Sistemas Operacionais – 2010/1

Questões de Provas de Períodos Anteriores

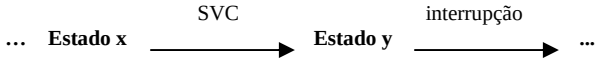
1) Dois importantes conceitos encontrados no sistema operacional Unix são: *(i) modo de execução* (“execution mode”); e *(ii) contexto de execução* (“execution context”). Discorra sobre eles.
2) Com relação ao sistema operacional Unix, desenhe a porção da sua máquina de estados que contém os estados “*User Running*”, “*Kernel Running*”, “*Ready to Run*” e “*Asleep*”. Caracterize um deles e explique as transições entre os mesmos. Relacione-os também com a ação de atribuição de prioridades pelo escalonador.
3) Descreva o procedimento geral de tratamento da interrupção do relógio (“*Clock Interrupt Handling*”) em sistemas Unix.
4) Descreva o esquema (tradicional) de escalonamento de processos do Unix. Avalie as suas vantagens e deficiências.
5) Fale sobre a criação de processos e a invocação de programas em sistemas Unix.
6) Explique o mecanismo geral de tratamento de interrupções no Unix, incluindo uma discussão sobre os seus diferentes níveis de prioridade.
7) Descreva o escalonador de processos do Unix SVR4. Avalie as suas vantagens sobre o escalonador tradicional.
8) O sistema operacional Linux utiliza dois algoritmos de escalonamentos, um de tempo compartilhado e outro baseado em prioridades, para tratar tarefas de tempo real. Cada processo é associado a uma classe de escalonamento. No algoritmo de escalonamento para processos de tempo compartilhado, o Linux usa um algoritmo por prioridades, baseado em créditos. Cada processo possui um determinado número de créditos (inicialmente, o número de créditos é igual à prioridade do processo); o processo com o maior número de créditos na fila de prontos é selecionado pelo escalonador. A cada interrupção do temporizador (1ms), o processo em execução perde um crédito; quando seu crédito chega a zero, o escalonador é ativado para selecionar outro processo para ganhar o processador. Se nenhum processo na fila de prontos tiver créditos, o algoritmo faz nova atribuição de créditos a todos os processos (inclusive aos processos bloqueados), de acordo com a seguinte regra: créditos := créditos /2 + prioridade.

   Avalie o algoritmo de escalonamento por prioridade usado pelo Linux quanto ao tempo médio de execução dos processos (calcule o *turnaround* de cada processo e faça a média). Para tanto, considere o seguinte volume de trabalho (processos de A a D chegam no sistema ao mesmo tempo):

ordem	processo	surto de CPU*	duração de I/O	tempo total de CPU	prioridade
1	A	2 ms	5 ms	6 ms	3
2	B	3 ms	10 ms	6 ms	3
3	C	---	---	14 ms	3
4	D	---	---	10 ms	3

(*) tempo de CPU necessário antes de cada solicitação de I/O (processos A e B ficam alternando entre surtos de CPU e em operações de I/O).

9) A partir do diagrama completo de transição de estados para processos em UNIX (ver figura a seguir), apresente uma possível seqüência de estados referente ao seguinte histórico de um processo:
"*O processo 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 requisição de E/S, a qual demanda um tempo 'longo' para ser atendida. Durante este tempo, o processo sofreu a ação do escalonador de médio prazo (swapper). Uma vez atendida a requisição de E/S, o processo voltou imediatamente a executar instruções comuns, até liberar a CPU para um outro processo. Ao ganhar a CPU novamente, o processo prosseguiu executando instruções comuns até o seu término*".
Nota: é necessário associar as transições presentes na seqüência de estados a cada evento listado no histórico. Um exemplo fictício de resposta é mostrado abaixo:



10) Explique porque o escalonar tradicional do Unix não é adequado a processos de tempo real. Por que ele não é escalável?

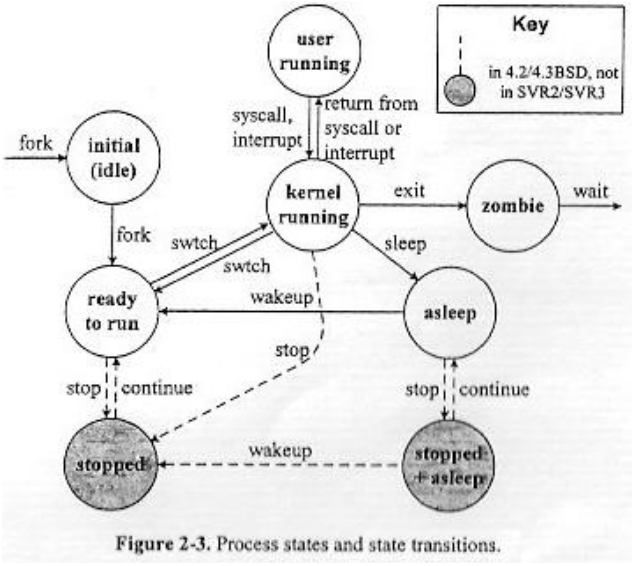


Figure 2-3. Process states and state transitions.

11) (1,0) Quantas vezes X é impresso no trecho do programa abaixo? Explique.

```
int x=10;
fork();
```

```
fork();
fork();
fork();
fork();
printf("x = %d \n",x);
```

12) (1,0) Dado do programa abaixo, desenhe a árvore que representa a hierarquia dos processos criados a partir da sua execução.

```
c2 = 0;
c1 = fork();
if (c1 > 0)
    c2 = fork();
if (c2 == 0)
    fork();
exit();
```

13) Com relação ao escalonador tradicional do Unix:
    a) Por que ele favorece processos I/O bound?
    b) Por que ele não é adequado a processos de tempo real?
    c) Por que ele não é escalável?

14) Discuta o efeito de cada um dos seguintes métodos de atribuição de quantum q no escalonamento circular (round-robin).
    a) q fixo e idêntico para todos os usuários
    b) q fixo e único para cada processo
    c) q variável e idêntico para todos os processos
    d) q variável e único para cada processo

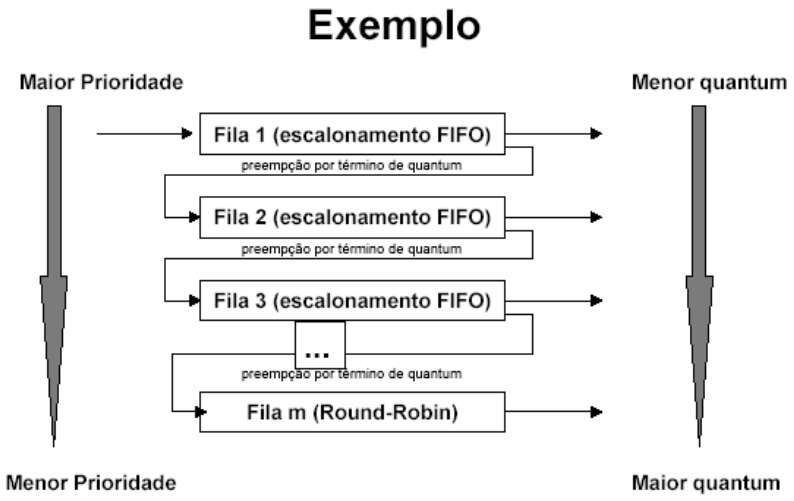
    Organize os sistemas em ordem crescente de sobrecarga de tempo de execução.

15) Dois objetivos comuns das políticas de escalonamento são minimizar tempos de resposta e maximizar utilização de recursos.
    a) Indique como esses objetivos se contrapõem um ao outro.
    b) Analise as políticas de escalonamento circular (*round robin*) e múltiplas filas com retorno/realimentação (*multiple queues with feedback*) com base nessas duas perspectivas.

16) Suponha um S.O. com escalonador de filas multiníveis na qual há cinco níveis. O quantum do primeiro nível é 0,5 segundos. Cada nível mais baixo tem um quantum de tamanho duas vezes maior que o quantum do nível anterior. Um processo não pode sofrer preempção até o seu quantum terminar. O sistema executa processos em lote (*cpu bound*) e interativos (*I/O bound*).
    a) Por que esse sistema é deficiente?
    b) Quais mudanças mínimas você proporia para tornar o esquema mais aceitável para o mix de processos que pretende?

17) Considere um sistema operacional cuja máquina de estados inclui os estados *Ready* e *Ready, Suspended*. Suponha que seja hora do S.O. despachar um processo e que existam nesse momento processos tanto no estado *Ready* como no estado *Ready, Suspended*, e que pelo menos um processo no estado *Ready, Suspended* possui prioridade maior do que qualquer processo no estado *Ready*. Duas políticas extremas seriam: *(a)* sempre despachar um processo no estado *Ready*, de forma a minimizar *swapping*; e *(b)* sempre dar preferência ao processo de mais alta prioridade, mesmo que isso possa significar a ocorrência de *swapping* quando este não é necessário. Sugira uma política intermediária (descreva-a e crie um algoritmo em pseudo-código) que tente balancear prioridade e desempenho.

18) Descreva a técnica de escalonamento mostrada abaixo. Apresente os méritos relativos dessa técnica no escalonamento de processos interativos, background, CPU-bound, I/O bound, real-time, etc.



19) Considere o seguinte conjunto de processos e os algoritmos de escalonamento FCFS, RR (com quantum = 2) e SJF preemptivo.

Process ID	CPU Burst (ms)
P1	20
P2	2
P3	4
P4	2
P5	10

Supõe-se que os processos entrem na fila de prontos na ordem P1, P2, P3, P4, e P5, todos no tempo zero. (a) Qual é o *turnaround time* de cada processo em cada um dos algoritmos acima? (b) Qual é o *waiting time* de cada processo em cada um dos algoritmos acima?

20) Considere um sistema que possui duas filas de escalonamento, com prioridades 0 e 1, sendo que somente pode ser escalonado um processo da fila de prioridade 1 não

existindo processos na fila de prioridade 0. Sabendo que o algoritmo utilizado nas duas filas é o Round-Robin, escreva o pseudo-código dos procedimentos *insere(p)*, onde p é o índice da tabela de descritores de processos e que possui um campo que contém a prioridade dos processos e *r= seleciona()*, que retorna o índice da tabela de descritores que descreve o processo selecionado. Cite duas situações em que cada procedimento é chamado.

21) Explique as diferenças no grau de favorecimento de processos curtos, feito pelos seguintes algoritmos de alocação:
(a) FCFS (b) RR (c) Multiple Queue with Feedback

22) Considere um sistema com um escalonador por revezamento (round-robin) e usuários em sua maioria interativos. Suponha que, na média, um processo seja executado 10ms até gerar uma operação de E/S. Quais dos seguintes valores de quantum você usaria? Por quê? q=1ms q=9ms q=11ms q=50ms

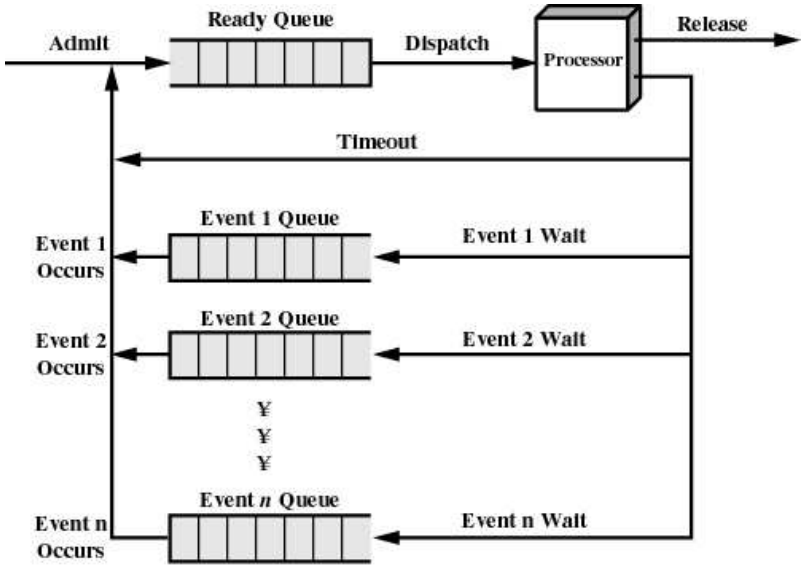
23) Em um certo sistema operacional, os seguintes estados são definidos para os processos: *execute* (*running*), *active* (*ready*), *blocked* e *suspended*. Um processo está *blocked* se ele está esperando por uma permissão para usar o recurso, e ele está *suspended* se ele está esperando por uma operação ser completada em um recurso que ele já adquiriu. Em vários sistemas operacionais esses dois estados são sobrepostos em um único estado *blocked*, e o estado *suspended* tem a definição usada em sala de aula. Compare os méritos relativos desses dois conjuntos de definições.

24) Suponha que os processos seguintes fiquem prontos para execução nos tempos indicados. Cada processo será executado pela UCP pelo tempo indicado. Na resposta às questões a seguir, use alocação não-preemptiva e tome todas as decisões com base na informação disponível no instante em que a decisão deve ser feita.

Processo	Duração da fase de uso da UCP	Tempo de chegada
P1	8	0,0
P2	4	0,4
P3	1	1,0

a. Desenho diagramas de Gantt ilustrando a execução desses processos usando os algoritmos de alocação FCFS e SJF.
b. Qual o tempo de processamento médio para esses processos com o algoritmo de alocação FCFS?
c. Qual o tempo de processamento médio para esses processos com o algoritmo de alocação SJF?

25) A figura abaixo sugere que um processo possa estar somente em uma fila de eventos de cada vez.
a. É possível permitir a um processo esperar em mais de uma fila de eventos ao mesmo tempo? Dê um exemplo.
b. Como você modificaria a estrutura de filas para suportar tal característica?



(b) Multiple blocked queues

26) Considere um sistema que deva executar muitos tipos de processos, incluindo grandes, pequenos, com uso intensivo da UCP, com uso intensivo de E/S, em lotes e interativos. Considere ainda os seguintes métodos de escalonamento: *Round-Robin*, FCFS, *Multiple Queue with Feedback*, SJF e STRF. Dada a lista de requisitos a seguir, diga qual a estratégia que melhor satisfaz a todos eles. Para cada uma das demais estratégias liste (e explique o porquê) pelo menos um requisito que não foi satisfeito.
    a. O número de processos interativos é relativamente pequeno. Eles deverão ter alta prioridade, mas são aceitáveis pequenos atrasos.
    b. Processos com uso intensivo de E/S devem ter prioridade alta para manter ativo os processadores de E/S.
    c. Processos com uso intensivo da UCP devem retardar processos com uso intensivo de E/S, mesmo que esses já estejam esperando há muito tempo.
    d. Quando os processos com uso intensivo da UCP são os únicos que estão prontos, a sobrecarga do sistema operacional deve ser minimizada.

27) Fale tudo sobre processos (programa, processo, PCB, imagem do processo, filas do sistema, operações sobre processos, escalonadores, etc.).

28) Suponha que os processos abaixo cheguem simultaneamente e de imediato passem a competir pela CPU. Considere ainda as seguintes abordagens de escalonamento: SJF preemptivo, FCFS e Round Robin.

Process ID	CPU *Burst* (ms)
1	6
2	2
3	4
4	8

a) Qual é o *turnaround time* de cada processo em cada um dos algoritmos acima?
b) Qual é o *waiting time* de cada processo em cada um dos algoritmos acima?

29) Considere um sistema operacional multiprogramado no qual existem três filas de escalonamento:
    0: de mais alta prioridade, na qual rodam os processos do sistema operacional;
    1: de prioridade intermediária, na qual rodam servidores especializados;
    2: de mais baixa prioridade, na qual rodam os programas de usuários.

Sabendo que cada uma das filas possui prioridade absoluta sobre as inferiores (isso é, somente roda um processo de prioridade 2 se não houver nenhum na fila de prioridade 0 nem na fila de prioridade 1), que na fila de prioridade 0 algoritmo utilizado é o FCFS e nas demais é o *round-robin*:
    a) escreva o pseudo-código do procedimento de tratamento da interrupção do relógio.
    b) escreva o pseudo-código do procedimento que seleciona um processo.

30) Dois objetivos comuns das políticas de escalonamento são minimizar tempos de resposta e maximizar utilização de recursos.
    a) Indique como esses objetivos se contrapõem um ao outro.
    b) Analise as políticas de escalonamento circular (*round robin*) e múltiplas filas com retorno/realimentação (*multiple queues with feedback*) com base nessas duas perspectivas.

31) Suponha um S.O. com escalonador de filas multiníveis na qual há cinco níveis. O quantum do primeiro nível é 0,5 segundos. Cada nível mais baixo tem um quantum de tamanho duas vezes maior que o quantum do nível anterior. Um processo não pode sofrer preempção até o seu quantum terminar. O sistema executa processos em lote (*cpu bound*) e interativos (*I/O bound*).
    c) Por que esse sistema é deficiente?
    d) Quais mudanças mínimas você proporia para tornar o esquema mais aceitável para o mix de processos que pretende?