UFES – Departamento de Informática

1ª. Prova de Sistemas Operacionais – 2015/1 – Profª. Roberta L. Gomes

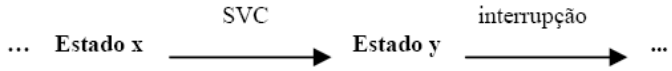
Nome: _____

1) (2,0) A partir do diagrama completo de transição de estados para processos em UNIX, apresente uma possível sequência de estados referente ao seguinte histórico de um processo:

"O processo A foi criado e iniciou a execução de instruções comuns (toda instrução que não corresponde a uma chamada de sistema), sem deixar a CPU até a ocorrência de uma interrupção, que faz com que o mesmo seja preemptado. Antes de rodar novamente, sofre ação do escalonador de médio prazo (swapper). Em algum momento retorna à memória, voltando em seguida a executar. Ao ganhar a CPU novamente, o processo prosseguiu executando instruções comuns até que realiza uma requisição de E/S, a qual demanda um tempo 'longo' para ser atendida. Enquanto aguardava a operação de E/S, um outro processo (B) enviou um SIGNAL ("kill..."), causando o término do processo A."

Nota1: Considere os estados do diagrama completo UNIX como sendo: Initial (*new*), Ready (pronto), Ready Suspended (pronto suspenso), Blocked (bloqueado), Blocked Suspended (bloqueado suspenso), Kernel Running, User Running, Zombie (*exit*).

Nota 2: É necessário associar as transições presentes na seqüência de estados a cada evento listado no histórico. Um exemplo fictício de resposta é mostrado abaixo :



2) (2,0) Cinco processos A, B, C, D and E chegam nesta ordem ao mesmo tempo com os seguintes CPU burst e prioridades (menor valor significa maior prioridade):

	CPU Burst	Prioridade
A	3	3
B	7	5
C	5	1
D	2	4
E	6	2

Preencha o quadro abaixo com o tempo de espera (*Waiting Time*) de cada processo, e o tempo de espera médio (*Average Waiting Time*) para cada algorítmo de escalonamento indicado (ignore o overhead devido a trocas de contexto).

Scheduling Policy	*Waiting Time*					*Average Waiting Time*
	A	*B*	*C*	*D*	*E*	
First-Come-First-Served						
Non-Preemptive Shortest-Job First						
Priority						
Round-Robin (time quantum=2)						

3) (2,0) O espaço de endereçamento de um processo em execução no UNIX é dividido em “process space” e “kernel space”. Descreva o que é cada um deles (que estruturas de controle/segmentos se encontram neles). Você também deve explicar quando cada um deles pode ser acessado, em função do estado de execução do processo (“User Running” ou “Kernel Running”) e do contexto de execução (“Process Context” ou “System Context”).

4) (2,0) Um grafo de precedência é um grafo direcionado em que a relação (a) → (b) indica que 'a' precede 'b'. Neste exercício, os nós do grafo devem representar as instruções numeradas no código abaixo. Desta forma o gráfo de precedência representará a ordem em que as instruções serão executadas. Observe que se o programa não tivesse fork(), o grafo seria linear pois a execução desse programa seria uma sequência de instruções (com desvios ou não).
Desenhe o grafo de precedência referente ao código a seguir:

```
     int f1, f2, f3;
     int main(){
[1]    printf("Alo do pai\n");
[2]    f1 = fork();
[3]    if (f1>0)
[4]       execlp(“codigo_f”,“codigo_f”,NULL);
[5]    printf("Filho 1 criado\n");
[6]    f2 = fork();
[7]    if (f2>0)
[8]       execlp(“codigo_f”,“codigo_f”,NULL);
[9]    printf("Filho 2 criado\n");
[10]   if(waitpid(f1,null,0)<0);
[11]        printf("erro!\n");
[12]   f3 = fork;
[13]   if (f3==0)
[14]        execlp(“codigo_f”,“codigo_f”,NULL);
[15]   printf("Filho 3 criado\n");
[16]   waitpid(f3,null,0);
[17]   printf("Filho 3 morreu\n");
[18]   waitpid(f2,null,0);
[19]   printf("Filho 2 morreu\n");
[20]   exit();
   }

//codigo_f
     int main()
[21]    printf("Aloha\n");
[22]    wait();
[23]    exit();
     }
```

5) (2,0) Quais das afirmações seguintes são verdadeiras e quais são falsas? Justifique/explique suas respostas.
A) Sistemas de tempo real geralmente usam escalonamento de processador (curto prazo) não-preemptivo.
B) Sistemas de tempo compartilhado geralmente necessitam que o kernel do S.O. seja preemptivo.

Boa Prova!