GABARITO
P1 - Sistemas Operacionais I
Professor: Leandro Marzulo
2013-1

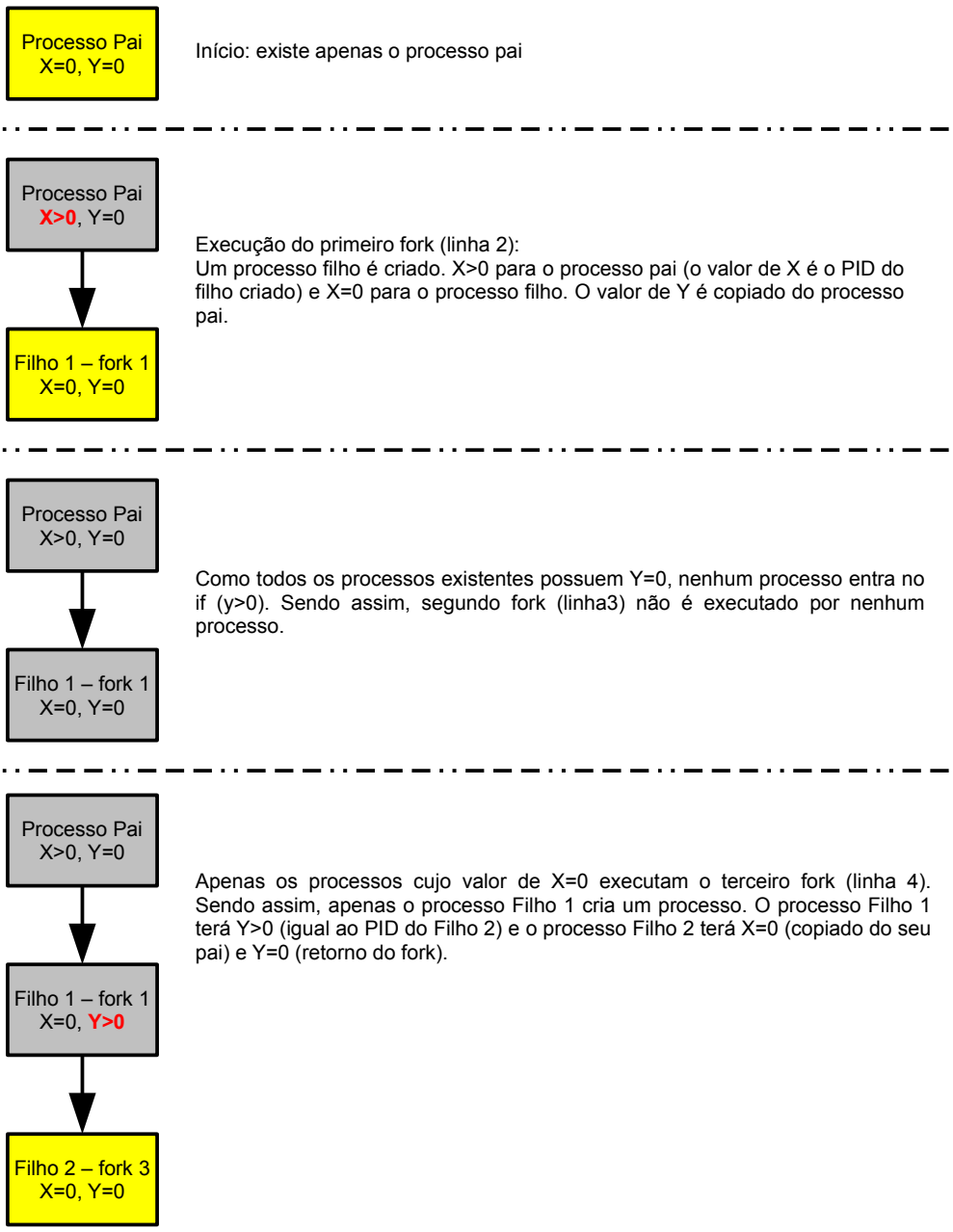
Nome:

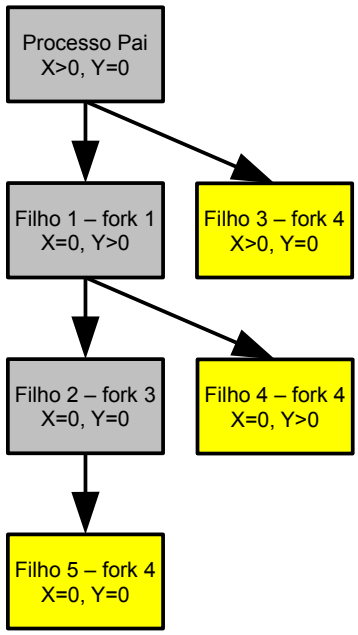
Instruções: Esta prova é composta de quatro questões totalizando 12 (doze) pontos, sendo a nota máxima 10 (dez). Responda as questões de forma sucinta e clara. O uso de lápis é permitido, no entanto, pedidos de revisão serão considerados apenas para questões respondidas a caneta. BOA PROVA!

1) (1,5) Considerando o trecho de código abaixo, responda:

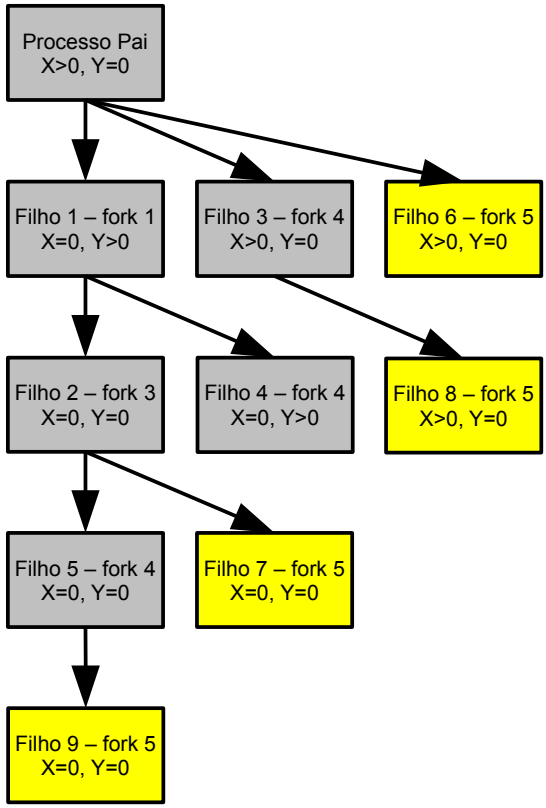
```
pid_t x=0, y=0;
x = fork();
if (y > 0) fork();
if (x == 0) y = fork();
fork();
if (y == 0) fork();
```

a) (1,0) Como fica a árvore de processos do processo inicial, com todos os filhos criados (considerando que nenhum deles terminou e assumindo que não houve falha na execução de nenhuma chamada fork())?





Todos os processos fazem o quarto fork (linha 5). Como o retorno do fork não é atribuído a nenhuma variável, os valores de X e Y para os processos criadores ficam inalterados e são copiados para os filhos criados.



Apenas os processos que possuem Y>0 executam o quinto fork (linha 6). Sendo assim, o processo pai e os filhos 2, 3 e 5 criam processos.

Como o retorno do fork não é atribuído a nenhuma variável, os valores de X e Y para os processos criadores ficam inalterados e são copiados para os filhos criados.

Esta é a árvore final!

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.: 10 processos são criados

2) (4,0) Com relação às técnicas de E/S, responda:

a) (1,0) Poderíamos ter multiprogramação sem interrupções? Justifique.

Resp.: Não, pois 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.

b) (1,0) O que é DMA? Porque podemos dizer que o uso de DMA maximiza os benefícios da multiprogramação?

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, sem DMA, 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.

c) (2,0) Em um sistema operacional no qual um processo só pode estar no estado PRONTO ou EXECUTANDO, há alguma vantagem no uso de DMA? Justifique.

Resp.: Um sistema operacional que não possui o estado BLOQUEADO para seus processos é um sistema operacional sem multiprogramação. Sendo assim, não é possível fazer operações de E/S enquanto outros processos são executados. Portanto, não faz sentido algum em usar DMA, pois o processador ficaria aguardando da mesma forma o término da cópia dos dados entre a memória da controladora do dispositivo e a memória principal.

3) (5,0) Considere o trecho de código em C abaixo e responda:

```
 1   pid_t pid;
 2    int x=30;
 3    int sid_a = shmget(IPC_PRIVATE,
sizeof(int)*2, SHM_R|SHM_W|IPC_CREAT);
 4    int *a = (int *) shmat(sid_a, NULL, 0);
 5    a[0]=0;
 6    a[1]=0;
 7    shmdt(a);
 8    pid = fork();
 9    if (pid == 0) {
10         a = (int *) shmat(sid_a, NULL, 0);
11         printf("%d\n",x);
12         x=60;
13         printf("%d\n",x);
14         a[1]=50;
15         a[0]=10;

16         shmdt(a);
17    }
18    else {
19         a = (int *) shmat(sid_a, NULL, 0);
20         while (a[0]!=10);
21         printf("%d - %d\n",a[1], x);
22         shmdt(a);
23         shmctl(sid_a, IPC_RMID, NULL);
24    }
```

a) (1,0) O que é impresso na tela após a execução desse trecho de código, considerando que ele será executado em um monoprocessador?

Resp.: Linha 11 → 30
Linha 13 → 60
Linha 21 → 50 – 30

b) (1,0) O que é feito na linha 3 do programa?

Resp.: É criada uma região de memória compartilhada com permissão de leitura e escrita do tamanho de 2 elementos do tipo inteiro. O identificador do segmento da região criada é armazenado na variável sid_a.

c) (1,0) O que é feito nas linhas 4, 10 e 19 do programa?

Resp.: Nessas linhas é feita a anexação da região de memória compartilhada ao processo que deseja usar os dados. Na linha 4 a anexação é feita pelo processo pai para inicializar os dados, na linha 10, a anexação é feita pelo processo filho, que altera os valores originais e na linha 19 o pai faz novamente a anexação para monitorar a alteração de a[0]. Em todas essas linhas, a anexação é feita com o identificador de segmento obtido na linha 3 e um ponteiro para a região anexada é retornado pela função shmat e atribuido a variável a.

d) (1,0) O que é feito nas linhas 17, 16 e 22 do programa?

Resp.: Nessas linhas é feita a desanexação da região de memória compartilhada anexada anteriormente (conforme explicado no item c).

e) (1,0) O que é feito na linha 23 do programa?

Resp.: Nessa linha é feita a destruição da região de memória compartilhada, através da função shmctl (CTL = control), usando o flag IPC_RMID (RMID = Remove Identifier)

4) (1,5) Suponha que dois processos, A e B, estejam em execução no mesmo processador de tal forma que o sistema operacional somente troca um processo pelo outro quando o primeiro executa uma operação de E/S. Se A e B fazem, cada um, apenas uma operação de E/S, e se A é o primeiro a ser executado, sob quais condições a execução com multiprogramação resulta no dobro do desempenho?

Sabemos que os dois processos fazem apenas uma operação de E/A, que o processo A executa primeiro e que os dois processos executam em apenas um processador. Essas características são fixas.

Consideremos que o tempo total de execução do processo A é dividido da seguinte forma:
A1 = Tempo de execução no processador do primeiro trecho
A2 = Tempo da primeira operação de E/S
A3 = Tempo de execução no processador do segundo trecho

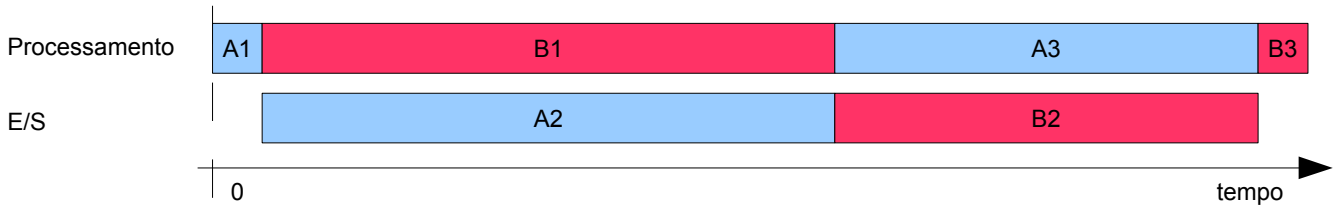
Analogamente, temos para o processo B:
B1 = Tempo de execução no processador do primeiro trecho
B2 = Tempo da primeira operação de E/S
B3 = Tempo de execução no processador do segundo trecho

Em um sistema sem multiprogramação, o tempo total de execução dos dois processos seria dado por A1 + A2 + A 3 + B1 + B2 + B3, pois o processador aguarda o término de cada operação de E/S

Em um sistema com multiprogramação, temos a superposição de E/S com a execução de outros processos. Queremos obter o dobro de desempenho, ou seja, executar os dois processos na metade do tempo. Para que isso aconteça, Precisamos que:

- A2 = B1;
- B2 = A3;

Desta forma, não teremos tempo de ociosidade e garantiremos que assim que um processo terminar de fazer E/S o outro já terá terminado, conforme na figura abaixo:



Podemos portando dizer que o tempo total de execução, nesta situação é dado por A1 + B1 + A3 + B3

Queremos que (A1 + A2 + A 3 + B1 + B2 + B3) / 2 = A1 + B1 + A3 + B3.

Como B1= A2 e A3=B2, temos:

(A1 + B1 + A 3 + B1 + A3 + B3) / 2 = A1 + B1 + A3 + B3
(A1 + 2 * B1 + 2 * A3 + B3) / 2 = A1 + B1 + A3 + B3
(A1/2) + (2 *B1/2) + (2*A3/2) + (B3/ 2) = A1 + B1 + A3 + B3
(A1/2) + B1 + A3 + (B3/ 2) = A1 + B1 + A3 + B3
(A1/2) + (B3/ 2) = A1 + B3 → a única forma desta igualdade ser satisfeita é se A1 e B3 forem 0.

Ou seja, para que o tempo total seja reduzido pela metade, sempre é preciso estar executando um processo enquanto o outro faz E/S. Entretanto, quando extamos executando A1, B não está fazendo E/S. Da mesma forma, quando estamos executando B3, A não está fazendo E/S. Podemos dizer então que para obter o dobro do desempenho, A1 e B3 devem tender a zero.