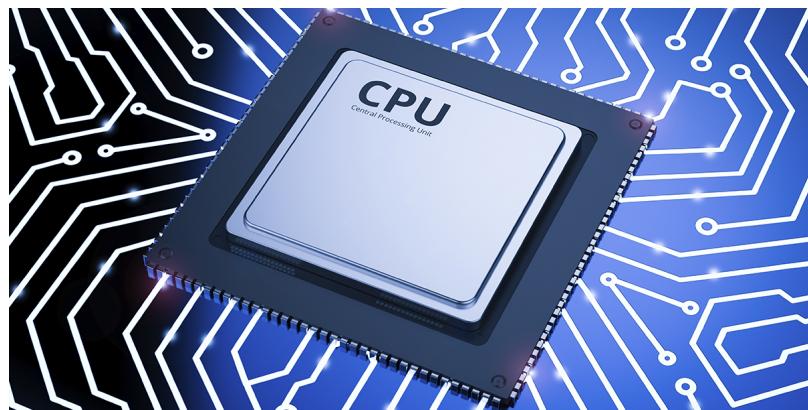


Sistemas operacionais

Aula 2 - Processos

INTRODUÇÃO



Inicialmente os Sistemas Operacionais (SO) eram monotarefa, ou seja, apenas uma tarefa (programa) era executada de cada vez.

Com a evolução dos sistemas computacionais, os SOs também evoluíram no sentido de dar suporte à execução de várias tarefas ao mesmo tempo, de forma real em sistemas com múltiplos processadores, ou de forma concorrente em sistemas com um único processador, por meio de multiprogramação.

Para que a execução de várias tarefas seja possível, é necessário o monitoramento das múltiplas atividades entre os vários programas, tarefa difícil e bastante complexa.

Para poder dar conta desta situação, os projetistas de Sistemas Operacionais desenvolveram o conceito de processo, que será objeto de estudo desta aula.

OBJETIVOS



Descrever as características de um processo;

Identificar os estados do processo;

Diferenciar aplicativos e serviços.

CONCEITO DE PROCESSO



Fonte da Imagem:

Talvez você se pergunte: Afinal, o que é um processo?

De forma simplificada, podemos definir processo como um programa em execução.

O conceito de processo é muito aplicado em [SO \(glossário\)](#). Com a introdução desse conceito foi possível isolar e especificar muitas das funções primitivas dos SOs.

Cada atividade de um SO consiste na execução de um ou mais programas que são chamados toda vez que a respectiva função é requerida. Usa-se a palavra *processo* para descrever uma atividade desse tipo. Assim, um processo pode ser considerado como sendo a sequência de ações realizadas pela execução de instruções (i.e. um programa) cujo resultado final fornece uma das funções do sistema.

Esse conceito pode ser estendido de forma que inclua as funções do usuário. Então, a execução de um programa de um usuário é também um processo.

Um processo pode envolver a execução de mais de um programa e, por outro lado, um mesmo programa pode ser requerido por mais de um processo. O conceito de processo é dinâmico; já o de programa é estático. Por exemplo, uma rotina que inclui um nó numa lista, pode ser usada por qualquer um dos processos encarregados pela manipulação da lista. O fato de sabermos que a rotina está ativa, em certo instante, não é suficiente para que se saiba qual das funções do sistema está sendo realizada.

Na realização das funções de um SO, algumas vezes é conveniente forçar o paralelismo de algumas atividades. As razões para o paralelismo são: **disponibilidade de agentes** (para levar a cabo a atividade) e **reduzir o tempo de execução da tarefa**.

Nem sempre é possível usar paralelismo, uma vez que o início de um processo pode estar condicionado à conclusão de outro (ou outros). Por exemplo:



ESTADOS DO PROCESSO

Ao longo de seu tempo de vida, um processo vai assumindo vários estados que representam sua situação atual. Quando um processo muda de estado dizemos que ele sofreu uma transição.

Veremos a seguir como os estados do processo funcionam nos dois tipos de sistema.

Sistemas Monotarefas

Neste tipo de sistema, cada programa é carregado do disco para a memória e executado até sua conclusão.

Os dados são carregados na memória juntamente com o código executável, e os resultados obtidos no processamento salvos no disco após a conclusão da tarefa.

A figura mostra os estados do processo neste contexto:



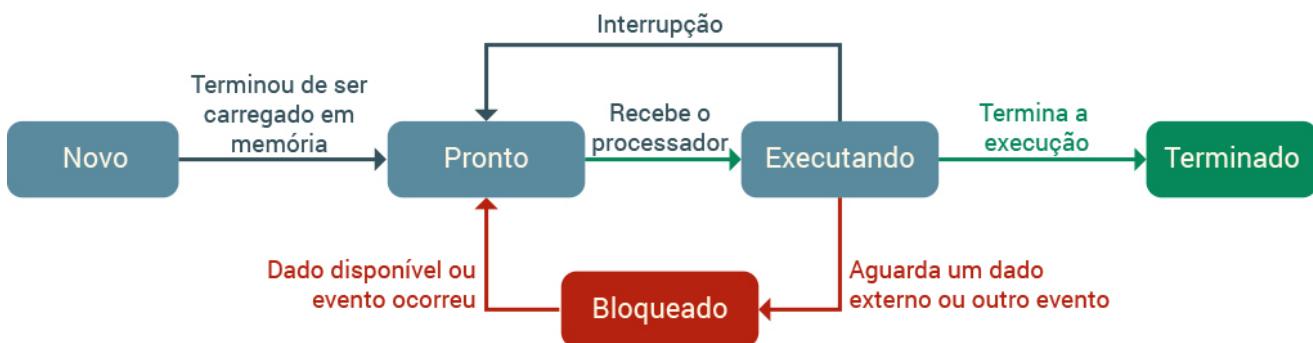
Onde:

Novo	Executando	Terminado
O processo está sendo criado, o código e os dados necessários para sua execução estão sendo carregados na memória.	As instruções do programa estão sendo executadas no processador.	O programa foi totalmente processado, produziu os resultados e pode ser removido da memória do sistema.

Sistemas Multitarefas

Neste tipo de sistema, vários programas podem ser executados ao **mesmo tempo (glossário)**.

O diagrama de transições mostra os diferentes estados possíveis de um processo, além do agente que causa a mudança de estado:



Novo

O processo está sendo criado, o código e os dados necessários para sua execução estão sendo carregados na memória, e as estruturas de dados do núcleo estão sendo atualizadas para permitir o controle de sua execução. Após o término de sua criação, ele passa ao estado de *pronto*.

Pronto

O processo é colocado em uma fila de prontos, ou seja, ele está preparado para executar (ou para continuar sua execução), apenas aguardando a disponibilidade do processador. O Algoritmo de Escalonamento escolhe um processo para “ganhar” a CPU, passando este, então, ao estado de *executando*.

Executando

As instruções do programa estão sendo executadas no processador. A partir deste estado, três transições podem ocorrer:

- O processo solicita uma operação de entrada/saída – o processo passa a *bloqueado*;
- O processo chegou ao seu final ou ocorreu um erro que o impede de prosseguir – passa ao estado de *terminado*;
- Ocorre uma interrupção – processo passa ao estado de *pronto*.

Bloqueado

O processo não pode ser executado porque depende de dados externos ainda não disponíveis (do disco ou da rede, por exemplo) necessitando de uma operação de entrada/saída ou aguarda algum tipo evento ocorrer. Ao terminar a operação de E/S ou ocorrendo o evento aguardado, o processo passa ao estado de *pronto*.

Terminado

O programa foi totalmente processado, produziu os resultados e pode ser removido da memória do sistema e as estruturas de controle podem ser apagadas.

Para facilitar a sua compreensão, observe a animação a seguir:

BLOCO DE CONTROLE DE PROCESSO

Os programas do interior do núcleo são responsáveis pela manutenção do meio ambiente onde os processos existem. Portanto, esses programas devem operar sobre estruturas de dados que correspondam à representação física de todos os processos do sistema.

Cada processo é representado por um descritor (bloco de controle ou BCP), que é uma área de memória contendo todas as informações relevantes do processo. Veja, a seguir, um exemplo de BCP:



No BCP, controlamos uma série de dados incluindo:

Informações de escalonamento de UCP: incluem prioridade de processo, ponteiros para filas de escalonamento e quaisquer outros parâmetros de escalonamento.
Estado do processo: pronto, executando ou bloqueado.
Número do processo: constitui o seu identificador dentro do Sistema Operacional.
Contador do programa: indica o endereço da próxima instrução a ser executada para esse processo.
Registradores de UCP: correspondem ao conteúdo dos registradores da CPU no momento em que o programa saiu do estado de executando para bloqueado ou pronto. Juntamente com o contador do programa, essas informações de estado devem ser salvas para permitir que o processo continue corretamente depois.
Informações de gerenciamento de memória: podem incluir dados como o valor dos registradores de base e limite, as tabelas de páginas ou tabelas de segmentos, dependendo do sistema de memória usado pelo sistema operacional.
Informações de contabilização: incluem a quantidade de UCP e o tempo real usados, limites de tempo, números de contas, números de jobs ou processos etc.
Informações de status de E/S: incluem a lista de dispositivos de E/S alocados para este processo, uma lista de arquivos abertos e outras informações.

As informações do BCP podem ser agrupadas em **contexto de hardware (glossário)** e em **contexto de software (glossário)** do processo, isto é, informações que precisam ser salvas quando o processo perde o controle de um processador, de maneira a possibilitar o reinicio da execução quando o processo passar novamente a executar.

, Os estados dos processos incluem ainda *novo* e *terminado*. Porém, estes estados não são representados no BCP, pois não correspondem a processos ativos. No caso de *novo* o BCP está sendo criado, e no de *terminado*, sendo destruído.

TROCA DE CONTEXTO

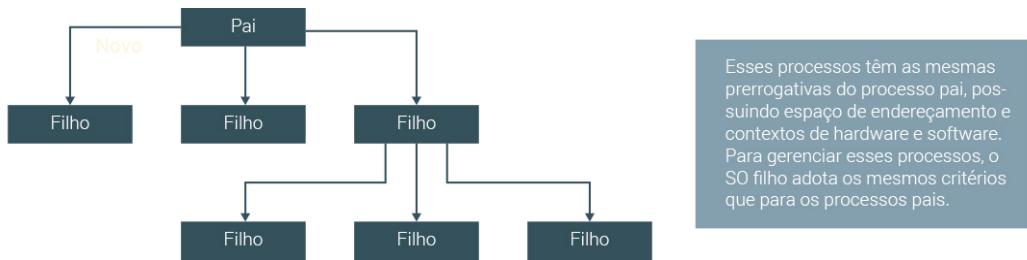
Quando ocorre a preempção ou um processo solicita E/S um novo processo é escalonado. Para que o processo interrompido possa voltar a ser executado corretamente no futuro, o seu contexto deve ser salvo no BCP, e o novo processo e o que irá iniciar a execução, ter o seu contexto lido no BCP.

A estas tarefas denominamos **Troca de Contexto** conforme exemplificado a seguir:



CRIAÇÃO DE PROCESSOS

Um processo, durante sua execução, pode ativar outros processos através de uma chamada de sistema para criar o que se chamam *processos filhos*, formando uma árvore de processos, como ilustrado a seguir:



TIPOS DE PROCESSO

Os processos podem ser classificados de acordo com o tipo de processamento que realizam, a saber:

CPU-bound (ligado à CPU)

Passa a maior parte do tempo no estado de execução, ou seja, utilizando o processador. Esse tipo de processo realiza poucas operações de E/S.

I/O-bound (ligado à E/S)

Passa a maior parte do tempo no estado de bloqueado, pois realiza um elevado número de operações de E/S.

THREADS

Naturalmente que a gerência necessária sempre que se muda o estado de um processo, é onerosa (em termos de tempo) e indesejável do ponto de vista da melhoria de *performance* do sistema.

Para contornar esse problema e diminuir o custo de manipulação de processos, instituiu-se outra unidade chamada *thread*, que não possui todas as prerrogativas de um processo, mas também permite algum tipo de paralelismo das funções.

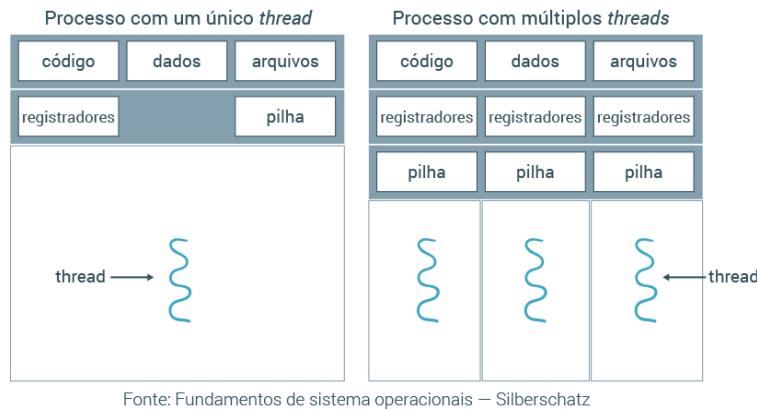
Mas, o que são Threads?



Você já ouviu falar em threads? Descreva o que você sabe sobre esse assunto.

Resposta Correta

Compare, abaixo, os dois tipos:



Fonte: Fundamentos de sistema operacionais – Silberschatz

Dentre os benefícios do uso de *thread* temos:

Capacidade de resposta

Ao se utilizar várias *threads*, em uma aplicação interativa, podemos permitir que parte de um programa continue executando, como, por exemplo, a interface do usuário, enquanto outra parte esteja bloqueada, realizando, por exemplo, uma operação de entrada/saída.

Compartilhamento de recursos

As *threads* compartilham o código, os dados e os arquivos do processo ao qual pertencem, o que permite que uma aplicação possua várias threads de atividades dentro do mesmo espaço de endereçamento.

Economia

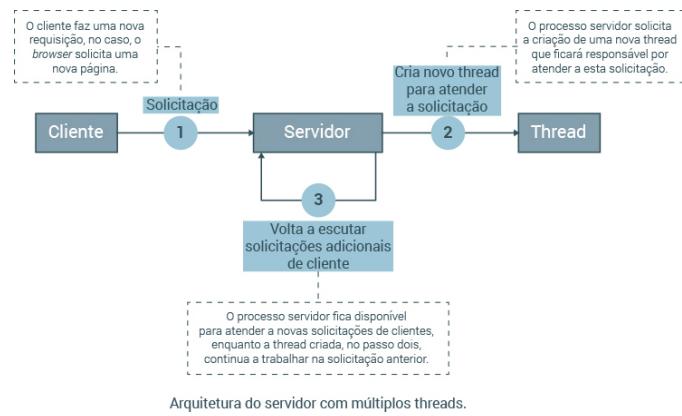
A alocação de memória e recursos para a criação de processos é dispendiosa. Como as *threads* compartilham estes recursos do processo, é muito mais rápido criá-las e permutar seus contextos.

Escalabilidade

Em sistemas multiprocessados, a utilização de threads permite que várias atividades, do mesmo processo, sejam realizadas em paralelo nos diversos núcleos.

VEJAMOS UM EXEMPLO DE USO DE THREAD:

Um programa servidor, como, por exemplo, o servidor web Apache obtém um enorme benefício como o uso de *threads*, pois ao invés de criar um novo processo para atender às requisições de usuários, pode criar *threads*.



APLICAÇÕES E SERVIÇOS

Como vimos, processo é um programa em execução, mas qual a diferença entre o processo de uma aplicação, como o Word, para o servidor de um banco dados como o Oracle?

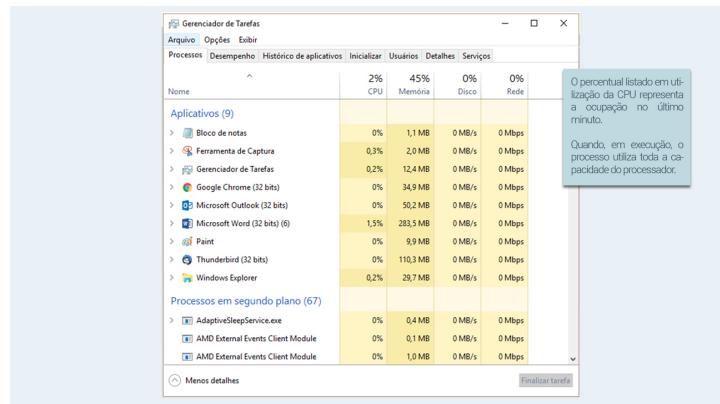


Existem aqui dois conceitos diferentes, uma aplicação é um programa que você chama para executar; ele cria um processo e fica ativo até você o fechar.

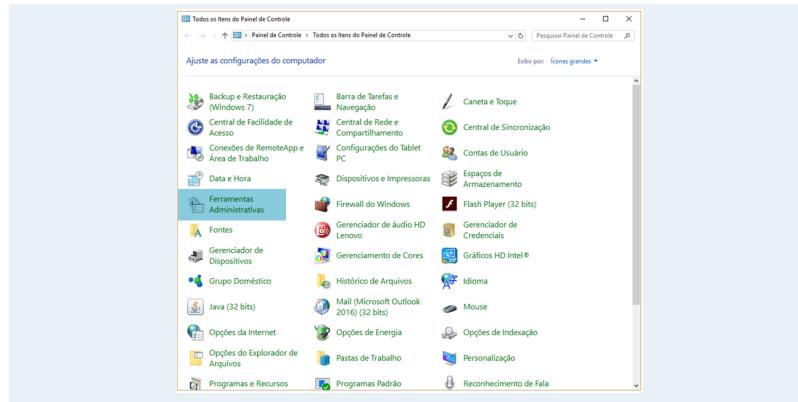
Um serviço é um programa que fica executando em segundo plano, normalmente para oferecer alguma funcionalidade de servidor.

A figura a seguir mostra o gerenciador de tarefas do Windows 10 onde podemos ver, na aba processos, os aplicativos e os processos em segundo plano, em sua maioria serviços.

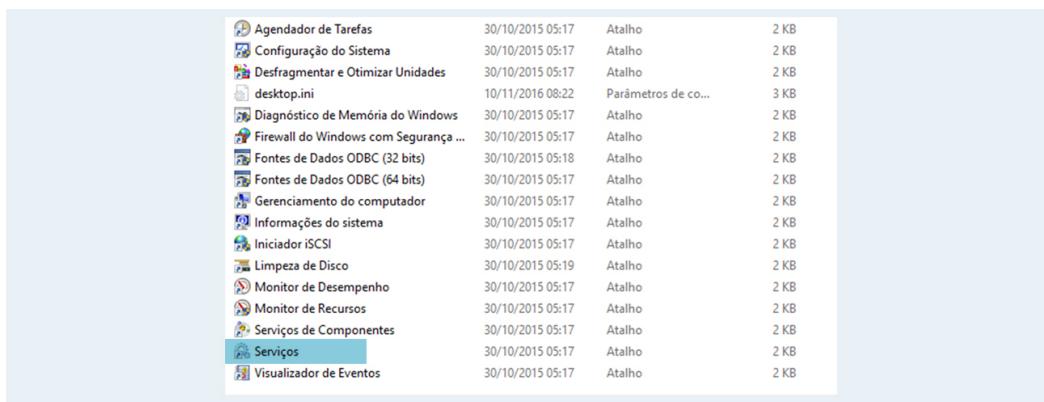
Observe ainda que o sistema faz o controle dos recursos consumidos por cada processo.



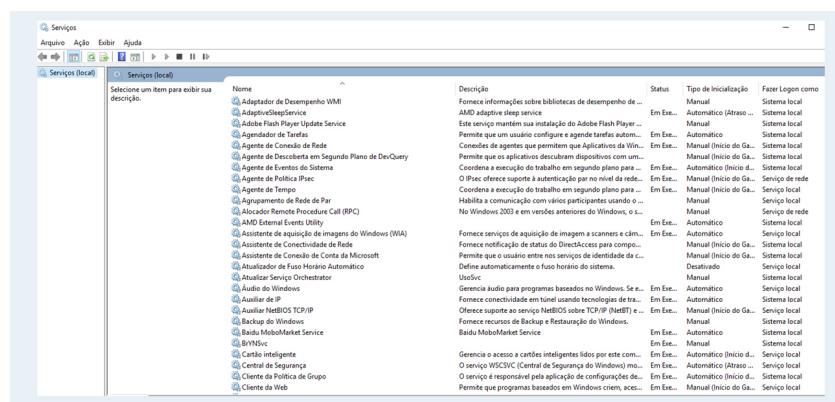
Para visualizar os serviços existentes, em sua máquina Windows, acesse no painel de controle: Ferramentas administrativas.



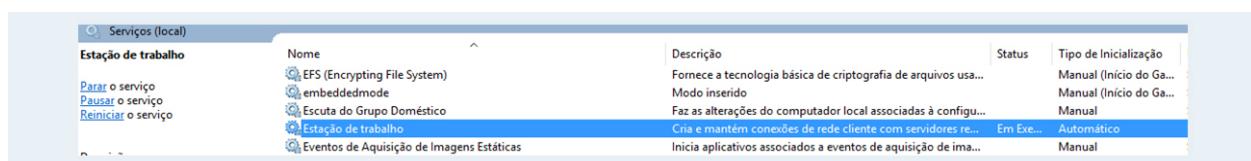
Na janela aberta, deve escolher *serviços*.



Será exibida então uma janela com todos os serviços instalados no SO.



Ao selecionar um serviço aparecerão as opções de *parar*, *pausar* ou *reiniciar* o serviço se ele estiver em execução (observe a coluna *status*).



Ou de iniciar o serviço se ele estiver parado:

Serviços (local)						
	Nome	Descrição	Status	Tipo de Inicialização	Fa	
Iniciar o serviço	EFS (Encrypting File System)	Fornecê a tecnologia básica de criptografia de arquivos usa...	Manual (Início do Ga...	Si		
	embeddedmode	Modo inserido	Manual (Início do Ga...	Si		
	Escuta do Grupo Doméstico	Faz as alterações do computador local associadas à configu...	Manual	Si		
	Estação de trabalho	Cria e mantém conexões de rede cliente com servidores re...	Em Exe...	Automático	Se	
	Eventos de Aquisição de Imagens Estáticas	Inicia aplicativos associados a eventos de aquisição de ima...	Manual	Si		

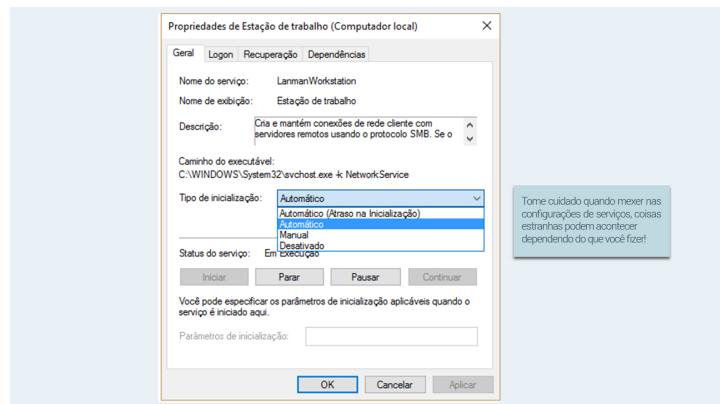
Um serviço pode ser iniciado automaticamente pelo SO ou de forma manual.

No caso de início automático, o serviço é inicializado durante o boot do sistema, já de forma manual deve ser inicializado pelo usuário.

Observe, abaixo, o tipo de inicialização do serviço.

Status	Tipo de Inicialização
	Manual (Início do Ga...
	Manual (Início do Ga...
	Manual
Em Exe...	Automático
	Manual
Em Exe...	Automático
	Manual
	Manual
Em Exe...	Automático
Em Exe...	Automático

Você pode alterar o tipo de inicialização, dê um duplo clique no serviço desejado, isso abrirá a janela de propriedades do serviço onde você poderá acessar várias informações e alterar configurações.



ATIVIDADE

Para esta atividade, você deverá fazer o download do [simulador S0sim \(glossário\)](#).

O S0sim é um software educacional para ser utilizado como ferramenta de apoio em aulas de sistemas operacionais.

Foi desenvolvido pelo Prof. Luiz Paulo Maia como parte de sua tese de mestrado, no Núcleo de Computação Eletrônica, da Universidade Federal do Rio de Janeiro.

Em seu livro, o professor disponibiliza várias atividades utilizando o simulador.

Após realizar o download, clique [aqui \(glossário\)](#) e leia a atividade.

Assista aos vídeos para aprender a utilizar o SOsim:

[Tutorial 1 \(glossário\)](#)

[Tutorial 2 \(glossário\)](#)

Para ver as resposta, clique [aqui \(glossário\)](#).

Você pode assistir ao vídeo de resolução da atividade. Para isso, clique [aqui \(glossário\)](#).

Glossário

SO

Como sabemos, o SO é um conjunto de atividades que fornece as funções básicas de controle da máquina, como por exemplo, o escalonamento de tarefas, a manipulação de E/S etc.

PARALELISMO

Conforme veremos na próxima aula, temos o pseudoparalelismo quando os programas apenas parecem ser executados ao mesmo tempo, como ocorre em sistemas monoprocessados, e o paralelismo real quando eles efetivamente são processados ao mesmo tempo, cada um em processador ou *core* diferente.

FILA DE PRONTOS

Os processos são organizados em uma fila para serem escalonados, ou seja, colocados em execução. A ordem de execução é determinada por algoritmos de gerencia de CPU que serão estudados na próxima aula.

CONTINUAR SUA EXECUÇÃO

Além dos processos que estão na fila, por terem sido criados (Novo) os processos que estavam bloqueados, esperando um dado ou evento, e os que estavam executando e sofreram preempção, voltam à fila de prontos para serem novamente escalonados, visando continuar a execução do ponto onde pararam.

INTERRUPÇÃO

Uma interrupção é um evento externo que faz o processador parar a execução do programa corrente e desviar a execução para um bloco de código chamado rotina de interrupção.

Normalmente, são decorrentes de operações de E/S ou do relógio do sistema.

Iremos estudar mais a respeito de interrupções em uma aula futura.

CONTEXTO DE HARDWARE

O contexto de hardware é constituído basicamente dos valores dos registradores/*flags* da máquina antes do bloqueio.

CONTEXTO DE SOFTWARE

O contexto de software é constituído da identificação do processo, seus limites em termos de recursos (número máximo de arquivos, tamanho de memória permitido, número de subprocessos permitidos etc.) e nível hierárquico do processo (que determina o que o processo pode ou não fazer em termos de operações protegidas do sistema).