MC504/MC514 —Sistemas Operacionais:

Profa. Islene Calciolari Garcia

Prova 2

13 de novembro de 2013

Questão	Nota
1	
2	
3	
4	
5	
6	
7	
Total	

Nome:

RA:

Instruções: Você pode fazer a prova a lápis (desde que o resultado final seja legível :-). Não é permitida consulta a qualquer material manuscrito ou impresso. Em caso de fraude, todos os envolvidos receberão nota zero.

1. (1.0) Qual é a influência do tamanho da fatia de tempo (quantum) no bom funcionamento de um algoritmo de escalonamento?

2. (1.5) O algoritmo de escalonamento Round Robin implementa uma fila circular simples de processos prontos para serem executados. Muitas vezes um processo é interrompido antes de sua fatia de tempo 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 desenvolver pensou em duas variações simples:

   - 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.
   - 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.

   Qual das duas abordagens você sugere ao desenvolvedor? Justifique sua escolha, explicando porque a abordagem escolhida é melhor do que a outra. Você também pode sugerir outras melhorias ao desenvolvedor, desde que justifique porque sua abordagem é melhor do que (a) e (b).

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 (j = 0; j < PAGESIZE; j++)
    for (i = 0; i < N; i++)
      mat[i][j] = c;
}
void funcao_B(char c) {
  int i, j;
  for (i = 0; i < N; i++)
    for (j = 0; j < PAGESIZE; j++)
      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. (1.5) Analise o código abaixo. Considere um sistema baseado em paginação e descreva em quais condições poderá acontecer um erro de execução.

```
#include <stdio.h>
void f() {
  int i;
  int* v = malloc (10);
  for (i = 0; i < 100; i++)
     v[i] = i;
}

int main() {
  f();
  return 0;
}
```

5. (1.5) Explique como funciona o ataque denominado *buffer overflow*. Quais são as proteções implementadas pelos sistemas modernos para impedir este ataque?

6. (1.0) Em um sistema Unix, é possível criar um link simbólico para um arquivo que não existe? E um hard link? Justifique sua resposta.

7. (2.0) Considere um sistema de arquivos ext2.

   (a) Após uma queda de energia, é possível que dois i-nodes distintos estejam apontando para um mesmo bloco de dados? Em caso afirmativo, apresente um cenário que leva a esta configuração. Caso contrário, apresente argumentos que justifiquem a sua resposta.

   (b) O que um programa tipo fsck poderia fazer para solucionar este problema? É possível recuperar o conteúdo dos dois arquivos?

Boa prova!