# MC504/MC514 —Sistemas Operacionais

Profa. Islene Calciolari Garcia

Prova 3

16 de junho de 2016

| Questão | Nota |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| Total | |

Nome:

RA:

Você pode fazer a prova a lápis e utilizar o verso das folhas para completar suas respostas. Não é permitida consulta a qualquer material manuscrito, impresso ou em formato eletrônico. Em caso de fraude, todos os envolvidos receberão nota zero. Boa prova!

1. (1.0) Qual é a principal diferença entre modo usuário e modo kernel? Quando um processo ou thread deixa de executar em modo usuário para executar em modo kernel?

2. (2.0) O algoritmo de escalonamento Round Robin utiliza uma fila circular simples de processos prontos para serem executados. Muitas vezes um processo é interrompido antes de sua fatia de tempo (quantum) acabar devido à execução de uma operação de I/O. Neste caso, quando a operação de I/O termina, o processo é colocado ao final da fila e perde o tempo restante de uso de CPU a que tinha direito na passada anterior. Para compensar esta injustiça, um desenvolvedor pensou em três variações simples descritas abaixo. Indique problemas e/ou desvantagens de cada uma das propostas apresentadas:

   (a) Um processo que não usou sua fatia de tempo será colocado no começo da fila, mas só poderá rodar pelo tempo que tinha sobrado na passada anterior.

   (b) Um processo que usou menos da metade de sua fatia de tempo será colocado no meio da fila (e não ao final dela). Este processo terá direito de rodar por um quantum, mas será favorecido porque não precisará esperar por todos os processos na fila.

   (c) Processos terão direito de acumular tempo de CPU. Assim, um processo que usou menos do que sua fatia de tempo inteira será colocado ao final da fila, mas terá direito de rodar por um quantum mais o tempo que ficou faltando nas passadas anteriores.

3. (1.5) Observe o programa abaixo.

```
#define PAGESIZE 4096
#define N 500000

char mat[N][PAGESIZE];

void funcao_A(char c) {
  int i, j;
  for (i = 0; i < N; i++)
    for (j = 0; j < PAGESIZE; j++)
      mat[i][j] = c;
}
void funcao_B(char c) {
  int i,j;
  for (j = 0; j < PAGESIZE; j++)
    for (i = 0; i < N; i++)
      mat[i][j] = c;
}

int main() {
  funcao_A('*');
  funcao_B('-');
  return 0;
}
```

Considere que o seu sistema utiliza paginação e que endereço mat[i][j] é calculado da forma abaixo:

```
mat[i][j] = mat + (i*PAGESIZE + j) * sizeof(char)
```

(a) Qual função (funcao_A ou funcao_B) irá, provavelmente, ser executada mais rápido?

(b) Descreva duas razões relacionadas ao gerenciamento de memória que justificam a sua resposta acima.

4. (1.0) No sistema Linux, as funções `get_user` e `put_user` executam a transferência de valores de variáveis simples entre o espaço de usuário e o de kernel. A documentação destas funções deve conter o alerta: "`This function may sleep.`"? Justifique sua resposta.

5. (1.0) Descreva uma proteção implementada pelos sistemas modernos para impedir ataque do tipo *buffer overflow*.

6. (1.5) Descreva três exemplos de inconsistências em sistemas de arquivos corrompidos do tipo ext2.

7. (2.0) Considere um sistema de arquivos que permite o estabelecimento de links simbólicos. Proponha um algoritmo para que um comando como

   ```
   $ cat link-simbolico-para-arquivo.txt
   ```

   não faça o sistema entrar em looping em caso de referências circulares. Indique eventuais limitações ou pontos negativos da sua abordagem.