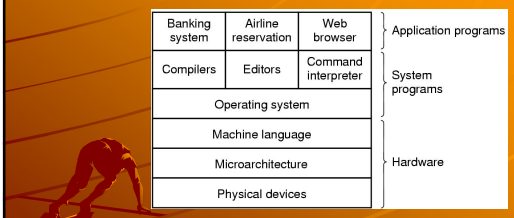
Introdução a Computação

Sistemas Operacionais PII

O que é um SO?



Consiste em:
- Hardware
- Programas de Sistema
- Programas de Aplicativos

O que é um SO?

- Hardware não proporciona controle de alto nível disponível para usuários
  - Linguagem de Montagem
- Extremamente difícil desenvolver programas que utilizem corretamente e eficientemente todos os componentes de hardware

O que é um SO?

- Um SO é um conjunto de programas, implementados tanto em software quanto em firmware, que torna o hardware possível de ser utilizado.
  - Extende a máquina escondendo características difíceis do hardware
  - Apresenta ao usuário uma máquina "virtual" fácil de se lidar

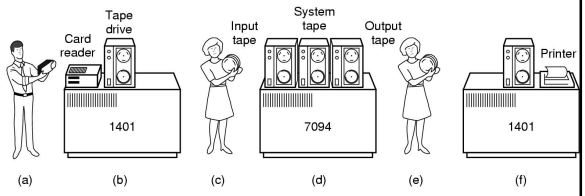
SO - Funções

- Apresenta ao usuário uma máquina mais flexível e adequada para se programar do que aquela que o hardware nu apresenta
- Gerencia e cria abstração para:
  - Processos
  - Memória
  - E/S
  - Informações
  - Proteção e Segurança
  - entre outros ...

Histórico

- Década de 40 e 50
  - Na década de 40, os primeiros computadores digitais não possuíam SO;
  - Os programas em linguagem de máquina eram entrados em cartões perfurados e as linguagens assembly foram desenvolvidas para acelerar o processo de programação
  - O primeiro SO foi desenvolvido pela *GM Laboratories* no início da década de 50 para o computador IBM 701;
  - Os sistemas operacionais da década de 50 eram do tipo lote (batch)

Histórico

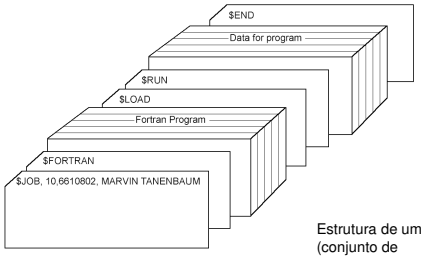


Exemplo de Sistema em batch: Primeiro, o programador traz os cartões a serem lidos; depois alguém coloca a fita onde será gravada a leitura; computador processa (maior porte); outro computador imprime o resultado

Histórico

- Década de 60
  - Os SOs da década de 60 também eram do tipo lote, mas passaram a utilizar melhor os recursos do computador, executando vários jobs de uma só vez;
  - Grandes inovações foram introduzidas através das quais pode-se obter um paralelismo entre operações de E/S e a execução de instruções pela UCP
    - interrupções e canais autônomos de E/S
    - multiprogramação (vários jobs são mantidos na memória ao mesmo tempo)
    - introdução de dispositivos de acesso aleatório
  - Surgem os SOs de **tempo repartido** (time-sharing) motivados pela necessidade de se aumentar a produtividade do programador
  - Surgem os SOs de tempo real (real time) para atender às necessidades de certas aplicações que exigem que o sistema reaja na ocorrência de certos eventos em rígidos limites de tempo.

Histórico

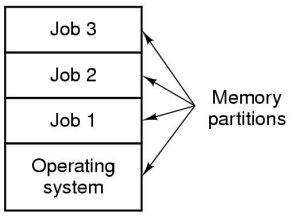


Estrutura de um job (conjunto de programas) típico

Histórico

- Década de 70
  - Predomínio dos SOs de tempo repartido que suportam processamento em lote e aplicações de tempo real;
  - O protocolo de comunicações TCP/IP tornou-se largamente utilizado e as LANs tornaram-se mais práticas e econômicas com o surgimento do padrão Ethernet desenvolvido pela Xerox.

Histórico



Compartilhamento de memória entre processos num sistema Multiprogramado

Histórico

- Década de 80
  - Desenvolvimento e popularização do modelo cliente/servidor e dos So´s de rede que provêem facilidades para o compartilhamento de recursos através da rede e incluem mecanismos de comunicação que permitem a processos executando em diferentes máquinas trocarem mensagens entre si;
  - Um computador executando um sistema operacional de rede atua de forma autônoma com relação aos demais computadores conectados em sua rede. No entanto, ele é ciente da existência dos mesmos e é capaz de se comunicar e compartilhar recursos com os mesmos.

Histórico

- Década de 90
  - **SOs Distribuídos:** Conjunto de módulos de, no mínimo, processador e memória interligados através de um subsistema de comunicação de topologia arbitrária.
    - Principal característica: descentralização do controle
    - Um SO distribuído deve se apresentar aos usuários como um sistema operacional centralizado, mas que, na realidade, tem suas funções executadas por um conjunto de máquinas independentes
    - O ponto fundamental neste tipo de sistema é o conceito de transparência: o usuário percebe este conjunto de máquinas como se fosse uma única máquina centralizada

Histórico

- Década de 90
  - A conectividade é facilitada através de padrões desenvolvidos por grupos internacionais como a International Organization for Standartization, o CCITT, Open software Fundation, X/Open e outros;
  - Adoção da filosofia de sistemas abertos;
  - Popularização da Internet.

Tipos de SO

- SO de .... :
  - MainFrames
  - Servidores
  - Multiprocessados
  - Computadores Pessoais
  - Dispositivos móveis
  - Tempo-Real
  - Sistemas Embarcados (microondas, tv ...)
  - Cartões Inteligentes

Instruções

- Surgiram com os sistemas multiprogramados para garantir que a atividade errônea ou maliciosa de um programa não cause interferência ou destruição de outro
- Introduz-se no hardware dois estágios:
  - usuário
  - supervisor
- Para executar uma instrução privilegiada, o usuário é obrigado a mudar o estado da máquina para supervisor através de instrução do tipo “chamadas ao sistema” que desvia o programa para uma rotina do SO

Chamadas ao Sistema

- Chamadas ao sistema são funções que o programador pode chamar, requerendo serviços oferecidos pelo sistema operacional. Normalmente elas envolvem o acesso a dados e recursos que usuários não podem acessar diretamente.
  - Criação de um novo processo
  - Leitura e escrita de blocos de arquivos no disco
  - Estabelecimento de uma conexão de rede com um computador remoto

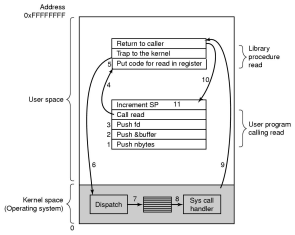
Chamadas ao Sistema

- Para realizar uma chamada ao sistema, deve-se link editar ao programa uma biblioteca do sistema.
- Quando uma chamada ao sistema é realizada, um chaveamento do modo usuário para o modo núcleo (ou kernel) é automaticamente realizada através de uma interrupção de software conhecida como **trap**
- As chamadas ao sistema provêem a interface entre um processo e o SO. Estão disponíveis em assembly, mas muitos sistemas como Unix e Windows também as disponibilizam através de linguagens de alto nível como C, C++ e Perl

Chamadas ao Sistema

- Os programas utilizam constantemente chamadas ao sistema. Por exemplo, para realizar uma simples leitura em um arquivo são necessários vários passos:

Chamadas ao Sistema



- Existe 11 passos para fazer a chamada ao sistema read(fd, buffer, nbytes)

Chamadas ao Sistema

- Para realizar a passagem de parâmetros à uma chamada ao sistema o processo normalmente utiliza uma das seguintes abordagens
  - Diretamente através de registradores;
  - Através de blocos ou tabelas em memória
  - Através da pilha de execução

Chamadas ao Sistema

- A utilização das chamadas ao sistema é normalmente escondida do programador pelos compiladores. As chamadas ao sistema podem ser classificadas nas seguintes categorias:

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

Chamadas ao Sistema

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Chamadas ao Sistema

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Chamadas ao Sistema

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Chamadas ao Sistema

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time