Nome: Pedro Gabriel Garcia Ribeiro Balestra		Matrícula: 1551
Curso: GEC	Período: P8	Matéria: C012

Caps.3 e 4 – Processos e Threads

1. Podemos dizer que um processo é um programa que está em execução, ou seja, para que ocorra um processo é necessário que o usuário de o famoso dois clicks no programa que se encontra "parado", dessa forma, fazendo que o mesmo entre em execução. De maneira direita o processo é a unidade ativa, enquanto o programa é a unidade passiva. Podemos classificar o processo em 2 níveis, Processo de Usuário e Processo de Kernel

2. Existem 5 estados em que o processo pode ser encontrado:
   - **Novo (New):** Quando o processo é criado
   - **Em Execução (Running):** Instruções do processo estão sendo executadas
   - **Em Espera (Waiting):** Momento em que o processo espera que um evento ocorra. Ex: finalização de uma operação IO
   - **Pronto (Ready):** O processo está pronto para ser atribuído a um processador
   - **Concluído (Terminate):** Fim de execução do processo

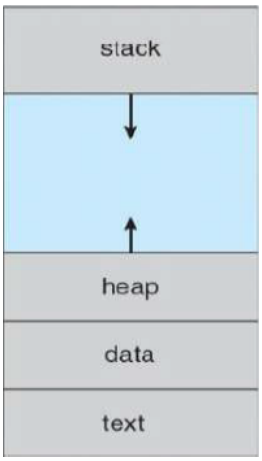
3. São filas onde os processos aguardam para poderem ser executados, conforme a prioridade. Sendo subdivididas:
   - **Curto prazo:** seleciona entre os processos que estão prontos para execução e aloca CPU para um deles.
   - **Médio prazo:** seleciona qual processo irá da memória para o Swapfile, ou 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. Context Switching em meu entendimento é o salvamento do estado de um processo atual e a restauração do estado de um processo diferente. Já overhead o é intervalo de tempo de um Context Switching onde o sistema não realiza um trabalho útil.

5. Para que o SO consiga gerenciar todos os processos em execução é adotado um método de arvore, onde tem um processo pai que cria o processo filho, onde ele pode criar vários outros processos respeitando o espaço de memória "herdado".

6. É estruturado da seguinte forma:

   Stack: Local onde se armazena os dados temporário. Ex: chamada de funções, entre outras.
   Heap: memória alocada durante a execução do processo, de forma dinâmica
   Data: Local de armazenamento de variáveis globais.
   Text: Código do programa.



7. São processos criados pelo processo pai, para “diminuir” a demanda de um único processo, podendo finalizar o processo filho em situações em que o filho excedeu o uso de recursos dado a ele, quando o pai é encerrado pelo SO e caso a tarefa do filho já tenha sido concluída

8. É o modo de identificação que um processo tem para se comunicar com outro processo, sendo composto por um par (IP,Porta), dessa forma o processo consegue identificar para qual o processo dever ser mandada a mensagem

9. Processos conversão entre si através de um mecanismo chamado IPC. Onde existe dois modelos.
   Transmissão de mensagens: Comunicação feita através de um link, usando os recursos send() e recive(). Usado para baixa quantidade de dados.
   Memória Compartilhada: Utilizada pra grandes quantidades de dados, sendo mais rápida que a transmissão de mensagens, pois utiliza menos System Calls. É necessário que dois ou mais processos “concordem” em eliminar a restrição do SO, para assim poder usar a memória de outro processo

10. São divisões feitas pelos processos em pequenas tarefas, dando a impressão de todos estão sendo executados simultaneamente. Tendo como benefício a capacidade de resposta, paralelismo, economia de dados.

11. ID, PC, heap e stack são informações específicas de uma Thread. Enquanto o Text Section, Data Section e arquivos abertos podem ser compartilhadas com outras Threads.

12. É mais fácil realizar troca de contexto entre Threads, pois a memória e recursos de processamento são compartilhadas por elas. Quando realizada troca entre processos, o uso de idle é maior, sendo assim mais lento.

13. As Threads de kernel são suportadas e gerenciadas pelo SO. Enquanto Threads de usuário são suportadas em uma camada acima do SO, implementadas por uma Thread library.