

TIPOS SISTEMAS OPERACIONAIS:

- **Monotarefa**
 - 1 processo, 1 usuário Exemplo: MS-DOS
- **Multitarefa**
 - Vários processos, 1 usuário (Windows, Linux).
 - Tempo de espera para E/S: 80 a 90% do tempo total de processamento => divisão da memória:
 - enquanto job espera E/S se completasse, outro poderia usar a CPU.
 - Compartilhamento de tempo (multiprocessamento)
- **Multiusuário**
 - vários processos, vários usuários
 - Sistemas operacionais em rede
- **Sistemas operacionais distribuídos**
 - Ilusão de um processador único.
 - Usuário não sabe onde o seu processo está sendo executado, ou onde os seus arquivos estão localizados.
- **PROCESSOS - Conceito chave em SO.**
 - Processo = programa em execução.
- **MULTIPROCESSAMENTO - CPU produz a ilusão de processos rodando simultaneamente. Como o SO interrompe os processos, há a necessidade de reiniciá-los mais tarde a partir do ponto onde foram interrompidos.**
- **ESCALONAMENTO:** recuperar informações sobre o estado do processo quando foi interrompido. Esta informação é armazenada em estrutura chamada tabela de processos.
- **GERENCIA DE MEMÓRIA**
 - A memória principal do computador (RAM) é utilizada para armazenar os espaços de endereçamento dos processos.
 - O mecanismo de memória virtual é utilizado para "estender" a memória principal (física) através do uso do disco.
- **ENTRADA E SAÍDA (E/S) :** teclado; mouse; vídeo; impressora; etc.
 - Impõem a necessidade de gerenciamento por parte do SO.
- **SISTEMAS DE ARQUIVOS - O SO deve ser capaz de localizar dados em arquivos armazenados em discos magnéticos ou outros meios.**
 - Organização do sistema de arquivos: diretórios = agrupamentos de arquivos e outros diretórios.
- **Em UNIX: Diretório raiz: / (barra)**
- **Em Windows:**
 - Diretório raiz: C:\

Introdução

- **Estudo avançado sobre funcionamento dos sistemas operacionais.**
- **Ex. Usuário precisa editar um texto e usar um sistema qualquer para gerenciamento de informações.**
- **Independente do que esta sendo utilizado a forma de acesso aos periféricos é igual para todos os programas.**
- **Para melhor aproveitamento do hardware (servidor por exemplo) vários usuários acessar um mesmo hardware através da rede.**
- **Assim os programas podem apresentar necessidades conflitantes.**
- **SO – é uma camada de software colocada entre o hardware e os programas que executa a tarefa para os usuários.**
- **Ele é responsável pelo acesso aos periféricos.**
- **Sempre que um programa precisa de uma operação de entrada e saída ele solicita ao SO.**
- **Programador ou usuário não precisa saber detalhes do hardware.**

Objetivos do SO

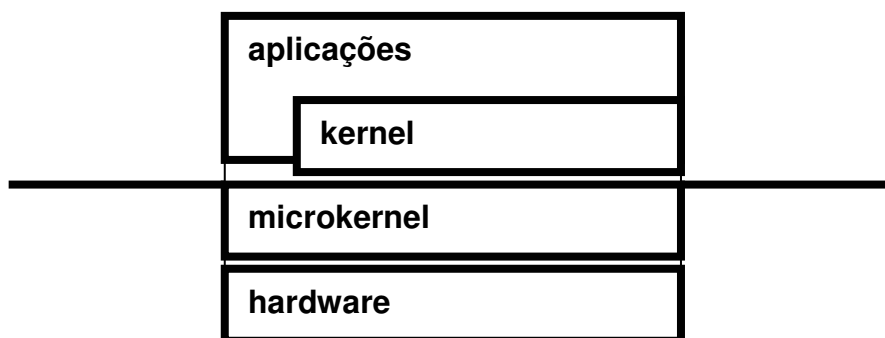
- **Procura tornar a utilização do computador mais eficiente e conveniente.**
- **Eficiente pelo maior retorno no investimento realizado no hardware, distribuindo os recursos de hardware com os programas que precisam de periféricos verificando o tempo de processador, acesso a disco ou espaço em memória.**
- **Conveniente pois esconde do programador detalhes de hardware quando imprimir um caractere na tela por exemplo. Ele não precisa saber qual a saída deve ser utilizada, por qual hardware.**

Tipos de serviço

- **O SO oferece meios para que o programa seja carregado na memória principal e executado. Assim o SO recebe o nome do arquivo, aloca memória par ao programa, copia o conteúdo do arquivo para a memória principal e inicia a execução. Também aborda o programa e retira-o da memória.**
- **O serviço mais importante do SO é o sistema de arquivos, que permite gravar, excluir arquivos no disco.**
- **Acesso aos periféricos**
- **Compartilhar o mesmo micro por usuários em rede**

Arquitetura do SO.

- Corresponde a imagem que o usuário tem do SO, como ele interage com o software. A imagem de interação é formada por:
 - Chamada de sistema – são serviços que os programas solicitam ao SO. AS chamadas de sistema passar a execução para o SO, através de parâmetros, o programa informa o que precisa. O retorno da Sub-rotina faz com que a execução do programa seja retomada a partir de onde segue a chamada para o SO. Ex. Programando em Assembly INT 21H. Em linguagens de alto nível isso fica escondido na biblioteca do compilador.
 - A parte responsável – núcleo ou kernel
 - Componentes de um kernel de qualquer SO – gerencia de processador, gerencia de memória, sistema de arquivos, gerencia de entrada e saída.
- Internamente o SO é dividido em microkernel que implementa serviços básicos do SO em relação ao hardware, os demais serviços são implementados pelo kernel.



Programas de sistemas – utilitários – programas normais executados fora do kernel pelo SO. Ex. utilitários para manipular arquivos (Ls -la, ou windows explorer).

- SO não serve para editar texto, sendo assim não resolve problema do usuário. Através dele podemos obter maior eficiência e conveniência do uso do computador.
- Eficiência – compartilhamento de recursos
- Compartilhamento interface confortável para utilização dos recursos.

MultiProgramação

- Vários programas são mantidos na memória ao mesmo tempo.
- A maioria dos programas não utiliza toda a memória quando executado, utilizam apenas uma parcela da memória disponível.
- Sem multiprogramação a memória não ocupada ficaria sem utilização.
- Ao terminar operação de entrada e saída do programa 1, ele pode voltar a ocupar o processador, entretanto o programa 2 está sendo executado. Será necessário selecionar qual deles ficará com o processador.
- Algoritmos para essa situação são utilizados para gerenciar o processador.

Processos

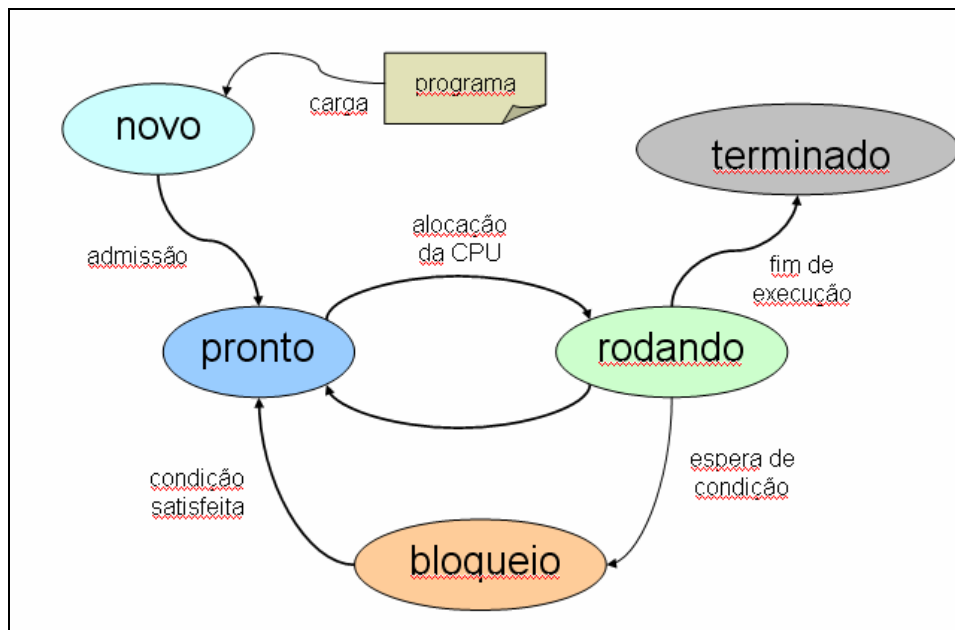
- Diferencia um programa em execução.
- Um mesmo programa pode estar sendo executado por vários usuários.
- Um programa é uma seqüência de instruções, não altera seu estado.
- Processo é um elemento que altera seu estado a medida que executa o programa.
- Processo faz chamadas de sistema ao executar programas.
- Processos executam tarefas dos usuários, porém algumas vezes podem realizar tarefas de sistemas, neste caso são chamados de processo de sistema.
 - Ex. Evitar conflito com impressora, sistemas trabalham com técnica de spooling. Para imprimir as informações vão para um diretório especial onde o processo copia deste diretório para a impressora, assim o usuário não precisa esperar a impressora ficar livre para poder imprimir.
 -
- E/S- ao ser criado o processo passa a ocupar processadores. As vezes ele deixa de ocupar o processador para realizar uma operação de E/S. Assim:
 - Ciclo de processador e ciclo de E/S.
 -

Estado de um processo

- Ao ser criado o processo entra em um ciclo de processador. Ele precisa ser processado para ser executado e neste tempo pode ser que o processador já esteja executando outros processos. Assim o processo deverá esperar sua vez.
- Ex. Processo 1 está acessando um dispositivo. Processo 2 está executando. Ao terminar a E/S do 1 ele precisa ser processado para voltar a executar. Mas o processador está ocupado com o 2 então 1 deve esperar. Entretanto o SO pode resolver processar o 1 imediatamente e interromper o 2.
- Fila de espera- fila de processos a executar, somente 1 pode usar o processador ao mesmo tempo.

- O processo que ocupa o processador esta estado rodando.
- O processa que esta na fila de apto para executar, esta pronto.
- O processo estando sendo executado. O processo faz chamadas ao sistema. Ate a chamada ser atendida ele não pode continuar a executar, então fica na fila de bloqueado. Ele SO pode continuar a disputar o Processo após a conclusão da chamada.
- **PROCESSOS PRONTOS COMPARTILHAM A CPU.**
- **CADA PROCESSO RECEBE UMA PEQUENA FATIA DE TEMPO (20 MS).**

Observe o diagrama de estado de um processo



Escalonador

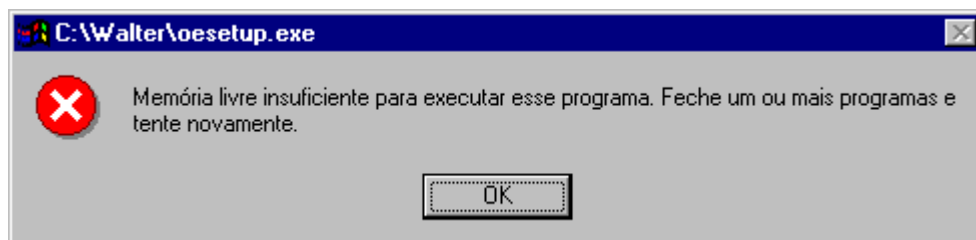
- Processo selecionado passa do estado de apto para executado.
- Muitos sistemas procuram evitar que um único processo monopolize a ocupação do processador. Se um processo esta há muito tempo no processador. Ele volta para o fim da fila de aptos. Um novo processo no topo da fila de aptos ganha o processador. Desta forma cada processo tem chance de executar um pouco.

Gerencia de filas.

- Os processos estando na fila de aptos esperando para utilizar o processador podem também utilizar dispositivos de ES como discos e impressora independente de outros processos que estão utilizando processador.
- O gerenciamento da fila é realizado entre processador e periféricos de ES. SE esses estiverem já executando, outros possíveis processos estarão bloqueados na fila de espera.

HIERARQUIA ENTRE PROCESSOS

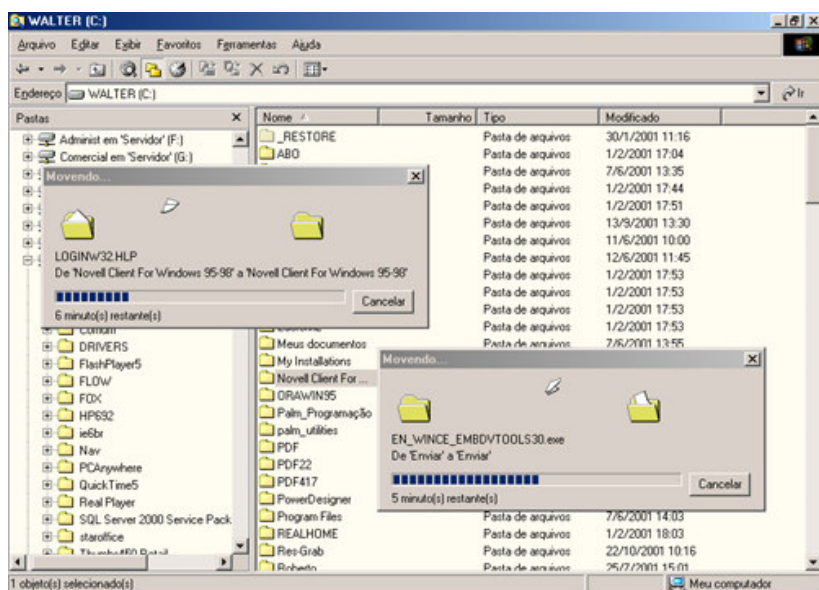
- Um processo pode criar outros processos e que podem, por sua vez, criarem também outros processos de maneira hierárquica.
- Este tipo de recurso evita que o usuário tenha que esperar que um processo termine para que sua requisição seja processada melhorando o desempenho do sistema.
- Nos SO's atuais que são multitarefa e muitas vezes são obrigados a rodarem em máquinas domésticas e de recursos de hardware limitados, de forma que depois de um certo número de subprocessos a situação se torna crítica gerando erros no sistema como podemos ver abaixo:



- Não confunda hierarquia de processos com multiprocessamento. O ambiente de multiprocessamento é aquele ambiente onde vários processos podem ser executados ao mesmo tempo, o chamado "Multitarefa".
- Subprocesso acontece quando por exemplo de um software Delphi vc chama um comando do Dos.

THREAD'S

- um processo pode efetuar várias operações concorrentemente ou simultaneamente através das chamadas "Linhas de execução".
- Threads compartilham o processador da mesma maneira que um processo.
- Por exemplo, enquanto uma Thread espera por uma operação de I/O, outra Thread pode ser executada mas todas elas compartilham o mesmo espaço de endereçamento pois lembre-se que o processo é um só.

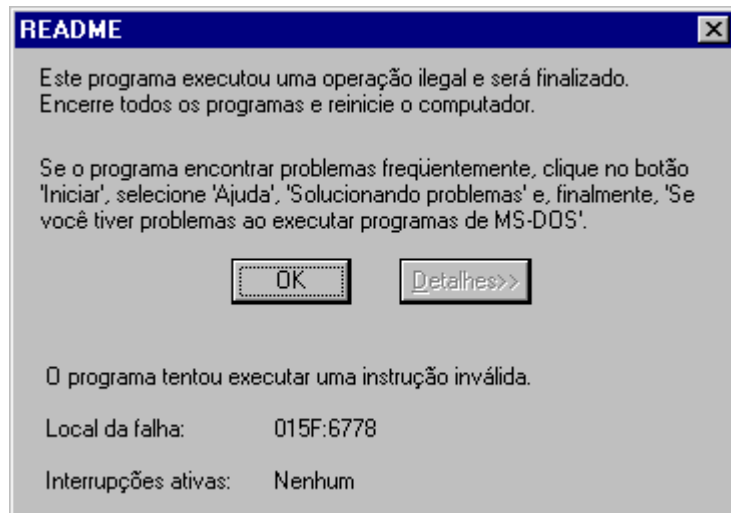


Exemplo de uma thread. Duas seções de copia de arquivos no Explorer são iniciadas

simultaneamente. Isto somente é possível porque cada seção corresponde a uma thread.

COMUNICAÇÃO ENTRE PROCESSOS

- É comum processos trabalharem concorrendo e compartilhando recursos do sistema, como arquivos, registros, dispositivos e áreas de memória. Um recurso mal usado ou usado indevidamente pode comprometer o sistema ou o próprio processo gerando falhas como podemos ver abaixo:



- Um processo só poderá gravar dados no Buffer caso ele não esteja cheio, da mesma forma, um processo só poderá ler dados armazenados no Buffer se existir algum dado a ser lido.
- Para gerenciar este compartilhamento de forma que dois ou mais processos tenham acesso a um mesmo recurso compartilhado, existe um mecanismo que controla este acesso, chamado de **MECANISMO DE SINCRONIZAÇÃO**.
- Propósito de garantir a confiabilidade e a integridade da gravação dos dados, evitando que os dados armazenados fiquem sem consistência.
- Como exemplo dois processos efetuando operações de gravação, de dados diferentes, em disco exatamente no mesmo setor ou no mesmo arquivo.

Figura abaixo mostra uma falha ocasionada quando um processo apresentou problemas e acabou interferindo no funcionamento de outro:

