Exame Preliminar de Doutorado em Sistemas Operacionais

Duração: 4 horas

7 de março de 1997

Atenção: Apesar da soma dos pontos das questões ser 14, tente responder corretamente o número máximo de questões, a fim de maximizar as suas chances de uma boa avaliação. A nota é o máximo das somas das notas de questões, tomado sobre os subconjuntos de questões cujo valor soma 10. Seja claro e preciso em suas respostas. Se quiser, use figuras para ilustrar suas respostas.
Boa sorte!

Questão 1 *(1 Ponto)*

Explique qual é o problema de se implementar em UNIX um processo que faz um append em um arquivo usando o seguinte trecho de código. E qual seria a solução para evitar este problema?

```
if ( (fd = open (``filename´´,O_WRONLY)) < 0) err_sys(...);
...
if ( (lseek(fd,0,SEEK_END) < 0)) err_sys(``lseek error´´);
if ( (write(fd,buffer,100) != 100)) err_sys(``write error´´);
```

Questão 2 *(1 Ponto)*

Dada a seguinte tabela de tarefas tipo "batch" com os respectivos tempos de chegada à CPU e tempos de execução necessários calcule o tempo médio de espera[1] para este conjunto para cada uma das seguintes políticas de escalonamento:

- First Come First Serve (FCFS)

[1] O tempo de espera de uma tarefa pode ser calculado como a diferença entre os instantes de término da tarefa com e sem competição de outras tarefas pela CPU.

- Shortest-Job-First não premptivo
- Shortest-Job-First premptivo

Tarefa	Chegada	Duração
T1	0	7
T2	2	5
T3	3	3
T4	5	1

Questão 3 *(1 Ponto)*

Descreva pelo menos duas formas diferentes de representar um arquivo em disco.

Questão 4 *(1 Ponto)*

Em um máquina onde apenas hardware para paginação está disponível, é possível implementar segmentação apenas por software? E o reverso, ou seja, é possível implementar paginação via apenas software quando hardware para segmentação está disponível? Justifique suas respostas.

Questão 5 *(2 Pontos)*

A função `signal` no Unix instala um *handler* para tratamento de um sinal. Da página de manual:

```
SYNOPSIS
  #include <signal.h>

  void (*signal(sig, func))()
  void (*func)();

BSD DESCRIPTION
  ...
  A return from the function unblocks the handled signal
  and continues the process at the point it was interrupted.
  Unlike previous signal facilities, the handler func
  remains installed after a signal has been delivered.
  ...

SYSTEM V DESCRIPTION
  ...
  A return from the function continues the process at the
  point at which it was interrupted.  The handler func does
  not remain installed after a signal has been delivered.
  ...
```

Note a diferença entre o comportamento no BSD e SysV. Explique as conseqüências disso para efeito de programação.

Questão 6 *(2 Pontos)*

Como é possível criar e apagar, em Unix, um arquivo cujo **nome** contenha o caractere “/”? Note que se está falando do nome, não do pathname. Por exemplo um arquivo chamado `exame/de/so` no diretório `/var,` sem que exista um diretório `/var/exame`.

Questão 7 *(2 Pontos)*

O *double-fork* é uma técnica usada em Unix para clonar processos sem um terminal controlador (em geral daemons). Consiste em:

1. O processo original gera um filho, com `fork`.
2. O filho gera um neto (o segundo `fork`), e termina.
3. O neto sobrevive, junto com o original.

Escreva o pseudo-código para um programa que possa fazer um double-fork.

Questão 8 *(2 Pontos)*

Suponha que, ao invés de implementar semáforos com uma fila FIFO, usa-se “last come, first served” (pilha).

(a) Este semáforo pode ser usado para implementar exclusão mútua “segura” com n processos cíclicos? (“segura” significa que nenhum processo fica bloqueado indefinidamente enquanto outros continuam executando).

(b) Considere uma árvore binária com n folhas, onde cada folha representa um processo. Cada nó interno está associado a um semáforo. Os processos que desejam entrar na região crítica executam operações “P” nos semáforos, sucessivamente da folha até a raiz da árvore. Após executarem a região crítica os processos executam operações “V” na ordem reversa.

   (a) Mostre que este esquema implementa exclusão mútua segura

   (b) Discuta como o formato da árvore binária pode afetar a velocidade relativa dos processos e resultar em um esquema de prioridade.

Questão 9 ***(2 Pontos)***

Considere o seguinte estado de alocação de recursos em um sistema:

Processo	*Alocação*				*Max*				*Disponível*			
	A	B	C	D	A	B	C	D	A	B	C	D
p_0	0	0	1	2	0	0	1	2	1	5	2	0
p_1	1	0	0	0	1	7	5	0				
p_2	1	3	5	4	2	3	5	6				
p_3	0	6	3	2	0	6	5	2				
p_4	0	0	1	4	0	6	5	6				

Considere as seguintes questões usando o algoritmo do banqueiro ("banker's" algorithm, de autoria de Dijkstra) para evitar deadlocks:

(a) O sistema está em um estado seguro? Justifique.

(b) Se um pedido de $(0, 4, 2, 0)$ chega do processo p_1, pode ser atendido imediatamente? Justifique.