 ms, e como o processo faz x operações de E/S com duração de 0,4ms, então $0,4x$ ms dos 5000 ms são gastos com operações de E/S, quando a execução ocorre sobre o hardware do computador. Logo, o processo executa no processador do hardware por $(5000 - 0,4x)$ ms. Note que a velocidade do processador ser reduzida em 35% significa que a velocidade do processador virtual é 65% da velocidade do processador real, o que por sua vez significa que, durante os $(5000 - 0,4x)$ ms, somente 65% das instruções são executadas. Com isso, quando o processo executa sobre a máquina

virtual, seu tempo de execução no processador virtual é de $\frac{5000-0,4x}{0,65}$ ms. Agora, como o processo executa $x + 1000$ operações de E/S na máquina virtual, e como o novo tempo de cada operação de E/S é de $\frac{0,4}{0,4} = 1$ ms (já que, similarmente à redução do tempo do processador, a redução da velocidade de cada operação de E/S em 60% significa que no mesmo tempo podemos, na máquina virtual, executar somente 40% das operações de E/S originais), então $(x+1000) \times 1 = x+1000$ ms dos 10000 ms do tempo de execução do processo na máquina virtual são gastos com E/S. Logo, o tempo de execução do processo no processador virtual é de $10000 - (x + 1000) = (9000 - x)$ ms e, com isso, podemos concluir que $\frac{5000-0,4x}{0,65} = 9000 - x$, ou seja, $x = 3400$, implicando que foram executadas 3400 operações de E/S sobre o hardware do computador e 4400 operações de E/S sobre a máquina virtual.

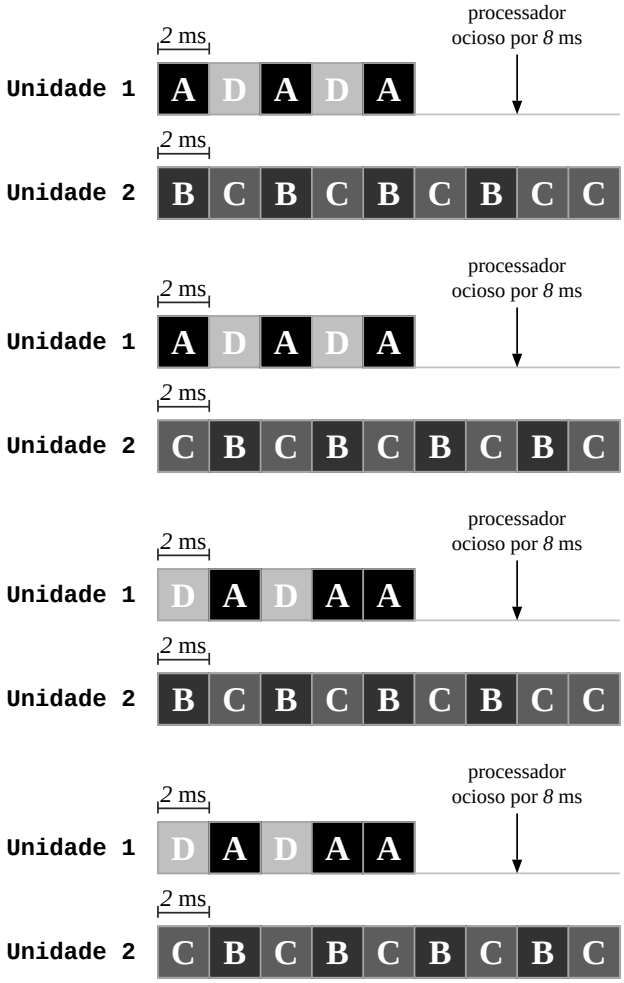
4. (1,5) Suponha que quatro processos, A, B, C e D, tenham sido executados como na figura a seguir, em um computador com somente uma unidade de processamento. Suponha ainda que o escalonador coloque um processo para executar novamente no processador por mais 2ms somente após todos os outros processos alocados àquele processador terem tido a chance de executar por 2ms. Responda, justificando a sua resposta:



   (a) (1,0) Como será a execução se agora existirem duas unidades de processamento, e se supusermos que os processos A e D sempre executem na unidade 1 e os processos B e C sempre executem na unidade 2?

   Resp.: Devido a existirem duas ordens iniciais para a unidade 1, A antes de D ou A depois de D, e duas ordens para a unidade 2, B antes de C ou B depois de C, existem quatro possíveis ordens de execução, mostradas na figura a seguir. Devido a todas elas serem equivalentes, a sua resposta será considerada correta se você

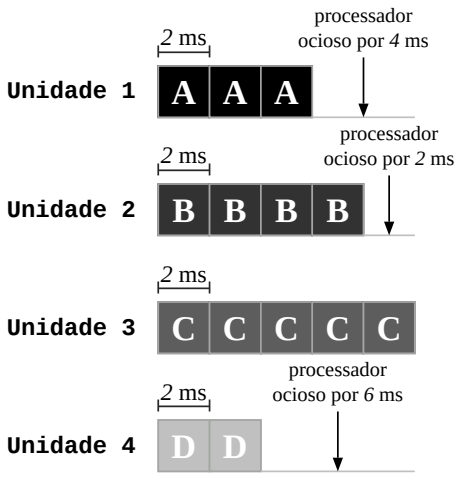
somente der uma delas.



(b) (0,5) Como será a execução dos processos se existirem quatro unidades de processamento, supondo que um processo sempre execute na mesma unidade?

Resp.: Devido a cada processo poder executar em uma das quatro unidades, existem 4!=24 possíveis escolhas mas, como ao contrário do item anterior, a única diferença entre elas é que os processos estarão em unidades diferentes, sempre executando na mesma unidade, mostramos na figura a seguir somente uma possibilidade,

supondo que A execute na unidade 1, B na unidade 2, C na unidade 3 e D na unidade 4. Assim como no item (a), você somente precisa dar uma ordem para a sua resposta ser considerada correta.



5. (2,0) Suponha que uma fila, que pode armazenar até n números, e que um conjunto, que pode armazenar um número ilimitado de números, sejam compartilhados por três processos, A, B e C. O processo A continuamente insere m números no conjunto, o processo B continuamente remove dois números do conjunto, calcula o produto deles, e depois coloca o produto no final da fila. Finalmente, o processo C remove três números do início da fila e imprime na tela a soma desses números. Como podemos garantir a correta execução dos processos A, B e C, usando o menor número possível de semáforos de contagem? Justifique a sua resposta. Suponha que o conjunto possua um procedimento $AdicionaNumero(a)$ para inserir a no conjunto e a função $RemoveNumero()$ para remover e retornar um número escolhido ao acaso do conjunto, e que a fila possua um procedimento $InsereFinal(b)$ para inserir b no final da fila e a função $RemoveInicio()$ para remover e retornar o número no início da fila.

Resp.: Vamos chamar o conjunto de X e a fila de Y. Como não foi definido o total inicial de números em X e em Y, vamos supor que X foi criado com $x \geq 0$ números e que Y foi criada $0 \leq y \leq n$ números.

Vamos precisar de cinco semáforos de contagem, sendo dois para X e três para Y. O semáforo $acesso_X$ é usado para garantir o acesso exclusivo a X, e é inicializado com 1 para simular um semáforo binário. O semáforo $conta_X$ conta o total de números em X, e é inicializado com x. Já o semáforo $acesso_Y$, usado para garantir o acesso exclusivo a Y, é inicializado com 1 novamente para simular um semáforo binário. O semáforo $conta_Y$ conta o total de números em Y, e é inicializado com y. Finalmente, o semáforo $livres_Y$ conta o total de números que ainda podem ser inseridos em Y, sendo inicializado portanto com $n - y$. A seguir mostramos os pseudocódigos para os processos A, B e C, usando os semáforos descritos e as funções definidas no enunciado.

```
void ProcessoA(void)
{
  while(1)
  {
    // Garante o acesso exclusivo a X.
    P(acesso_X);
    for(i = 0; i < m; i++)
    {
      // Código para retornar um novo número em a.
      // Insere a em X.
      AdicionaNumero(a);
      // Usa a operação V sobre conta_X para registrar que a foi inserido em X.
      V(conta_X);
    }
    // Libera o acesso exclusivo a X.
    V(acesso_X);
  }
}
```

```
void ProcessoB(void)
{
  while(1)
  {
    // Usa a operação P sobre conta_X para garantir que pelo menos dois
    // números existam em X.
    P(conta_X);
    P(conta_X);
    // Garante o acesso exclusivo a X.
    P(acesso_X);
    // Remove dois números de X e os armazena em a_1 e a_2.
    a_1 = RemoveNumero();
    a_2 = RemoveNumero();
    // Libera o acesso exclusivo a X.
    V(acesso_X);
    // Garante que exista espaço em Y para armazenar o produto de a_1 e a_2.
    P(livres_Y);
    // Garante o acesso exclusivo a Y.
    P(acesso_Y);
    // Insere o produto a_1a_2 em Y.
    InsereFinal(a_1a_2);
    // Libera o acesso exclusivo a Y.
    V(acesso_Y);
    // Usa a operação V sobre conta_Y para registrar que o produto de a_1 e a_2
    // foi inserido em Y.
    V(conta_Y);
  }
}
```

```
void ProcessoC(void)
{
  while(1)
  {
    // Usa a operação P sobre conta_Y para garantir que pelo menos três
    // números existam em Y.
    P(conta_Y);
    P(conta_Y);
    P(conta_Y);
    // Garante o acesso exclusivo a Y.
    P(acesso_Y);
    // Remove três números do início de Y e os armazena em a_1, a_2 e a_3.
    a_1 = RemoveInicio();
    a_2 = RemoveInicio();
    a_3 = RemoveInicio();
    // Código para imprimir a_1 + a_2 + a_3 na tela.
    // Libera o acesso exclusivo a Y.
    V(acesso_Y);
    // Usa a operação V sobre livres_Y para registrar que três números
    // foram removidos de Y.
    V(livres_Y);
    V(livres_Y);
    V(livres_Y);
  }
}
```

6. (1,5) Suponha que o sistema operacional use o algoritmo por prioridades, sendo que a prioridade de um processo é dobrada a cada a ms de execução no processador, e que o processo com a menor prioridade execute no processador até existir um outro processo com prioridade menor. Como será a ordem de execução de três processos, A, B e C, se eles precisarem executar por, respectivamente, $6a$ ms, $8a$ ms e $4a$ ms, e se as suas prioridades iniciais forem de, respectivamente, 10, 18 e 2? Justifique a sua resposta.

   Resp.: Pelo enunciado, vemos que a ordem de execução dos processos é como dada nas tabelas a seguir. Em cada tabela, mostramos como os processos são escolhidos pelo algoritmo, sendo que cada coluna refere-se à execução de um processo dando o tempo de início, o processo correspondente, e a prioridade cujo valor definiu que o processo deveria continuar executando no processador. Como podemos ver pelas tabe-

las, a ordem de execução será CCCACBABABABABABBB.

0	a	$2a$	$3a$	$4a$	$5a$	$6a$	$7a$	$8a$	$9a$
C	C	C	A	C	B	A	B	A	B
2	4	8	10	16	18	20	36	40	72

$10a$	$11a$	$12a$	$13a$	$14a$	$15a$	$16a$	$17a$
A	B	A	B	A	B	B	B
80	144	160	288	320	576	1152	2304