**Tarefa 1 (Individual): Questionário 1.2 – Entregar no Moodle até 19/03/2012.**

**Nome: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1.1 Explique, resumidamente, o que é um processo em um Sistema Operacional.**

Todos os softwares que podem executar em um computador, inclusive o SO (os mais tradicionais são assim), são organizados para serem executados num processador, como vários processos sequenciais (também chamados processos). Um **processo** é uma atividade (ou tarefa) de um programa, que contém o código e dados de uma atividade. Essas são: leitura de dados, escrita de dados, cálculos no processador, comunicação com o usuário, comunicação com um BD, comunicação com a rede interna ou externa, entre outras. Um processo define **a unidade de processamento concorrente**, que é executada num dado instante num processador, utilizando um contador de programa lógico, usando o único contador de programa físico (registro no processador), valores em registradores, variáveis do programa e uma pilha de execução. Processos são escalonados para o processador, que faz uma troca a todo momento do processo sendo executado, através do mecanismo chamado **multiprogramação**.

**1.2 Explique a diferença entre programa e processo.**

A diferença entre um processo e um programa é sutíl, mas crucial. **Um programa é um algoritmo expresso por uma linguagem adequada ao computador que contém atividades que devem ser executadas** e são chamadas de processos. Um programa (um software) corresponde a um conjunto de processos. A idéia principal é que um processo constitui uma atividade (tarefa) e que tem um espaço de endereçamento. O modelo de processo é baseado em dois conceitos independentes: agrupamento de recursos e fluxo de execução. Por recursos entende-se arquivos abertos, processos filhos, alarmes pendentes, tratadores de eventos, sinais no kernel, informação sobre contabilidade de execução. O processo deve facilitar o gerenciamento desses recursos.

**1.3 Thread** e **processo** são conceitos diferentes. Explique sucintamente, a diferença entre o conceito de processo e o conceito de thread. O que as threads acrescentam ao modelo de processo ?

Em SO tradicionais, cada processo tem um único **fluxo de execução** (o que define uma **thread**), a unidade de processamento concorrente destinada para ser executada sob as condições de desempenho de um processador da época. Com o surgimento de processadores de mais alto desempenho, uma nova unidade de processamento concorrente pôde ser definida dentro do próprio processo, materializando novas unidades de fluxo de execução e assim pode-se ter múltiplos fluxos de execução (múltiplas threads) num mesmo processo.

O que as threads acrescentam ao modelo de processo é permitir que múltiplos fluxos de execução ocorram no mesmo ambiente do processo, com um grau de independência uma das outras. Assim, **múltiplas threads executam concorrentemente em um processo**, e é análogo a **múltiplos processos executando concorrentemente em um único computador**.

No primeiro caso, threads compartilham o mesmo espaço de endereçamento e recursos do processo onde são executadas e o termo *multhreading* é usado para descrever a situação em que múltiplas threads são executadas no mesmo processo. Quando um processo com múltiplas threads é executado em um SO com um único processador, as threads são escalonadas para execução, alternando rapidamente entre as threads, dando a ilusão que são executadas em paralelo num processador mais lento que o processador real.

**1.4** (**Verdade** ou **Falso**) **Threads distintas em um processo não são tão independentes quanto processos distintos.** Explique, resumidamente.

Verdade. Todas as threads tem exatamente o mesmo espaço de endereçamento, o que significa que elas compartilham as mesmas variáveis globais do processo. Além de compartilharem o mesmo espaço de endereçamento, todas as threads compartilham o mesmo conjunto de recursos do processo (arquivos abertos, processos filhos, alarmes pendentes, tratadores de eventos, informação sobre contabilidade de execução, entre outros).

**1.5** (**Verdade** ou **Falso**) **Não há proteção entre threads porque é impossível e desnecessário.**

Falso. No caso de processos diversos, que podem ser de usuários diferentes e até mutuamente hostis, um processo sendo criado por um usuário, presume-se que este tenha criado múltiplas threads no processo para que essas possam cooperar e não competir. Do ponto de vista de threads em processos diferentes, e ainda mais de esses processos forem de usuários diferentes, proteção entre threads é importante.

**1.6** (**Verdade** ou **Falso**) Itens propriedade de processos são: Espaço de endereçamento, Variáveis globais, Contador de programa lógico, Registradores, Pilha, Estado e Recursos.

Verdade. Os itens refletem o que é um processo.

**1.7** (**Verdade** ou **Falso**) Itens propriedade de threads são: Variáveis globais, Contador de programa lógico, Registradores, Pilha, Estado.

Verdade. Compare com 1.6 e veja que espaço de endereçamento e recursos pertencem aos processos.

**1.8** (**Verdade** ou **Falso**) Thread é uma unidade de gerenciamento de seus recursos.   
  
Falso. Quem gerencia recursos é o processo.

**1.9** (**Verdade** ou **Falso**) O esquema de escalonamento (*scheduling*) fundamental para threads é preemptivo (força uma thread parar sua execução) e baseado em prioridade. O *scheduler* (escalonador), neste caso, é um algoritmo não baseado em fracionamento de tempo (*time-slicing* é preemptivo, mas preempção não implica em *time-slicing*), que permite threads de mais alta prioridade executarem tanto tempo quanto elas necessitem.

Verdade. Ou o escalonamento é preemptivo por fracionamento de tempo para threads de igual prioridade, ou escalonamento é preemptivo baseado em prioridade estabelecida para threads.

**1.10** Dê exemplo de uma aplicação (pode ser que processem uma quantidade grande de dados) que é melhor construída considerando-se múltiplas threads compartilhando o mesmo espaço de endereçamento ou espaços distintos. Descreva a estrutura dessa aplicação e dizendo quais as finalidades de suas threads.