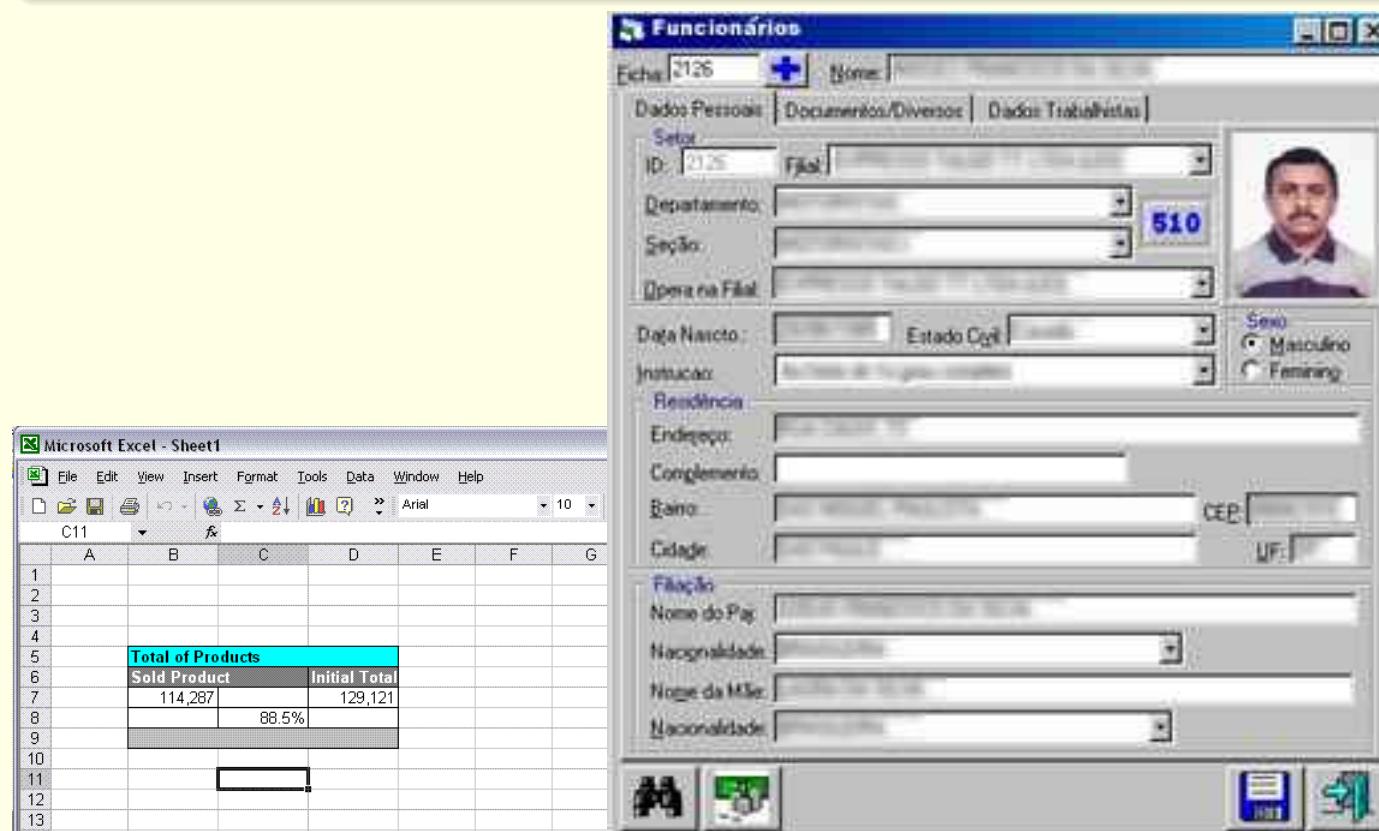


O que espero do meu computador ?

- Que me ajude a resolver problemas...
- Que controlar a contabilidade da empresa;
- Que armazene os cadastros dos funcionários da empresa;
- Que resolva problemas matemáticos, etc.



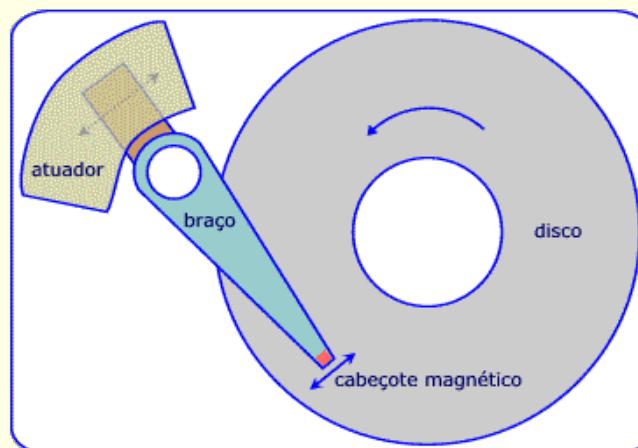
Mas não é bem assim que são as coisas....

Infelizmente o computador (hardware) é um dispositivo simples e utilizar um dispositivo simples pode ser tedioso. Imagine que você é um programador C e escreveu o programa abaixo:

```
#include <stdio.h>

int main (void) {
    FILE *entrada;    unsigned char buffer[128];

    entrada = fopen ("meu_arquivo.dad", "rb");
    fread(buffer, 1, 128, entrada);
    .
}
```



Infelizmente o computador (hardware) é um dispositivo simples e utilizar um dispositivo simples pode ser tedioso. Imagine que você é um programador C e escreveu o programa abaixo:

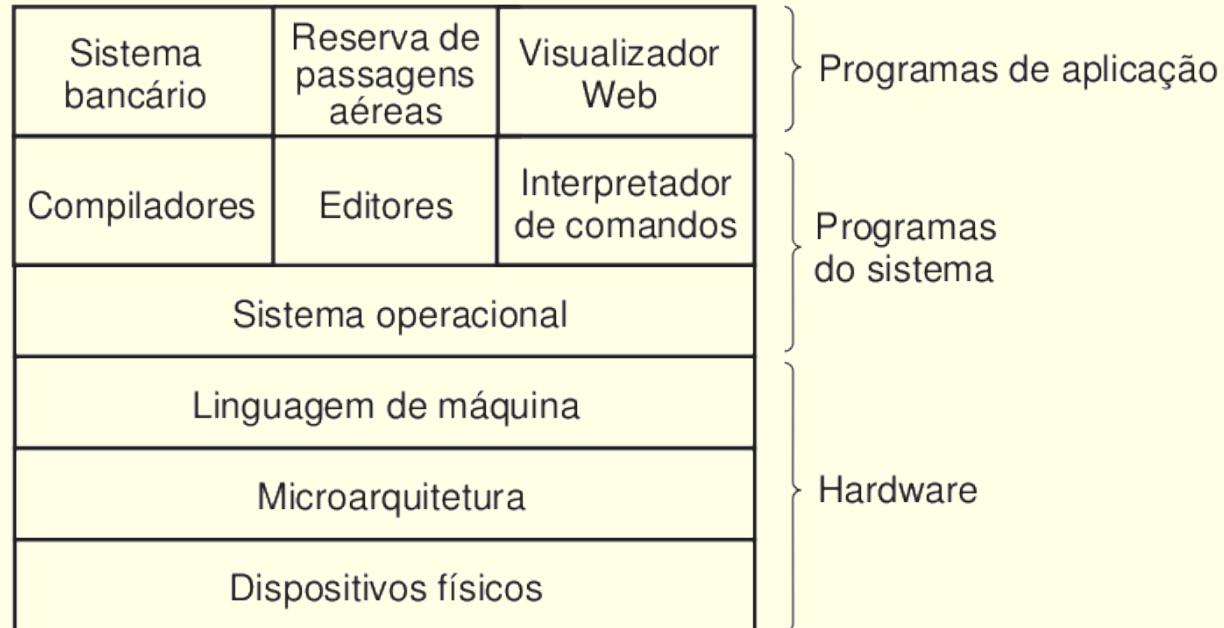
```
#include <stdio.h>

int main (void) {
    FILE *entrada;  unsigned char buffer[128];

    entrada = fopen ("meu_arquivo.dad", "rb");
    fread(buffer, 1, 128, entrada);
    .
}
```

Certamente você não sabe e não tem interesse em saber em que parte do disco rígido está o arquivo `meu_arquivo.dad`, muito menos se o disco rígido terminou de transferir os dados, se ele está rodando numa velocidade adequada para leitura, etc. Você simplesmente ignora todos os detalhes por que existe o Sistema Operacional (SO) que cuida deles por você.

O computador é projetado como um sistema em camadas



Imagine todos os dispositivos que temos hoje em dia: Disco rígido, porta serial, porta paralela, porta USB, cartão de memória SD/MMC, teclados, impressoras, telas de LCD, mouse, placa de som, placas de rede com e sem fio, placa bluetooth, etc. Sem a ajuda do SO, o programador de aplicação seria responsável por escrever rotinas para controlar cada um destes dispositivos.

Visões do SO?

- Máquina estendida
- Gerente de recursos
- Projeto

Máquina estendida

- Nesta forma, o SO esconde do usuário/programador os detalhes do hardware.
- O SO provê ao usuário uma forma fácil de acessar os dispositivos.
- O usuário do SO não sabe a velocidade da CPU, velocidade de gravação do HD, posição das cabeças do HD. Como estas coisas funcionam é responsabilidade do SO. Ele que faz o acesso ao hardware. Ou seja, o usuário pensa que possui um computador que entende “Arquivos”, “Pastas”, “mandar documento para a impressora”.

Gerente de recursos

- Nesta forma o SO gerencia os recursos do hardware. Se dois usuários querem imprimir um documento o que acontece ? E se forem usuários querendo escrever no disco ao mesmo tempo ? Será que todos os usuarios podem acessar e apagar todos os arquivos (inclusive de outros usuários) ?

Como classificar os SOs

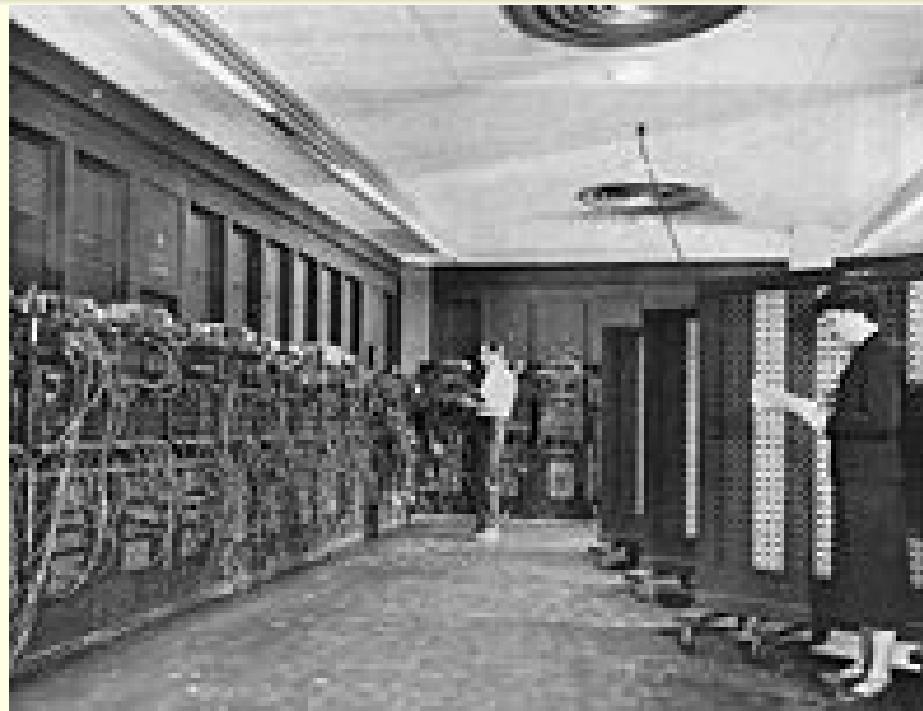
Projeto

Como o SO está organizado internamente
Chamadas de sistema
Implementação do sistema de arquivos (FAT, FAT32, NTFS, ext3, etc)
Implementação da gerencia de memória RAM
Implementação de processos (programas)
Implementação de entrada/saída

Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Primeira geração 1945-1955

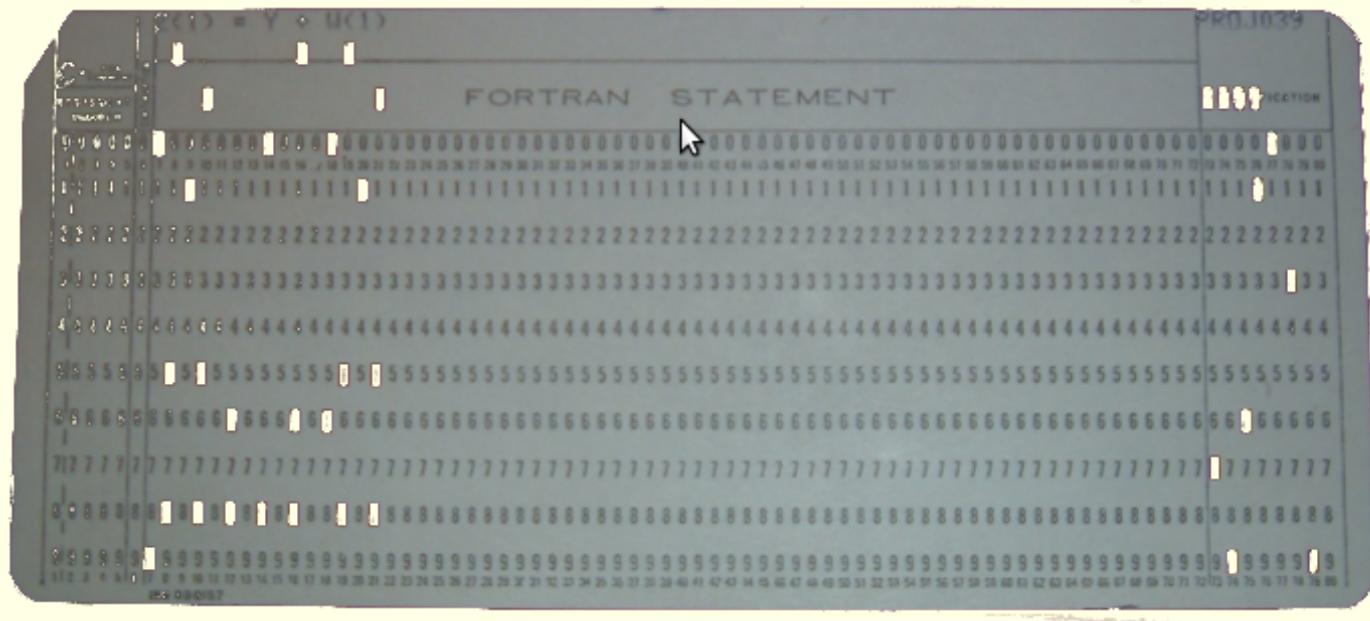
- Não existia SO, os programadores tinham acesso completo ao computador nos seus horários;
- Os programas acessam o hardware diretamente;
- Programador é também operador, projetista, construtor;



Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Segunda geração 1955-1965

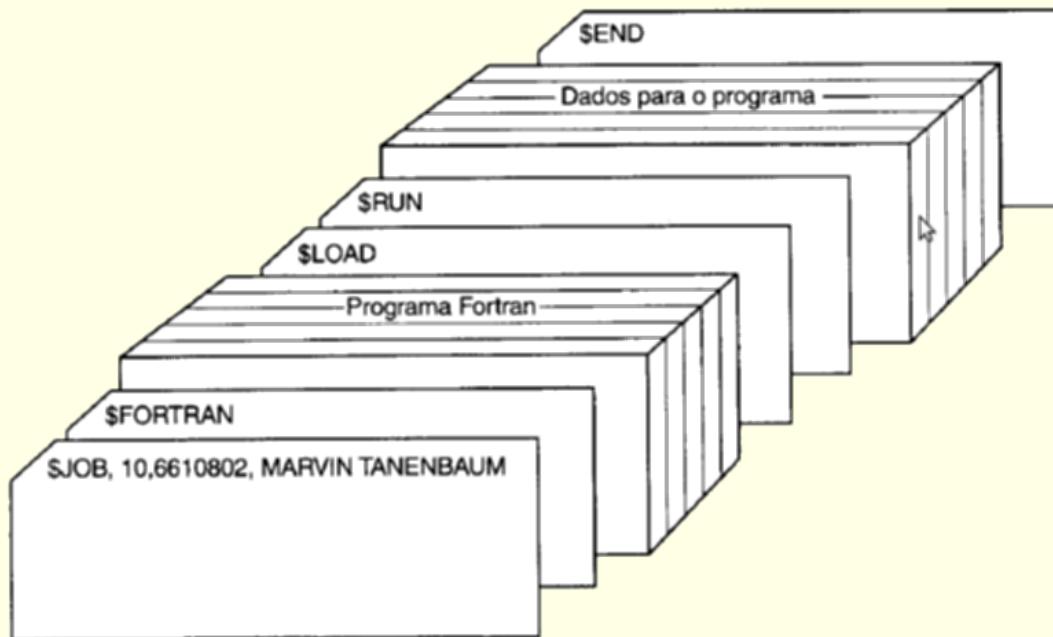
- Painéis de interligação evoluem para cartões perfurados;
- Um programa (job) em execução ocupa toda a máquina.



$$Z(1) = Y + W(1)$$

Este cartão representa apenas 1 linha de um programa escrito na linguagem Fortran.

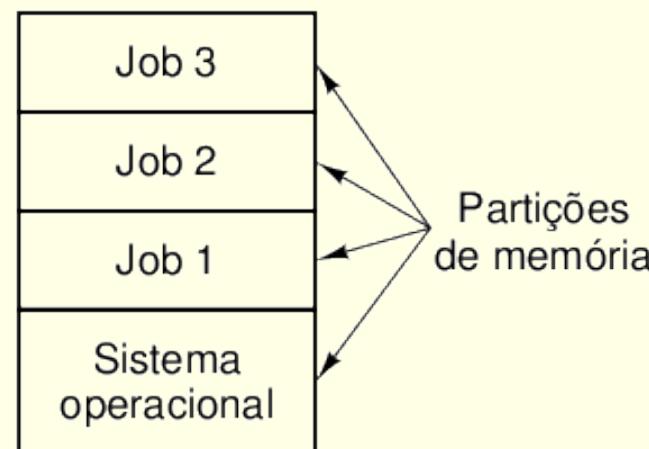
Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais



Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Terceira geração 1965-1980

- Circuitos integrados são utilizados em computadores, aumentando sua velocidade;
- Surge o conceito de multiprogramação;
- Sistemas operacionais passam a trabalhar com vários programas “ao mesmo tempo”;



Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Terceira geração 1965-1980

- Surge o Multics que foi um projeto que pretendia criar um SO revolucionário. Como ele avançado demais para a época, acabou sendo descartado. Algumas empresas que trabalhavam no grupo que tentou criar o Multics saíram do ramo da computação e nunca mais retornaram. O Multics influenciou gerações de projetistas de sistemas operacionais.
- Numa tentativa de fazer um Multics limitado foi criado o Unix que fez um tremendo sucesso e hoje é produzido por várias empresas ou disponibilizado como software livre (AIX, GNU-Linux, Solaris).
- Ken Thompson, Brian Kernigan, Denis Ritchie e seu papel na história.

Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Quarta geração 1980-hoje

- Geração conhecida como a dos computadores pessoais;
- Os Cls tornaram-se mais baratos e com mais transistores em seu interior;
- A Intel produziu em 1974 o primeiro microprocessador de uso geral, o 8080;
- Em 1980 a IBM lançou o primeiro PC e contratou Bill Gates para escrever uma linguagem de programação e o sistema operacional MS-DOS que seria utilizado em milhões de PCs, existiria por décadas e faria de Bill Gates o homem mais rico do mundo.



Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Quarta geração 1980-hoje

- O MS-DOS tinha um interface em que o usuário precisava digitar todos os comandos e digitar o nome dos programas que deseja executar;
- Com o tempo foi criada uma interface gráfica para facilitar o uso do computador. Inicialmente pela Apple e depois pela Microsoft;
- O mesmo ocorreu com o mouse, inventado pela Xerox e depois utilizado pela Apple e microsoft;



(Introdução)

Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Quarta geração 1980-hoje

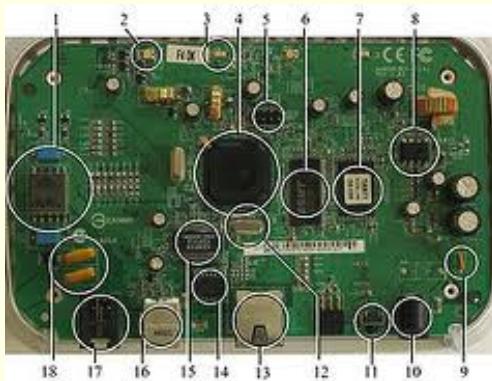
- Naquela época poucos deram valor para interface gráfica ou o mouse. Mas essas ferramentas ajudariam a criar impérios.



Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Sistemas operacionais quanto ao tipo de computador em que executa:

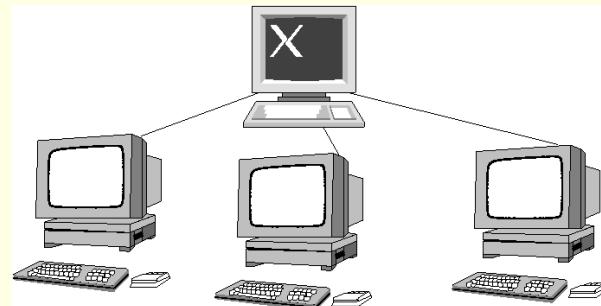
- SOs para mainframes
- SOs para computadores pessoais
- SOs para sistemas embarcados



Vamos voltar no tempo e ver como surgiram os Sistemas Operacionais

Sistemas operacionais também podem ser classificados em :

- SOs de tempo real
- SOs distribuídos



Sistemas operacionais quanto a utilização dos recursos de hardware:

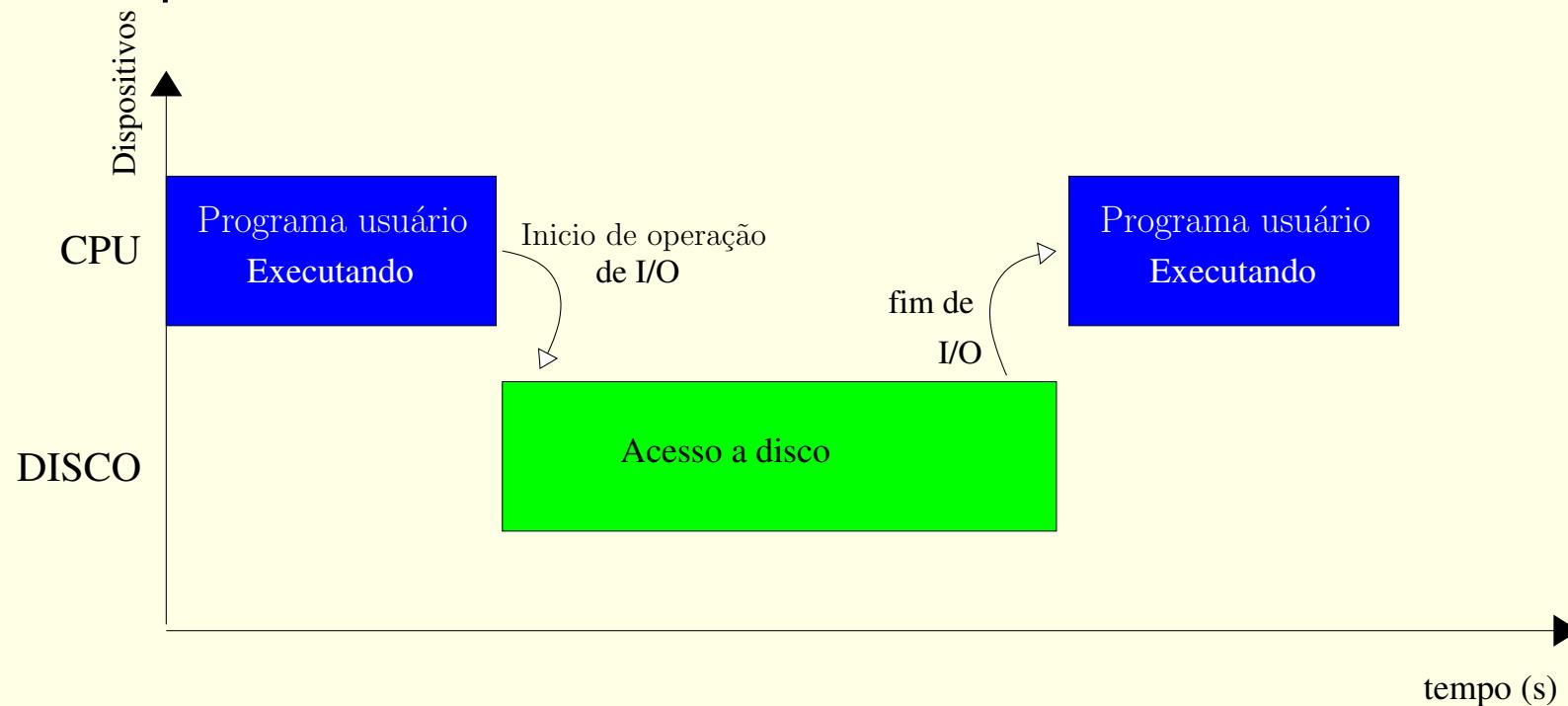
- SOs monotarefas

- Sistemas em que o processador, a memória e os periféricos ficam dedicados a um único usuário. Nesses sistemas, enquanto o programa aguarda por um evento, como a digitação de uma tecla, o processador ficará ocioso, sem realizar qualquer tarefa útil. A memória é subutilizada caso o programa não a preencha totalmente, e os periféricos estão dedicados a um único usuário. SOs monotarefas foram os primeiros SOs criados e permaneceram no mercado por muitos anos. Exemplos: CPM, MS-DOS.

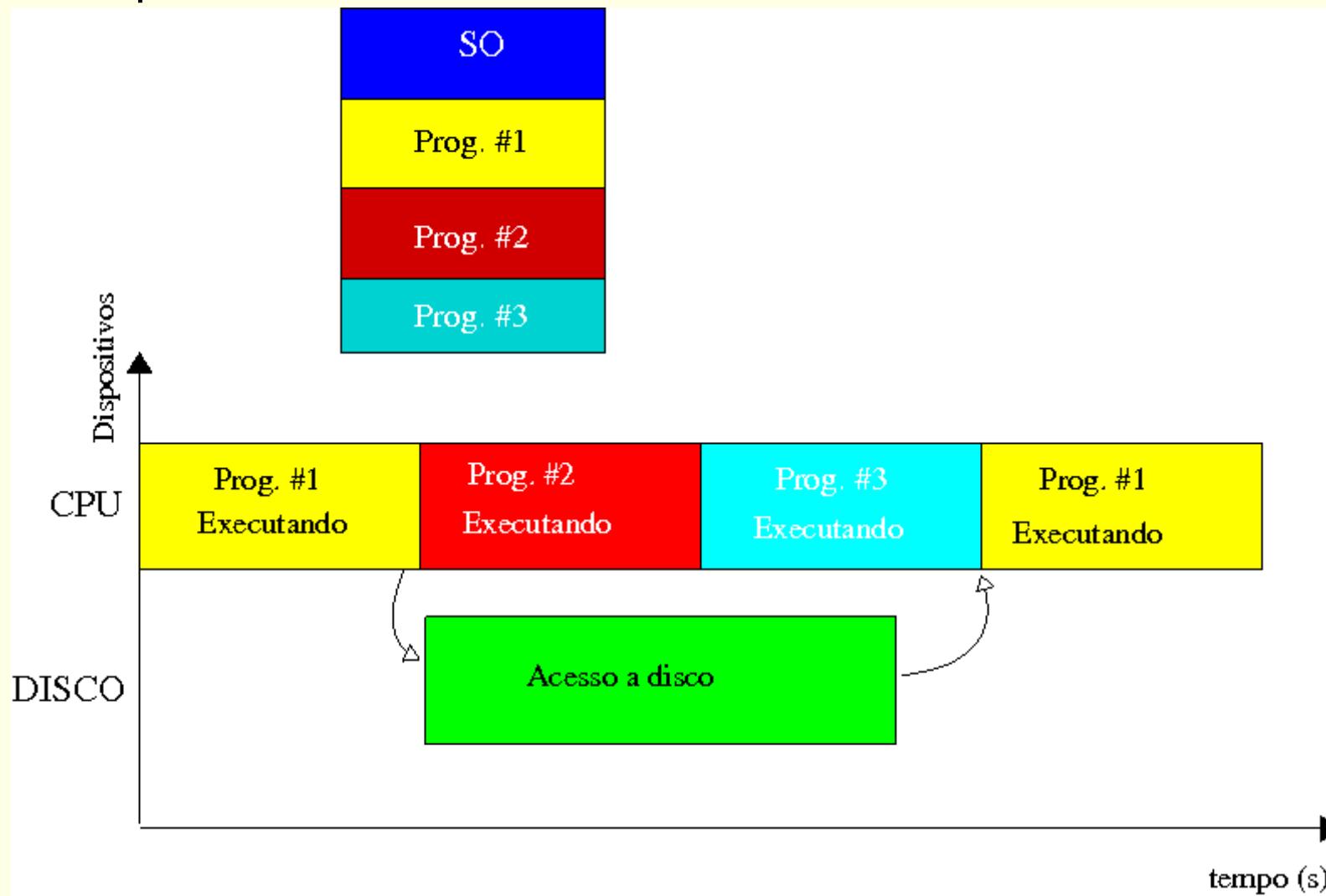
- SOs multitarefas

- Podem realizar várias tarefas ao mesmo tempo como imprimir um arquivo na impressora, rodar um processador de textos, compilar um programa, etc. Sistemas multitarefas são mais complexos de construir mas tem como benefício a melhor utilização dos recursos do computador. Se um programa não está fazendo nenhuma operação (está parado esperando a digitação de uma tecla) outro programa pode ser executado. O processador está sempre processando alguma informação.

Exemplo de um sistema Monotarefa:



Exemplo de um sistema Multitarefa:



Sistemas operacionais quanto a quantidade de usuários:

- SOs monousuários
 - Sistemas em que apenas um usuário pode utilizar o computador a cada vez. Este tipo de sistema é mais fácil de construir e foi muito popular entre os microcomputadores.
- SOs multiusuários
 - Vários usuários podem utilizar um computador ao mesmo tempo. Eles usuários podem **logar** no computador, disparar a execução de um programa e após isso **deslogar-se** do sistema deixando que outro usuário possa executar seus programas. Também é possível que um usuário esteja utilizando o computador e executando seus comandos remotamente. Sistemas multiusuários precisam garantir que o acesso a arquivos de um computador é permitido apenas para o dono do arquivo ou para quem o dono permitiu. Existem várias questões de segurança envolvidas neste tipo de sistema;

Sistemas operacionais quanto a forma de distribuição/propósito:

- SOs comerciais
 - São sistemas que precisam ser adquiridos em loja para serem instalados no computador do usuário. Compra-se um CD/DVD onde o SO está gravado. Chegando em casa o usuário lendo o manual que vem junto pode colocá-lo no seu computador. Geralmente o usuário conta com assistência técnica para ajudá-lo a resolver problemas na instalação/uso; Ex: Windows 7, Vista, etc.
- SOs grátis
 - O usuário pode obter pela internet o sistema operacional ou pode mesmo pagar pelo preço do CD/DVD mas o software é grátis. Em alguns casos o usuário pode *baixar* o sistema operacional pela internet utilizando algum computador, gravar num CD/DVD e instalar no seu computador em casa; Ex: Solares

Sistemas operacionais quanto a forma de distribuição/propósito:

- SOs software livre
 - Assim como o software grátis o usuário não precisa pagar pelo sistema operacional e alem disso, pode obter os códigos de computador que fazem o sistema operacional funcionar. Assim, caso queira, pode fazer alguma melhoria ao sistema ou estudar como este funciona. Ex: GNU-Linux
- SOs educativos
 - São sistemas operacionais que foram escritos para permitir que estudantes estudem o seu código e entendam os conceitos de sistemas operacionais. O professor apresenta em sala de aula um assunto da disciplina de SOs e depois mostra como aquele assunto foi utilizado num sistema operacional. Ex: Minix

Um conceito importante em sistemas operacionais é o **Processo**. Um processo é basicamente um programa em execução, sendo constituído de seu código executável (instruções), dos seus dados (valores de suas variáveis), do estado de sua pilha, do estado atual dos seus registradores (*PC-program counter*) dentre outros.

Processos

Quando temos um sistema compartilhado, os processos executam durante algum tempo e terminada uma fatia de tempo (por exemplo 10ms) o processo precisa ser suspenso e em seu lugar outro processo será executado. Mais tarde, o processo retorna a execução como se nada tivesse ocorrido, ou seja, se ele estav executando a instrução número 500 antes de ser suspenso e ao retornar deverá executar as instruções iniciando na 501. Se o processo estivesse trabalhando com arquivos e estes deveriam permanecer nas mesmas posições após o processo voltar a executar.

Tabela de processos - O sistema operacional deve manter uma lista dos processos que estão no sistemas. Essa lista contem os valores dos registradores no momento em que este foi suspenso.

Árvore de processos - Um processo pode, durante sua execução, criar um ou mais processos (**processos filhos**) e estes por sua vez podem criar outros processos. O resultado é uma árvore de processos. Existem formas de um processos terminar a execução de um processo filho e esperar o término natural de processo filho.

Comunicação entre processo - Um processo pode enviar informações para um outro processo no sistema, tal como enviamos mensagens através da rede.

Arquivos

O sistema de arquivos é a parte do sistema operacional mais conhecida pela usuário. O sistema de arquivos esconde do usuário (onde usuário é o utilizador do sistema e também o programador) as peculiaridades do acesso ao hardware e apresentar ao usuário um modelo abstrato como arquivos independentes dos dispositivos onde estão implementados. Existem chamadas de sistemas para **criar**, **remover**, **ler** e **escrever** arquivos. O arquivo precisa ser **aberto** antes de ser lido e após a leitura deve ser **fechado**. Existem chamadas de sistema para todas essas operações.

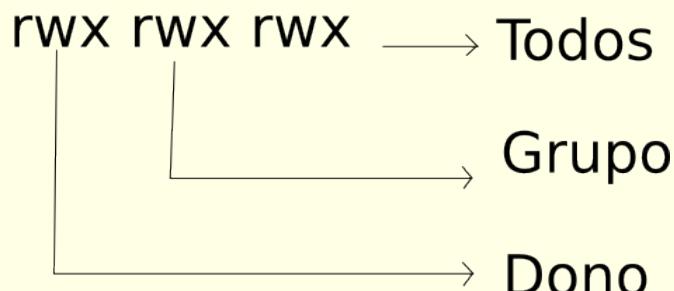
Diretório - Muitos sistemas implementam o conceito de diretório que é uma estrutura fictícia que permite agrupar um conjunto de arquivos. O sistema operacional deve prover chamadas para **criar diretórios**, colocar arquivos dentro de diretórios, remover um arquivo de um diretório.

Nome de caminho - Os diretórios, assim como os processos podem ter filhos (sub-diretórios) e estes mais filhos, formando uma árvore de diretório. O topo é denominado **raiz do diretório**. O nome do caminho consiste na lista de diretórios que precisa ser atravessada desde o diretório raiz até o arquivo.

Arquivos

Permissões Num sistema UNIX os arquivos e diretórios possuem bits de permissões para identificar quem pode realizar que operações sobre o arquivo/diretório. As permissões são denominadas **bits rwx**. Onde os bits indicam as operações que podem ser realizadas. bit r (*Read*), bit w (*Write*) e bit x (*eXecution*)

Exemplo: A listagem abaixo mostra que o arquivo introducao.tex possui os bits **rwxrwxrwx** mas nem todos estes estão ajustados. Os bits são separados em grupos de 3.



-rw-r--r--	1	frr	frr	20989	Mar	8	10:24	introducao.tex
drwxr-xr-x	2	frr	frr	4096	Mar	8	10:43	imagens

Arquivos especiais - Sistemas Unix representam dispositivos como se fossem arquivos, ou seja, pode existir um dispositivo chamado ttyS0 que é a porta serial do computador. Caso o arquivo seja aberto e dados sejam escritos no tal arquivo, estes mesmos dados serão enviados pela porta serial. O mesmo vale para outros tipos de dispositivos como cdrom, hd, pen-drive, cartão mmc. Possuir dispositivos representados como arquivos facilita em muito a vida do desenvolvedor de software que não precisa saber como acessar o hardware diretamente, ele apenas precisar acessar um arquivo.

Arquivos especiais de bloco - É como o Unix representa os dispositivos que podem ser acessados em qualquer posição, tal como HD, pen-drive, cartão mmc.

Arquivos especiais de caractere - É como o Unix representa os dispositivos que são sequências de caracteres como por exemplo a porta serial, a tela, a interface de rede.

O SO ajudando os programadores...

Quando um programador C deseja mostrar alguma mensagem na tela ele faz uma chamada de rotina de uma biblioteca, neste caso, a rotina **printf**. Por sua vez a rotina **printf** precisa realizar uma chamada de sistema (*system call*) para que os valores desejados sejam apresentados no dispositivo de saída (Tela).

Programa do usuário 1

Biblioteca C
(LibC)

Programa do usuário 2

Sistema Operacional

```
FILE *f;  
f = fopen ("arquivo.txt","r");  
fgets(buffer,100,f);
```

```
printf (char * st)  
{ INT $81 $4 $st  
}  
fopen (char *name, char *type)  
{  
    INT 81 $1 $10 $name ...  
}  
fgets (char *bf, int size, FILE *a)  
{ .... INT $81 $3 $bf $a ...  
}
```

```
if (x>10)  
{  
    printf("teste");  
}
```

```
INT $81:  
asm { mov $123, ax }  
asm { mov $cx,bx }  
switch (parametro) {  
    case $1: // eh open  
        .....  
        break;  
    case $4: //eh printf  
}  
return;  
//retorna ao ponto de chamada
```

Modo usuário

Modo Kernel

Chamadas de sistema - POSIX

Ou seja, o programa C faz uma solicitação e o SO atende a solicitação.

- Alguns sistemas operacionais criaram chamadas LE_DADO, outros LE_DISCO...Fazem a mesma coisa mas tem nomes diferentes
- Os programadores ficaram confusos, pois não sabiam se deveriam usar uma ou outra. Surgiu a necessidade de padronizar o nome das chamadas de sistema
- POSIX é um padrão que diz quantas chamadas de sistema, seus nomes e como são utilizadas.

Interpretador de comandos

O sistema operacional geralmente vem acompanhado de diversos programas (compilador C, editor de texto, montador assembly, linker e interpretadores de comandos). Estes programas não fazem parte do sistema operacional mas são muito importantes. Um destes, o interpretador de comandos denominado **shell** faz uso pesado de chamadas de sistema e é a interface entre o usuário e o sistema operacional.

Interpretador de comandos

Quando o usuário acessa a sua conta num computador UNIX o sistema operacional (após verificar a identificação do usuário) dispara a execução de um programa shell. O shell toma conta do teclado e da tela permitindo que o usuário digite palavras que são enviadas para a tela. Quando ele tecla ENTER, a palavra digitada é verificada e caso seja o nome de um programa o shell irá criar um processo filho para executar e executar este programa. Enquanto o processo filho estiver executando, o shell ficará esperando. Quando o processo filho termina, o shell envia uma nova mensagem para a tela e espera ler o novo comando.

```
date
```

```
date > saida.txt
```

```
sort < arquivo1.txt > arquivo2.txt
```

```
cat arq1 arq2 arq3 | sort >/dev/lp
```

```
xcalc
```

```
xcalc &
```

Estruturas dos Sistemas Operacionais

Sistemas Monolíticos - Nesta estrutura o sistema operacional é escrito como um conjunto de funções, umas funções chamam outras funções sempre que necessário. Todas estas funções são visíveis e precisa-se de especial cuidado para padronizar os parâmetros e retornos das funções pois caso contrário será uma grande confusão.

Sistemas em Camadas - Uma forma mais organizada de estruturar o sistema operacional é utilizando uma hierarquia de níveis. Ou seja, o sistema deve ser programado como conjuntos de rotinas que tratam de assuntos específicos. Por exemplo, no nível mais baixo teríamos rotinas que controlam o alocação do processador e chaveamento entre processos. Assim o nível mais baixo provê a existência de processos. Estes processos podem ser utilizados pelas camadas acima. O nível acima (nível 1) trata do gerenciamento de memória, o nível 2 trata da comunicação entre processos e assim por diante.

Máquinas Virtuais