GABARITO
P1 - Sistemas Operacionais I
Professor: Leandro Marzulo
2013-2

1) (3,0) Com relação às técnicas de E/S, responda:

a) (1,5) 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? Qual o nome da técnica concebida para eliminar este problema?

Resp.: Quando não existir uma controladora de 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. O nome da técnica é DMA (Direct Memory Access).

b) (1,5) Qual a vantagem de E/S com pooling em relação a E/S controlada por interrupção?

Resp.: Na E/S com polling o processador fica dedicado a verificar o estado da operaçãoo de E/S. Desta forma, o término das operações de E/S é detectado praticamente de forma instantânea e o processo que aguardava pelo término da operação segue com a sua execução. Já na E/S controlada por interrupção, processos que solicitam E/S são bloqueado, liberando o processador para executar outras tarefas. Quando a E/S termina, o processador será interrompido e o processo que estava aguardando será colocado no estado pronto. Ele voltará a executar dependendo do mecanismo de escalonamento usado.
E/S com pooling pode ser mais interessante para processos com restrições de execução em tempo real, além de apresentar menor overhead, pois não há tratamento de interrupções.

2) (4,0) Considere dois processos A e B que serão executados em uma máquina com apenas 1 núcleo de processamento. Suponha que tanto A quanto B usem, cada um, duas rajadas de CPU e uma rajada de E/S e que A é iniciado T u.t antes de B (T≥0). Vamos chamar de A_{CPU1}, A_{CP2}, B_{CPU1} e B_{CPU2} as rajadas de CPU de A e B e vamos chamar de $A_{E/S}$ e $B_{E/S}$ as rajadas de E/S de A e B. Responda:

a) (2,0) Qual seria o tempo de ociosidade do processador (se houver) do cenário em questão, caso a o sistema operacional usado seja anterior ao da terceira geração? Justifique.

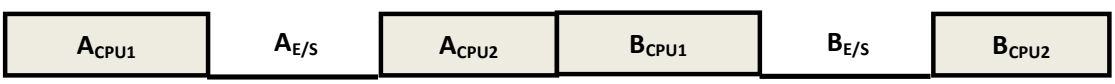
Resp.: Sistemas anteriores ao da terceira geração não implementam o conceito de multiprogramação, ou seja, quando um processo começa a executar ele vai até o final do processo. Se o processo solicitar E/S o processador ficará ocioso (usa E/S com pooling). Para determinar o tempo de ociosidade temos que considerar os tempos de E/S de A e B ($A_{E/S}$ e $B_{E/S}$) e o tempo T entre a criação de A e B.

Se T > 0, o processo A executa primeiro. Neste caso, temos duas possiblidades:

- **B é colocado para executar no processador logo após o termino de A, o que significa que B é criado antes do término de A, ou seja:**

  $T \leq (A_{CPU1} + A_{E/S} + A_{CPU2})$

  Assim temos:

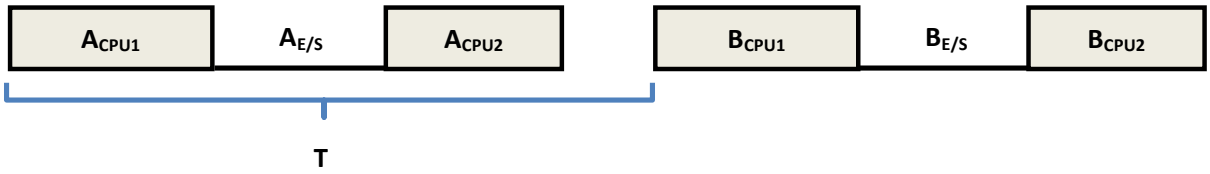


Tempo de ociosidade = $A_{E/S} + B_{E/S}$

- **B é colocado para executar no processador algum tempo depois do termino de A, o que significa que B é criado depois do término de A, ou seja:**

  $T > (A_{CPU1} + A_{E/S} + A_{CPU2})$

  Assim temos:



  Tempo de ociosidade = $T - A_{CPU1} - A_{CPU2} + B_{E/S}$

Se T=0, B pode começar antes de A, resultando na seguinte situação:

Tempo de ociosidade = $B_{E/S} + A_{E/S}$

b) (2,0) Quais são as condições necessárias para eliminar a ociosidade do processador do cenário em questão, caso a o sistema operacional usado seja de terceira geração? Justifique.

Resp.: Sistemas de terceira geração implementam o conceito de multiprogramação, ou seja, quando um processo solicitar E/S o processador ficará livre para executar outras tarefas. Nesse caso, a ociosidade só será eliminada se tivermos outros processos para executar durante todo o tempo de E/S.

Se $T > A_{CPU1}$ teremos ociosidade, pois B não existirá ainda quando A for fazer E/S. Logo, A primeira condição necessária para eliminar a ociosidade é:

$T \leq A_{CPU1}$

Além disso, temos que avaliar as seguintes possibilidades:

- **Se T>0 A começa a executar antes. Logo, para eliminar a ociosidade, precisamos que a execução seja da seguinte forma:**

  Para que isso aconteça, precisamos satisfazer as seguintes condições:
  $A_{E/S} \leq B_{CPU1}$
  $B_{E/S} \leq A_{CPU2}$

- **Se T=0 B Pode começa a executar antes. Logo, para eliminar a ociosidade, precisamos que a execução seja da seguinte forma:**

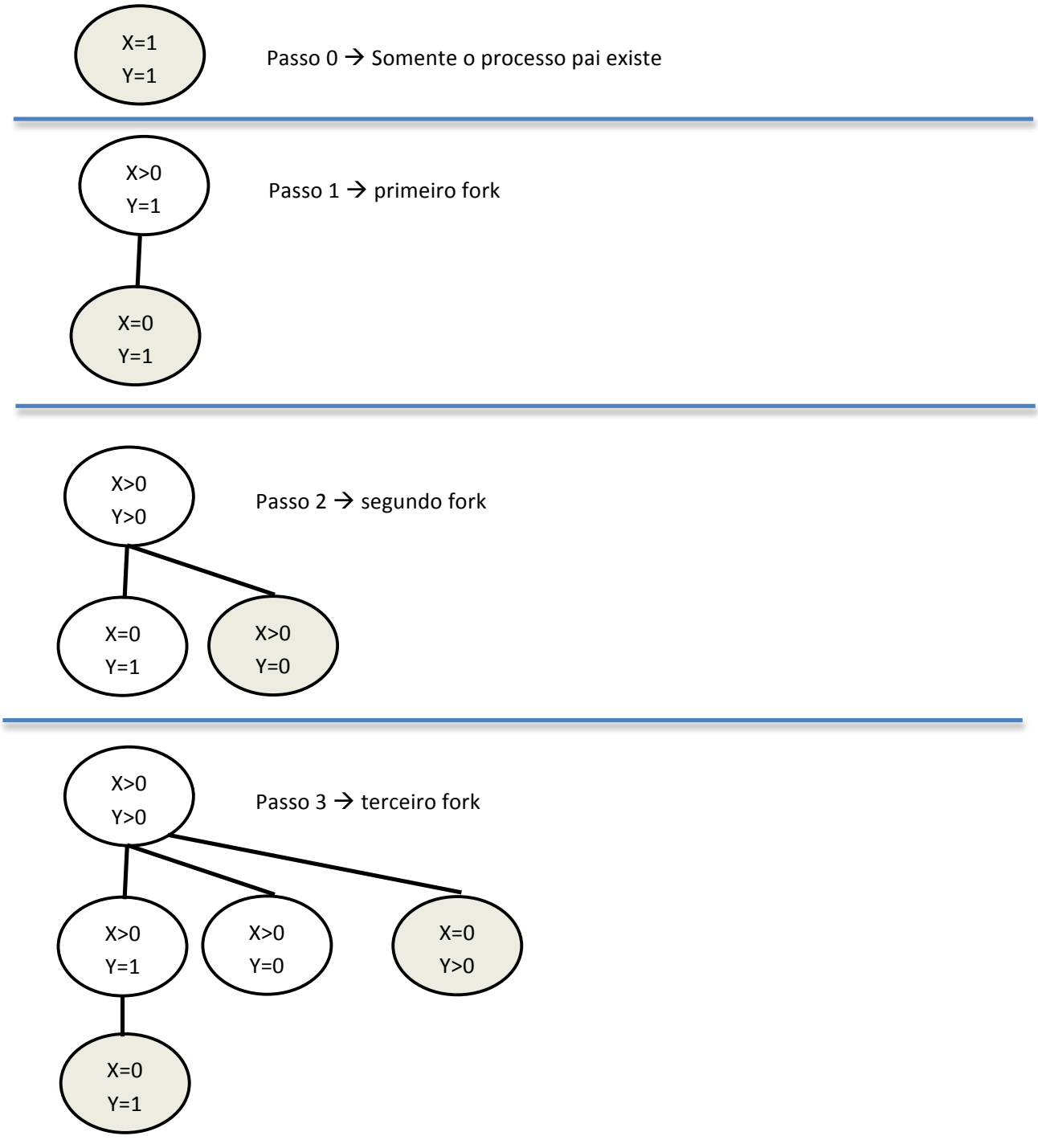
  Para que isso aconteça, precisamos satisfazer as seguintes condições:
  $B_{E/S} \leq A_{CPU1}$
  $A_{E/S} \leq B_{CPU2}$

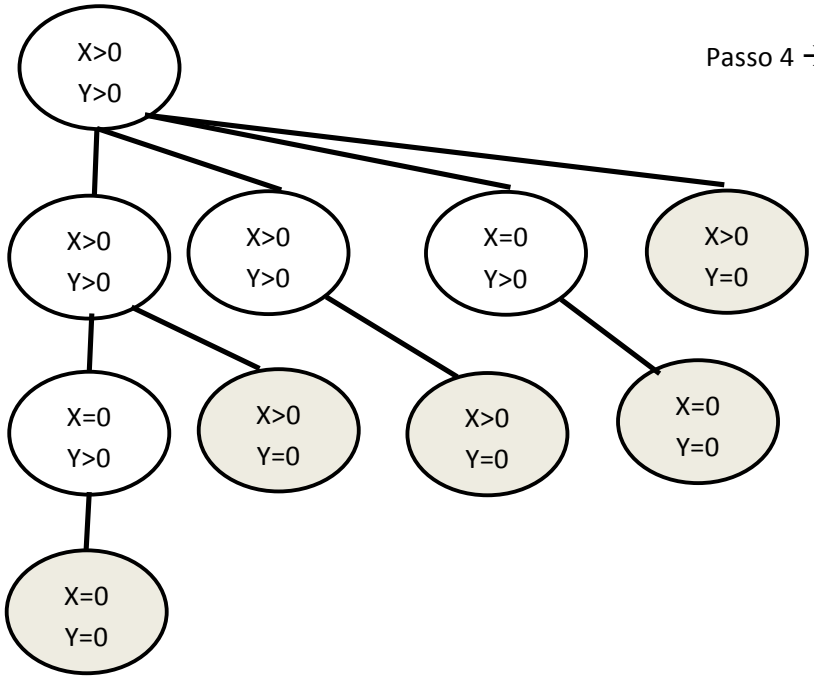
3) (2,0) Considerando o trecho de código abaixo, responda:

```
pid_t x=1,y=1;
x = fork();
if (x > 0) y = fork();
if (y > 0) x = fork();
y = fork();
if (y == 0) fork();
```

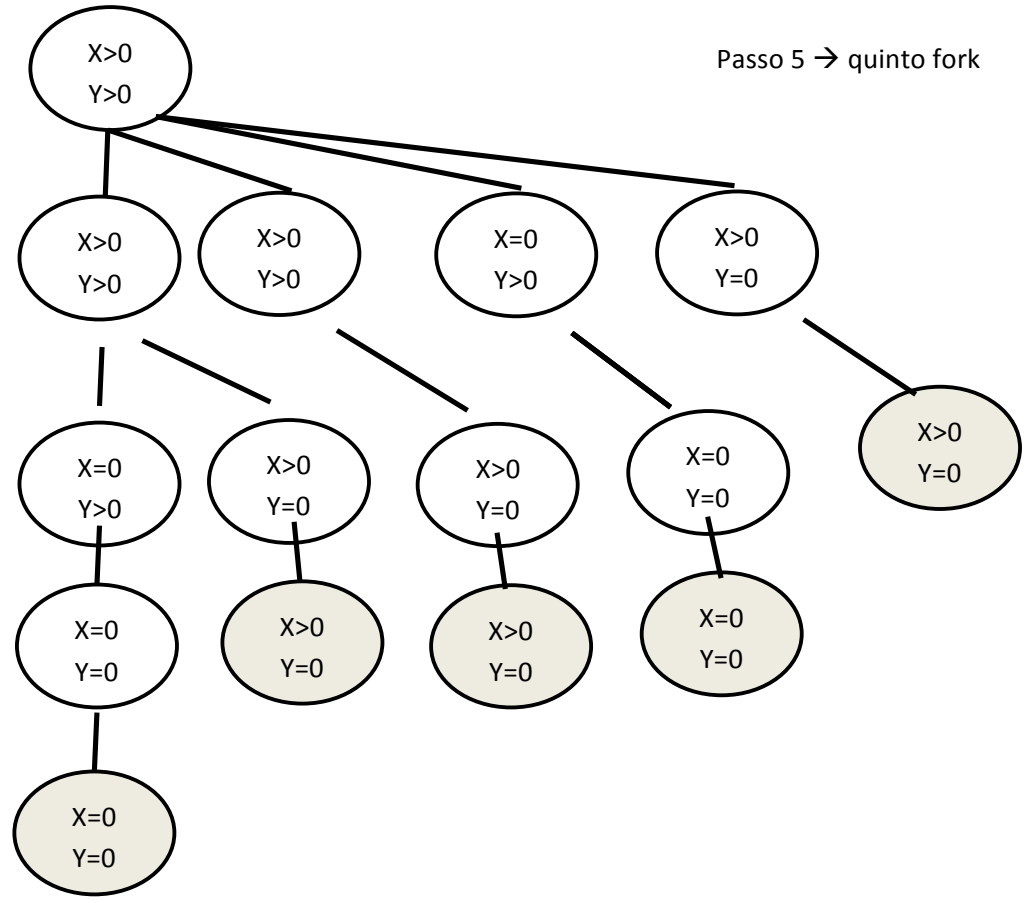
a) (1,5) Mostre o passo a passo da criação da árvore de processos do processo inicial até que todos os filhos sejam criados (considerando que nenhum deles terminou e assumindo que não houve falha na execução de nenhuma chamada fork())? Mostre em cada passo, para cada nó da árvore (processo) os valores das variáveis x e y.

Resp.:





Passo 4 → quarto fork



Passo 5 → quinto fork

b) (0,5) Ao executar este trecho, quantos processos são criados, incluindo o processo inicial, assumindo que não houve falha na execução de nenhuma chamada fork()?

Resp.: 15

4) (3,0) Em relação diagrama de processos de 5 estados (pronto, executando, bloqueado, novo e terminado) responda:

a) (1,5) Faria sentido adicionar uma transição do estado de pronto para o estado bloqueado? Justifique.

Resp.: Não, pois um processo só é bloqueado quando ele solicita uma operação de E/S, faz uma chamada de sistema bloqueante ou quando ocorre uma exceção. Isso só pode acontecer se o processo estiver executando

b) (1,5) Em quais circunstâncias um processo pode sair do estado de executando e para qual estado ele é enviado em cada situação?

Resp.:

Executando → Terminado : O processo termina
Executando → Pronto: Ocorre uma interrupção
Executando → Bloqueado : O processo solicita E/S, faz uma chamada de sistema ou lança uma exceção.