BCC34G – Sistemas Operacionais
Lista de Exercícios #03

1. O que são *threads*? Quais as vantagens e benefícios do uso de *threads* em relação ao uso de processos?
2. Compare *threads* do usuário (*user level threads*) e do núcleo (*kernel level threads*).
3. Explique os modelos de implementação de *multithreading*: um-para-um, muitos-para-um e muitos-para-muitos. [1]
4. O que é o *Pthreads*? [1]
5. Explique o que é um *pool de threads*. [1]
6. Explique e compare as duas formas para cancelamento de *threads* (assíncrono e adiado). [1]
7. Cite três modos pelos quais um thread pode terminar. [2]
8. Por que se deve permitir que um thread desabilite seu sinal de cancelamento? [2]
9. Qual dos seguintes componentes de estado de um programa são compartilhados pelas threads em um processo com múltiplas threads? (a) valores do registrador (b) memória do heap (c) variáveis globais (d) memória da pilha. [1]
10. Forneça dois exemplos de programação em que a criação de múltiplos *threads* proporciona melhor desempenho do que uma solução com um único *thread*. [1]
11. Uma solução com múltiplos *threads* usando múltiplos *threads* de nível de usuário pode obter melhor desempenho em um sistema multiprocessador do que em um sistema com único processador? Explique. [1]
12. Um sistema com dois processadores dual-core tem quatro processadores disponíveis para escalonamento. Uma aplicação CPU-intensiva está sendo executada nesse sistema. Todas as entradas são fornecidas na inicialização do programa, quando um único arquivo deve ser aberto. Da mesma forma, todas as saídas são fornecidas logo antes de o programa terminar, quando os resultados do programa devem ser gravados em um único arquivo. Entre a inicialização e o encerramento, o programa é totalmente limitado pela CPU. Sua missão é melhorar o desempenho dessa aplicação tornando-a *multithreaded*. A aplicação é executada em um sistema que usa o modelo um-para-um de criação de threads (cada thread de usuário é mapeado para um thread do kernel). [1]
    a) Quantos threads você criará para executar a entrada e a saída? Explique.
    b) Quantos threads você criará para a parte da aplicação que é CPU-intensiva? Explique.
13. Considere o segmento de código abaixo e responda: [1]
    a) Quantos processos únicos são criados?
    b) Quantos threads únicos são criados?

```
  pid_t pid;
pid = fork();
  if (pid == 0) {/*processo-filho */
   fork();
   thread_create( … );
  }
  fork();
```

14. Por que threads do mesmo processo em geral se comunicam mais eficientemente do que em processos separados? [2]
15. Presuma que você esteja tentando baixar um arquivo grande de 2 GB da internet. O arquivo está disponível a partir de um conjunto de servidores espelho, cada um deles capaz de fornecer um subconjunto dos bytes do arquivo; presuma que uma determinada solicitação especifique os bytes de início e fim do arquivo. Explique como você poderia usar os threads para melhorar o tempo de download. [3]

Referências:

[1] SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Fundamentos de Sistemas Operacionais.** 9. ed. LTC, 2015. págs. 167-171.

[2] DEITEL, H.; DEITEL, P. J.; CHOFFNES, D. R. **Sistemas Operacionais.** 3. ed. São Paulo: Pearson, 2005.

[3] TANENBAUM, A. S.; BOS, H.. **Sistemas Operacionais Modernos.** 4a ed. Pearson, 2016.