Nome:	WESLEY MARCOS BORGES					Curso:	GEC
Matrícula:	1651	Período:	P8	Matéria:	C012	Turma:	

Cap. 3 e 4 – Processos e Threads

1) No mundo dos Sistemas Operacionais, sabemos que os programas ficam armazenados na memória secundária (HD/SSD). Quando o processador precisa usar algum desses programas, é solicitado à memória principal (RAM). O conceito de ***programa*** se refere à uma sequência de instruções em código fonte que executa uma série de tarefas. Quando o programa é introduzido à RAM, ele deixa de ser um programa (atua como passivo) e passa a se chamar ***processo*** (atua como ativo). Ou seja, processo é um programa em execução.

2) Ao iniciar um processo, ele terá estados que serão executados durante seu funcionamento. Eles são:

- **Novo (New) =** nesse estado o processo é criado, ou seja, o programa é executado.
- **Pronto (Ready) =** depois do processo ser criado no estado New e admitido, ele esperará ser atribuído ao processador.
- **Em execução (Running) =** nesse estado o processo é executado.
- **Em espera (Waiting) =** nesse estado o processo aguarda um novo evento no qual ele precisa para dar continuidade, como I/O, por exemplo.
- **Concluído (Terminated) =** o processo termina sua execução.

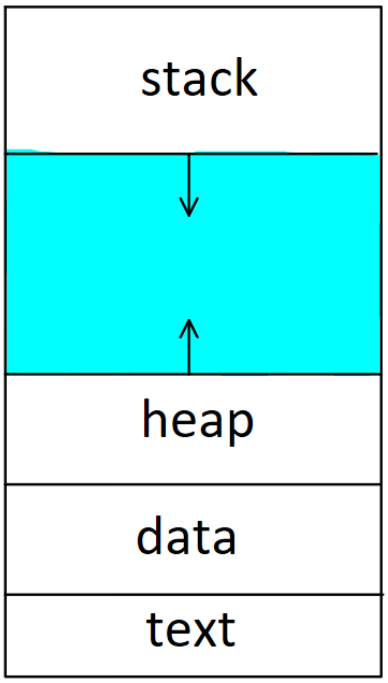
3) Filas de Scheduling são filas onde os processos aguardam para serem executados, de acordo com suas prioridades. As filas são subdivididas em:

- **Curto prazo =** seleciona entre os processos que estão prontos para execução e aloca CPU para um deles.
- **Médio prazo =** seleciona qual processo irá da memória para o Swapfile, ou retornará do Swapfile para a memória.
- **Longo prazo =** seleciona quais processos serão admitidos para serem executados. Escalona processos da fila de new para a fila de ready.

4) O ***Context Switching*** é o salvamento do estado do processo corrente e a restauração do estado de um processo diferente. Isso ocorre ao alocarmos CPU a outro processo. Ao fazermos essa mudança de contexto, um tempo é necessário para essa troca. Esse tempo que o sistema não faz nada (mas é essencial para a troca) é chamado de ***Overhead (idle).***

5) O gerenciamento por parte do SO é feito por um registro chamado PCB (Processo Control Block). No PCB é mantido todos os detalhes referentes ao processo. Para manter esses registro organizados, é usada uma estrutura de fila.

6) A estrutura de um processo na memória principal é feita da seguinte forma:



Stack = armazena dados temporários como variáveis locais, chamadas de funções, entre outras.

Heap = memória alocada durante a execução do processo, de forma dinâmica.

Data = armazena variáveis globais.

Text = código do programa.

7) Um processo filho é um processo criado por outro processo, onde chamamos o processo gerador de pai e o processo gerado de filho. O processo filho é criado, geralmente, para distribuir tarefas específicas, ajudando o processo gerador. O processo pai pode finalizar o processo filho quando o processo filho consome mais recursos do que o que foi alocado à ele, quando ele já finalizou sua tarefa ou quando o SO encerra o processo pai, derrubando o processo filho também.

8) Sockets são definidos como extremidades de comunicação, onde dois processos se comunicam em uma rede, de forma remota, através de um par (IP, PORTA). Essa combinação (IP, PORTA) é chamado de sockets.

9) Diferentes processos podem sim se comunicar, através de um mecanismo chamado IPC (Interprocess Comunication). Existem dois modelos básicos de IPCs, são eles:

Transmissão de mensagens = a comunicação é feita através de um link, tendo dois principais recursos padrão: send() e receive(). São utilizadas para ***baixa quantidade de dados***.

Memória Compartilhada = mais rápida que a transmissão de mensagens (exige menos System Calls), requer que dois ou mais processos concordem em eliminar a restrição do SO de não poder usar memória de outro processo. Quem compartilha a memória é o processo que faz a requisição de comunicação. São utilizadas para ***grande quantidade de dados***.

10) As threads são divisões criadas pelos processos para executas pequenas tarefas, dando a impressão de “simultaneidade”. Existem diversos benefícios para o uso das threads, como: maior capacidade de resposta, maior economia de dados, compartilhamento de recursos e escalabilidade (paralelismo).

11) As Threads possuem informações específicas como ID, PC, conjunto de registradores, stack (pilha) e heap. Existem algumas informações que são compartilhadas com outras Threads de um mesmo processo como Text Section (Code), Data Section e arquivos abertos.

12) É mais realizar um Context Switching entre Threads, pois a memória e recursos de processamento do processo são compartilhadas pelas Threads. Quando o Context Switching é feito entre processos, o idle é bem maior, fazendo com que a troca fique mais lenta.

13) As Threads de kernel são suportadas e gerenciadas pelo SO. Já as Threads de usuário são suportadas em uma camada acima do SO, implementadas por uma thread library.