UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA

SISTEMAS OPERACIONAIS IN - Prova 1 - Turma B

Muita atenção na hora de responder as questões, e seja objetivo. A prova **não** contém "pega-ratões", então em caso de dúvida (sobre as questões, não as respostas!), pergunte. Sim, pode ser a lápis. Boa sorte!

Questão 1 (1.0 pontos)

Sobre as funções básicas de um sistema operacional responda:

(a) Um sistema operacional pode ser visto como um **alocador de recursos**. Sob esta visão, qual o papel do sistema operacional?

(b) Um sistema operacional pode ser visto como um **programa de controle**. Sob esta visão, qual o papel do sistema operacional?

Resposta:

(a) O SO distribui os recursos computacionais disponíveis (como CPU, memória, periféricos) entre os processos que estão rodando, de maneira a aproveitar bem estes recursos e provêr a cada processo aqueles recursos que ele necessita.

(b) "protege" um usuário (processo) do outro: evita que um processo acesse memória de outro sem permissão, evita que usuários leiam arquivos aos quais não tem permissão, evita que processos dominem a CPU. Garante que a política do sistema seja obedecida.

Questão 2 (1.5 pontos)

Sobre multiprogramação e *time-sharing*, responda:

(a) Qual o objetivo da multiprogramação?

(b) Caracterize processos *I/O bound* e *CPU bound*.

Resposta:

*(a) Aproveitar da melhor maneira possível os recursos computacionais disponíveis. **Não é** rodar vários processos ao mesmo tempo, isso é a forma com a qual o objetivo é atingido. **Não é** prover a ilusão de paralelismo, esse é o objetivo do time-sharing.*

(b) IO Bound x CPU Bound

- *IO: Passam a maior parte do tempo realizando entreda e saída, ou seja, aguardando a resposta de algum dispositivo de I/O*
- *CPU: Passam a maior parte do tempo (comparado com os IO-Bound!) realizando operações na CPU*

Questão 3 (1.0 pontos)

Considere o pseudo-código abaixo.

```
1  int saldo = get_saldo();
2  int retirada = get_valor_pedido();
3  if(retirada > saldo) {
4     printf("Seu pobretao!\n")
5  }
6  else {
7     set_saldo(saldo - retirada);
8     emitir_dinheiro(retirada);
9  }
```

Processo	Tempo do *Burst*	Prioridade
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

Tabela 1: Processos

O professor possui R$ 300 na conta, e está sacando 5 reais no caixa automático para pagar o almoço do RU. Ao mesmo tempo, sua respectiva, que está no Rio de Janeiro com as amigas, está tentando sacar 290 para comprar um lindo vestido em promoção.

(a) Considerando que as duas ações estão ocorrendo aproximadamente ao mesmo tempo, e que existe um *lag* (atraso) na propagação das informações via rede, quais os possíveis valores do saldo do professor após as 2 operações realizadas?

Resposta:

(a) Valores possíveis:

- *5*
- *10*
- *295*

Questão 4 (1.5 pontos)

Sobre processos e threads, responda.

(a) Cite e explica duas vantagens de um modelo de threads n:1 (threads implementadas em espaço de usuário) em relação ao modelo n:n (threads implementadas pelo kernel).

(b) Cite e explica duas vantagens de um modelo de threads n:n (threads implementadas pelo kernel) em relação ao modelo n:1 (threads implementadas em espaço de usuário).

Resposta:

(a) n:1

- *Troca de contexto muito mais rápida*
- *Portabilidade*
- *Escalonamento pode ser customizado para a aplicação*

(b) n:n

- *Se uma thread bloqueia esperando I/O, outra thread pode executar*
- *Paralelismo real em máquinas multiprocessadas*

Ambos os modelos de threads compartilham a memória (variáveis globais, heap e texto), logo não é vantagem nem desvantagem de qualquer dos modelos este compartilhamento.

Questão 5 (2.5 pontos)

Considere os processes da tabela 1. Assuma que os processos chegaram ao escalonador no tempo 0, na ordem P_1, P_2, P_3, P_4, P_5.

(a) Desenhe os gráficos de Gantt que ilustram a execução destes processos com os seguintes algoritmos de escalonamento: FCFS (First-Come-First-Served), SJF (Shortest-Job-First), por prioridades não preemptivo e RR (Round-Robin) com quantum = 1.

Processo	FCFS	RR	SJF	Prio
P_1	0	9	9	6
P_2	10	1	0	0
P_3	11	5	2	16
P_4	13	3	1	18
P_5	14	9	4	1

Tabela 2: Tempo de espera

(b) Preencha a tabela 2 com os tempos de espera de cada processo.

(c) Qual dos algoritmos obteve o menor tempo de espera médio (considerando todos os processos)?

Resposta:

(a) ...

(b) Na tabela

(c) SJF (não precisa nem fazer conta)

Tempo de espera é o tempo que o processo passa na fila de aptos, sem estar executando. No round robin, o processo é interrompido antes de terminar e VOLTA pra fila de aptos, logo deve-se somar esses tempos para obter o tempo de espera de um processo.

Questão 6 (1.5 pontos)

Descreva a arquitetura de micro-kernel: como é, quais as motivações por trás desta idéia, vantagens e desvantagens.

Resposta:

- *Arquitetura: apenas o mínimo necessário de código roda em modo privilegiado, a principal função de um microkernel é prover* ***comunicação*** *entre processos. Funcionalidades como escalonamento, memória virtual, sistemas de arquivos, drivers, são implementados como processos.*
- *Benefícios: mais segura, pois mais difícil de um bug em um dos serviços do SO afetar outro, design simplificado (baixo acomplamento), fácil de adicionar novo serviço ao kernel, estabilidade.*
- *A principal desvantagem da arquitetura em micro-kernel é o custo extra de estar constantemente utilizando mensagens para a realização de operações fundamentais do SO.*

Questão 7 (1.0 pontos)

Considere a alocação de memória contígua para cada processo.

(a) Quais as estratégias para alocar um novo processo em uma lacuna, e qual a justificativa para cada uma delas?

(b) Considere 5 partições livres (lacunas) de 100 KB, 500 KB, 200 KB, 300 KB, e 600 KB (em ordem). Como cada uma das estratégias irá alocar processos de 212 KB, 417 KB, 112 KB, e 426 KB (em ordem)?

Resposta:

(a) Estratégias

- *First-fit: aloca a memória na primeira lacuna de tamanho suficiente encontrada. Justificativa: evita percorrer toda a lista de lacunas.*
- *Best-fit: aloca a memória na MENOR lacuna de tamanho suficiente. Justificativa: Deixar a menor lacuna possível disponível (evitar fragmentação), e manter as lacunas grandes para um eventual processo grande.*

- *Worst-fit: aloca a memória na MAIOR lacuna de tamanho suficiente. Justificativa: Deixar que a nova lacuna seja a maior possível, maximizando a chance de um futuro programa poder ser alocado ali. Evita deixar lacunas muito pequenas que nunca possam ser preenchidas.*

(b) Alocação:

- *First-fit*
  1. *212K ⇒ partição 500K (sobra: 288K)*
  2. *417K ⇒ partição 600K*
  3. *112K ⇒ partição 288K (sobra da de 500k)*
  4. *426K não pode ser alocada*
- *Best-fit*
  1. *212K ⇒ partição 300K*
  2. *417K ⇒ partição 500K*
  3. *112K ⇒ partição 200K*
  4. *426K ⇒ partição 600K*
- *Worst-fit*
  1. *212K ⇒ partição 600K (sobra: 388k)*
  2. *417K ⇒ partição 500K*
  3. *112K ⇒ partição 388K (sobra da de 600k)*
  4. *426K não pode ser alocada*