

**Caps.3 e 4 – Processos e Threads**

1. O que são Processos? Como são classificados? Qual a diferença entre um Processo e um Programa?

**R-** Processos são programas em execução. São classificados em Processos de Kernel e Processos de Usuário. Um Processo é uma unidade *ativa*, enquanto um Programa é uma unidade *passiva*.

1. Quais são os possíveis estados de um Processo? Explique sucintamente como o processo se encontra em cada estado.

**R-** Os possíveis estados de um Processo são:

NEW: Quando o Processo está sendo criado, alocando recursos;

READY: O Processo está esperando ser atribuído a um processador;

RUNNING: Instruções estão sendo executadas;

WAITING: O Processo está esperando que algum evento ocorra (EX: esperando a finalização de uma operação de IO);

TERMINATED: O Processo terminou sua execução;

1. O que são Filas de Scheduling? Como são subdivididas? Explique.

**R-** São filas que tem como finalidade organizar o escalonamento de processos. Geralmente, todos os SO’s possuem 3 tipos de filas. São elas:

FILA DE JOBS (job queue): Composta por todos os programas do sistema que aguardam para serem alocados na RAM;

FILA DE PRONTOS (ready queue): Processos que estão residindo na memória principal e estão prontos esperando execução;

FILA E DISPOSITIVO (device queue ou i/o queue): Lista de processos em espera por um dispositivo de I/O. Cada dispositivo tem sua própria fila de dispositivos;

1. O que você entende por Context Switching? E Overhead?

**R-** Context Switching (troca/mudança de contexto) é o processo computacional de armazenar e restaurar o estado (contexto) de uma CPU de forma que múltiplos processos possam compartilhar uma única instância de CPU. O intervalo de mudança de contexto é puro Overhead, ou seja, o sistema não executa trabalho útil algum durante a mudança.

1. Como o Sistema Operacional consegue gerenciar cada processo em execução de forma adequada? Que tipo de estrutura é mantida por ele a fim de prover esta organização?

**R-** A partir do momento em que vários processos estão no estado de pronto, critérios são estabelecidos para determinar qual processo será escolhido para fazer uso do processador. Os critérios usados para a escolha compõem a chamada política de escalonamento, que é a base da gerência do processador e da multiprogramação em um SO.

Dentro da gerência de processos também estão várias estruturas de um processo que ajudam o SO a gerenciá-los, como por exemplo, o Bloco de Controle de Processo. O PCB de um processo é como se fosse um repositório que guarda/gerencia todas as informações deste processo para o SO.

1. Como é estruturado um Processo na memória Principal? Explique em detalhes.

**R-** Na memória principal, um Processo é formado pelas seguintes partes:

TEXT SECTION: contém o código do programa e suas constantes. Este segmento é alocado durante a criação do processo (''exec'') e permanece do mesmo tamanho durante toda a vida do processo.

DATA SECTION: este segmento é a memória de trabalho do processo, aonde ficam alocadas as variáveis globais e estáticas. Tem tamanho fixo ao longo da execução do processo.

STACK: contém a pilha de execução, onde são armazenados os parâmetros, endereços de retorno e variáveis locais de funções. Pode variar de tamanho durante a execução do processo.

HEAP: contém blocos de memória alocadas dinamicamente, a pedido do processo, durante sua execução. Varia de tamanho durante a vida do processo.

1. O que são processos filhos? Por quais razões um processo pai pode finalizar um processo filho?

**R-** Processos filhos são processos criados. O processo *pai* cria processos *filhos*, que podem criar mais processos, formando assim uma hierarquia de processos, porém nela apenas existirá um processo pai e ter ou não um ou mais processos filhos.

Um pai pode encerrar a execução de um de seus filhos por várias razoes, são elas:

- O filho excedeu o uso de alguns dos recursos que recebeu;

- A tarefa atribuída ao filho não é mais necessária;

- O pai está sendo encerrado e o SO não permite que um filho continue executando (Encerramento em Cascata);

1. O que você entende por Sockets? Quando são utilizados?

**R-** Sockets são representados como descritores de arquivos e permitem a comunicação entre processos distintos na mesma máquina ou em máquinas distintas, através de uma rede.

São utilizados na comunicação em sistemas cliente-servidor. Diversas aplicações que utilizamos no dia-a-dia fazem uso de sockets pra se comunicar, como por exemplo, o nosso navegador web que utiliza sockets pra requisitar páginas.

1. Diferentes processos podem se comunicar entre si? Em caso afirmativo, quais são as técnicas utilizadas para prover esta comunicação e em que situações devem ser utilizadas?

**R-** Sim. Existem dois modelos básicos de IPC (Inter-Process Communication), são eles:

**Transmissão de Mensagens:** ocorre por meio de troca de mensagens entre os processos. É utilizada para troca de pequenas quantidades de dados e ideal para troca de dados entre processos remotos.

**Memória Compartilhada:** uma região de memória é compartilhada entre os processos. É utilizada para troca de grandes volumes de dados entre processos locais.

1. O que você entende por Threads? Cite três grandes benefícios desta tecnologia.

**R-** Thread é a tarefa que um determinado programa realiza.

É uma forma de um processo se autodividir em duas ou mais tarefas que podem ser executadas concorrencialmente.

**Escalabilidade:** Um processo com um único Thread só pode ser executado em um processador. O uso de várias Threads em uma máquina com muitas CPU’s aumenta o paralelismo;

**Capacidade de respostas:** O uso de várias Threads em uma aplicação pode permitir que um programa continue a ser executado mesmo se parte dele estiver bloqueada ou executando uma operação demorada;

**Compartilhamento de recursos:** Os processos só podem compartilhar recursos através de técnicas como a memória compartilhada ou transmissão de mensagens. No entanto, por default, os Threads compartilham a memória e os recursos do processo ao qual pertencem;

1. Quais informações são específicas de uma Thread? E quais informações de um Processo são compartilhadas entre todas as suas Threads?

**R-** Informações específicas: Um Thread ID, seu próprio PC, seus próprios conjuntos de registradores, sua própria stack e sua própria heap;

Informações compartilhadas: Seu text section, o data section e arquivos abertos;

1. Em um Context Switching, o que é mais fácil, realizar a troca de contexto entre Threads ou a troca de contexto entre Processos? Explique.

**R-** É mais fácil realizar a troca de contexto entre Threads. Geralmente demora muito mais criar a gerenciar Processos do que criar e gerenciar Threads, pois as Threads compartilham os recursos do processo ao qual pertencem, logo é mais fácil e menos custoso alterar seus contextos.

1. Qual a diferença entre Threads de Kernel e Threads de Usuário?

**R- Threads de Kernel**

São suportadas e gerenciadas pelo SO;

São criados por chamadas a System Calls (clone());

O SO provê recursos para a criação, escalonamento, gerenciamento, remoção, etc. de Kernel threads

São “vistas” pelo Kernel;

Entram no escalonamento (process scheduling) do SO;

**Threads de Usuário**

São suportadas acima do Kernel e gerenciadas sem suporte dele;

São implementadas por uma thread library em nível de usuário;

A thread library possui rotinas para a criação, gerenciamento, remoção, etc. de user Threads;

Não são “vistas” pelo Kernel;

Não entram diretamente no escalonamento do SO;