Introdução a Sistemas Operacionais

Objetivos

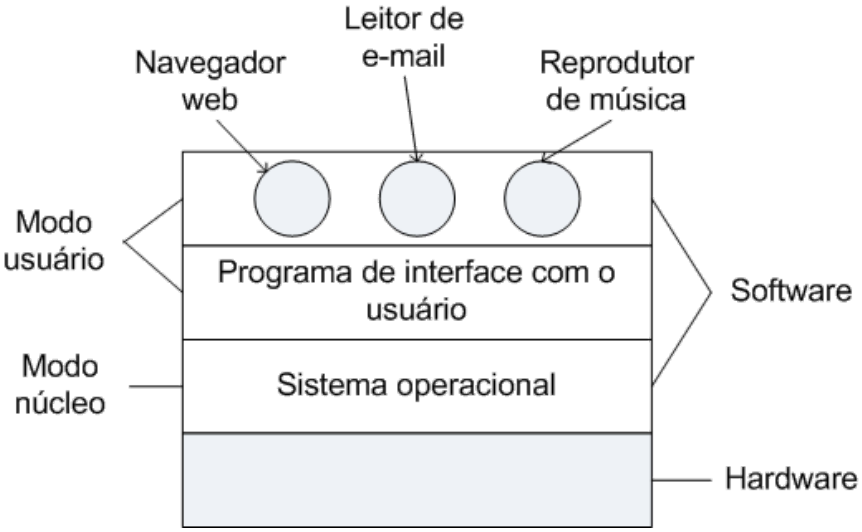
- Entender o que é um sistema operacional
- Conhecer a evolução dos sistemas ao longo da história
- Compreender a classificação de sistemas operacionais
- Aprender conceitos básicos relacionados a sistemas operacionais
- Determinar estruturas de sistemas operacionais

Introdução a Sistema Operacionais

- Sistema computacional atual

- Muitos componentes
  - Programador não entende de todos em detalhes
  - Gerenciar e utilizá-los de forma otimizada é difícil
- Sistema operacional
  - É software básico
  - Gerencia e facilita o acesso a estes componentes
- Exemplos de sistemas operacionais

- Não confundir
  - Interpretador de comando (shell)
  - Interface gráfica (Graphical User Interface - GUI)
- Panorama



- Hardware

- Software
  - Sistema operacional + aplicativos
- Modos
  - Modo núcleo ou supervisor
    - Acesso irrestrito ao hardware
    - Pode executar qualquer instrução
  - Modo usuário
    - Acesso restrito a subconjunto de intruções
      - Não afeta o controle da máquina
      - Não permite realizar E/S
- Caraterística de sistemas operacionais
  - Grandes
  - Complexos
  - Vida longa
- Curiosidades
  - Windows e Linux têm cinco milhões (5000000) de linhas
  - Inclusão de GUI, bibliotecas e aplicativos básicos aumentam em 10 a 20 vezes
  - Não é fácil descartar e reescrever um sistema operacional

O que é um sistema operacional?

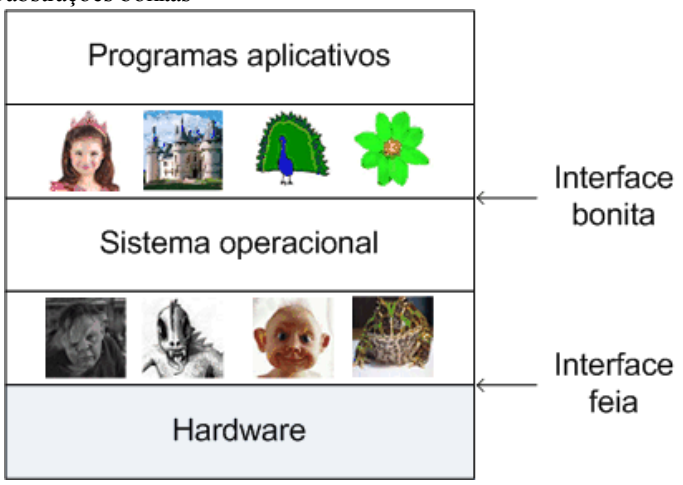
- Software que realiza duas funções
  - Fornece recursos abstratos de forma clara para programadores de aplicativos
  - Gerencia recursos de hardware

O sistemas operacional como uma máquina estendida

- Arquitetura em nível de linguagem de máquina
  - Primitiva e de difícil programação
- Exemplo
  - Controlador de disquete NEC PD765

    - 16 Comandos de 1 a 9 bytes
      - Iniciação, sinalização, reiniciação e recalibração
      - Leitura e escrita de dados
      - Movimentação do braço
      - Formatação de trilhas
    - 13 parâmetros em 9 bytes
      - Endereço de bloco de dados
      - Números de setores por trilhas
      - Modo de gravação
    - 23 campos de status e erros em 7 bytes
    - Ligar e desligar motor

- Abstração de um disco
  - Coleção de arquivos com nomes
  - Abertura
  - Leitura e/ou escrita
  - Fechamento
- Sistemas operacionais são baseados em abstrações
  - Ocultam detalhes do hardware
  - Oferecem abstrações precisas, claras, elegantes e coerentes
  - Transforma hardware feio em abstrações bonitas

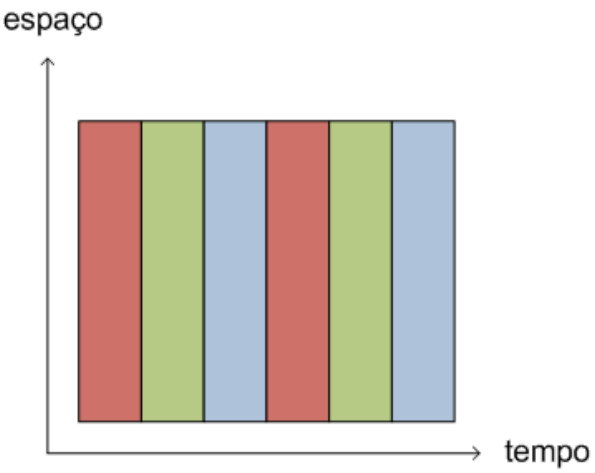


- Uso de abstrações
  - Windows

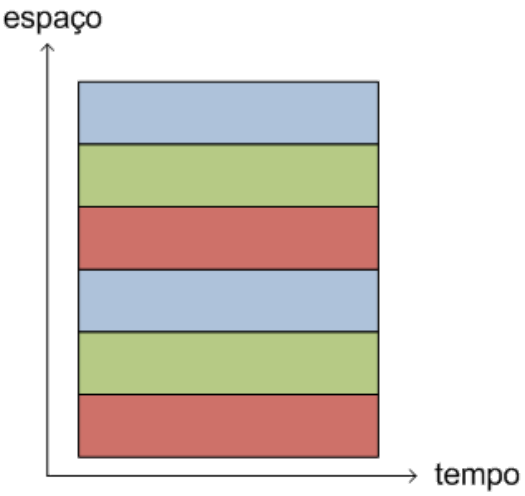
  - Linux

O sistema operacional como um gerenciador de recursos

- Problema na execução de múltiplos programas
  - Três programas enviam páginas para impressão simultaneamente?
    - Primeiras linhas do programa 1
    - Algumas linhas do programa 2
    - Outras linhas do programa 3
- Solução com sistema operacional
  - Armazenamento de páginas em disco
  - Envio sequencial para impressão
- Sistema operacional gerencia e protege recursos
  - Mantém o controle sobre usuários
  - Garante requisição de recursos
  - Media conflitos entre requisições
- Compartilhamento ou multiplexação
  - Temporal
    - Programas aguardam a vez de utilizar recurso
    - Ordem e tempo de uso determinado pelo sistema operacional
    - Exemplos
      - Processador
      - Impressora



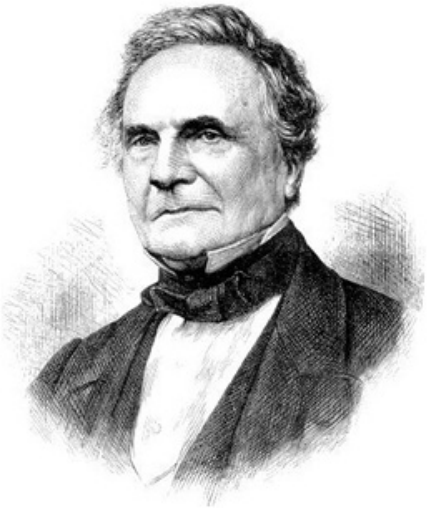
- Espacial
  - Recurso dividido para os programas
  - Utilização em paralelo sob controle do sistema operacional
  - Exemplo
    - Memória
    - Discos



História dos sistemas operacionais

- Evolução dos computadores ↔ Evolução dos sistemas operacionais
- A máquina analítica

  - Construída por Charles P. Babbage (1792 - 1871)



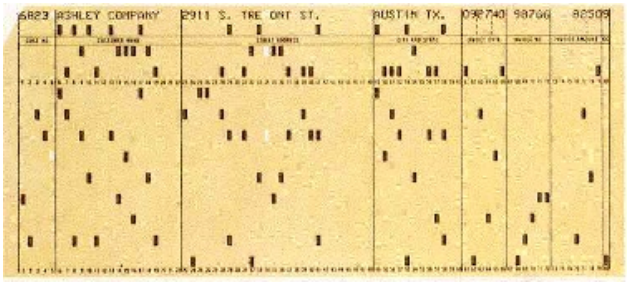
- Programável e totalmente mecânica
- Sem sistema operacional
- Gastou sua fortuna e não conseguiu implementá-la totalmente
  - Necessitava de engrenagens muito precisas para época
- Auxiliado por Ada Byron King - Condessa de Lovelace (1815 – 1852)
  - Filha do Lord Byron
  - Primeira programadora do mundo

A primeira geração (1945 - 1955) - relés e válvulas

- Construção de computadores motivadas pela Segunda Guerra Mundial

Ano	Nome	Local	Construtor(es)
1941	ABC	Iowa, EUA	John Atanasoff e Clifford Berry
1941	Z3	Berlim, Alemanha	Konrad Zuse
1944	Colossus	Bletchley Park, Inglaterra	Tommy Flowers
1944	Mark	Harvard, EUA	Howard Aiken
1946	ENIAC	Pensilvânia, EUA	William Mauchley e J. Presper Eckert

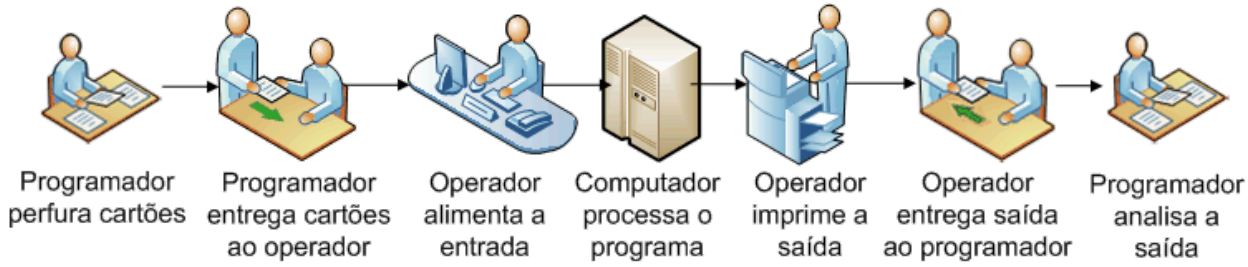
- Características
  - Levavam segundos para executar cálculos simples
  - Tarefas realizadas pelo grupo de pesquisa
    - Projeto
    - Construção
    - Programação
    - Operação
    - Manutenção
  - Programação através de fios e plugs
    - Evolução para cartões perfurados



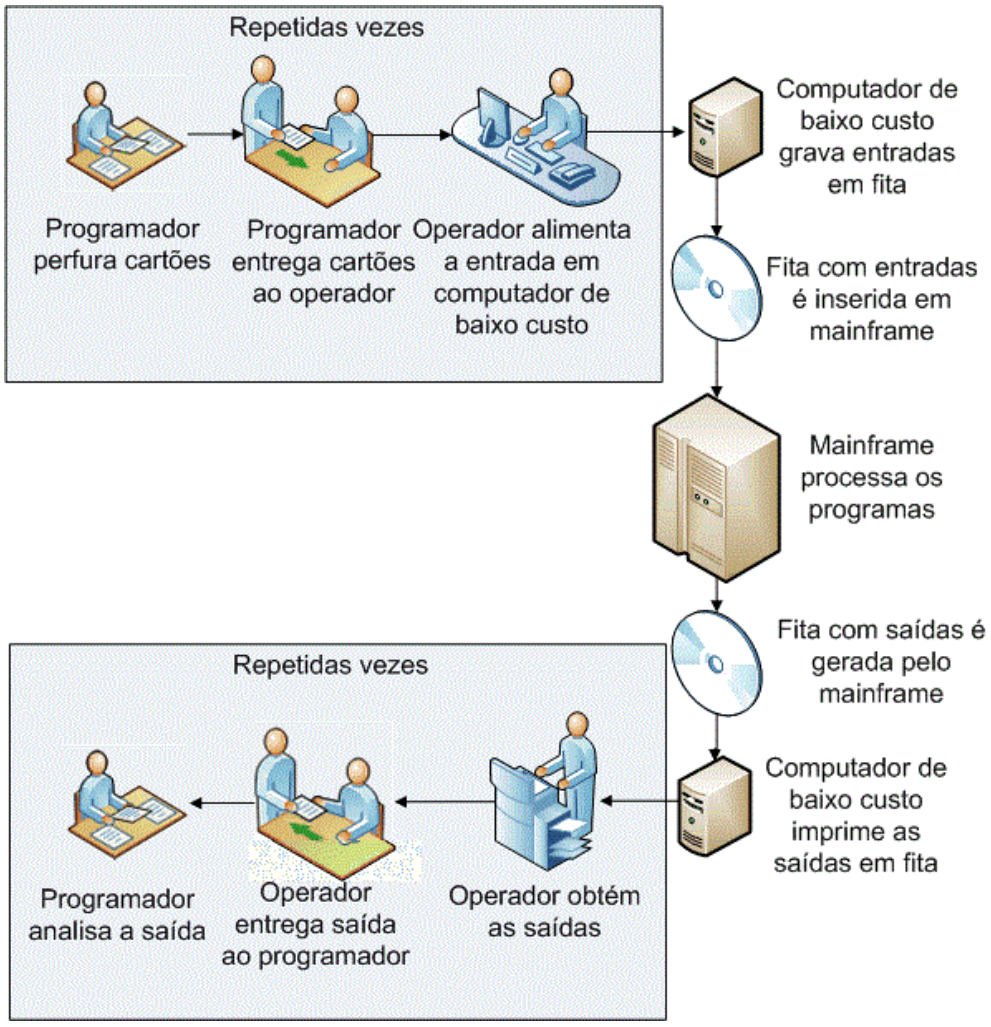
- Falhas frequentes devido a queima de componentes

A segunda geração (1955 - 1965) - transistores e sistemas em lotes (batch)

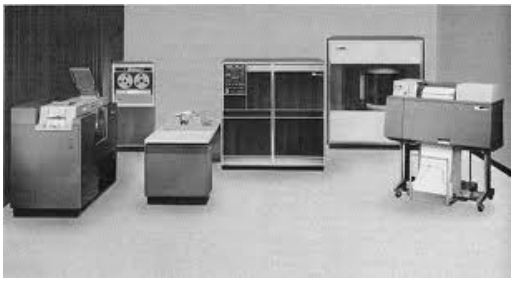
- Utilização do transistor
  - Velocidade e confiabilidade
- Separação de atribuições
  - Projetistas
  - Fabricantes
  - Programadores
  - Técnicos em manutenção
- Computadores de grande porte/mainframes
  - Custo muito alto
  - Acessível a grandes corporações, agências governamentais e universidades
- Ineficiente na utilização



- Processamento em lotes



- Combinação comum na época
  - IBM 1401 para geração das fitas



  - IBM 7094 para execução dos programas

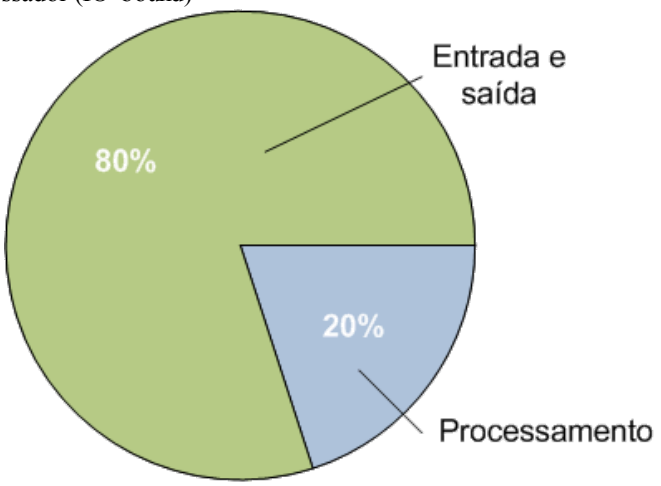
- Primeiros sistemas operacionais
  - Gerenciamento da execução dos programas em fita
  - Exemplos
    - Fortran Monitor System (FMS)
    - IBSYS para IBM 7094

A terceira geração (1965 - 1980) - CIs e multiprogramação

- Utilização de circuitos integrados
  - Mais velocidade e menos custo
- Duas linhas distintas
  - Cálculos científicos orientado a palavras
  - Comerciais orientados a caracteres
- Problemas
  - Manutenção de duas linhas distintas
  - Difícil escalabilidade
- Solução da IBM
  - Lançar o System/360
    - Máquinas de pequeno a grande porte
    - Mesma arquitetura em todos os modelos
    - Diferenças no preço e desempenho
    - Voltadas para computação científica e comercial

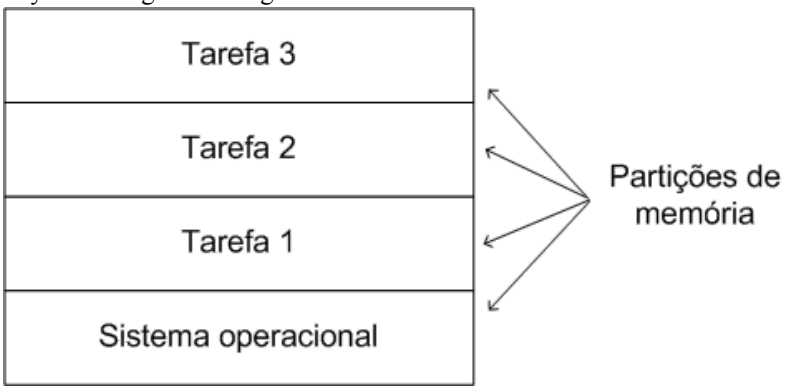
- Adoção do sistema operacional OS/360
  - Execução em qualquer modelo System/360
    - Alta diversidade de configurações
  - Tornou-se complexo
    - Milhões de linha de código Assembly

        - Milhares de erros
            - Atualizações corrigiam uns e inseriam outros
- Problema
    - Cálculos científicos
        - Entrada e saída é pouco frequente
        - Muito processamento (CPU-bound)
    - Processamento comercial
        - Ocorre o inverso
        - Ociosidade do processador (IO-bound)



Processamento de dados comerciais

- Multiprogramação
    - Divisão da memória em partições
    - Alocação de tarefas distintas em cada partição
    - Processador pode executar tarefa enquanto outro espera E/S
    - Hardware especial no System/360 garantia integridade entre tarefas



- Problema
    - Afastamento dos programadores da operação da máquina
    - Muito tempo para correção de erros simples
- Timesharing ou tempo compartilhado
    - Usuários se conectavam através de terminal
    - Serviço interativo aos usuários
    - Processamento de tarefas em background
- MULTICS
    - Multiplexed information and computing service

Massachusetts Institute of Technology Bell Laboratories

    - Centenas de usuários em tempo compartilhado
    - Problemas
        - Codificação em PL/I
        - Bell Labs saiu do projeto
        - General Eletric abandonou o ramo da computação
    - Vendido para Honeywell

Honeywell

    - Utilizado pela General Motors, Ford e Agência de Segurança Nacional até 1990

    - Serviu de referência para os sistemas operacionais subsequentes
- Minicomputadores
    - Computadores com menor poder de processamento
        - Preço dezena de vezes menor que um mainframe
    - Série PDP da DEC
        - Iniciou com o PDP-1 em 1961
            - Memória de 4k com 18 bits
            - Preço $120.000,00 correspondia a 5% do IBM 7094

        - Foi até o PDP-11

- UNIX
    - Desenvolvido por Ken Thompson e Dennis Ritchie no Bell Labs

    - Versão simplificada e monousuário do MULTICS
    - Escrito para PDP-7
    - Popular no ambiente acadêmico, agências governamentais e empresas
    - Diversas versões
        - System V da AT&T
        - BSD - Berkeley Software Distribution da Universidade da Califórnia em Berkeley
        - Padronização POSIX - Portable Operating System Interface do IEEE
- MINIX
    - Clone do UNIX com fins educionais
    - Suporte ao POSIX
    - Desenvolvido por Andrew Tanebaum em 1987

- Linux
    - Variação do MINIX

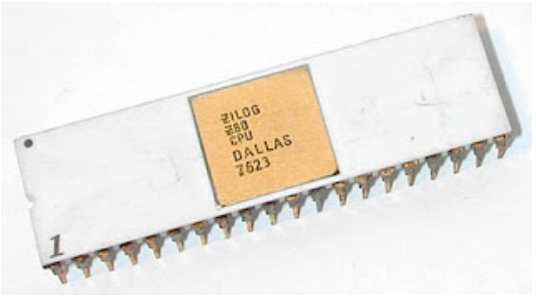
- Criado pelo estudante finlandês Linus Torvalds

A quarta geração (1980 - presente) - computadores pessoais

- Lançamento dos microcomputadores
  - Utilização de circuitos integrados LSI - Large Scale Integration
- CP/M
  - Control Program for Microcomputer
  - Sistema operacional para Intel 8080 de 8 bits

  - Criado por Gary Kildall

  - Aperfeiçoado pela Digital Research para Zilog Z80 e outros



  - Dominante durante 5 anos
- MS-DOS
  - IBM projetou PC em 1980
  - Negociações entre IBM e Digital Research não se concretizaram
  - Microsoft comprou DOS da Seattle Computer e licenciou para IBM
  - Integração de conceitos do UNIX
    - XENIX era o UNIX da Microsoft
- Interfaces gráficas
  - GUI - Graphical User Interface
  - Propostas por Doug Engelbart
  - Implementadas nos laboratórios da Xerox

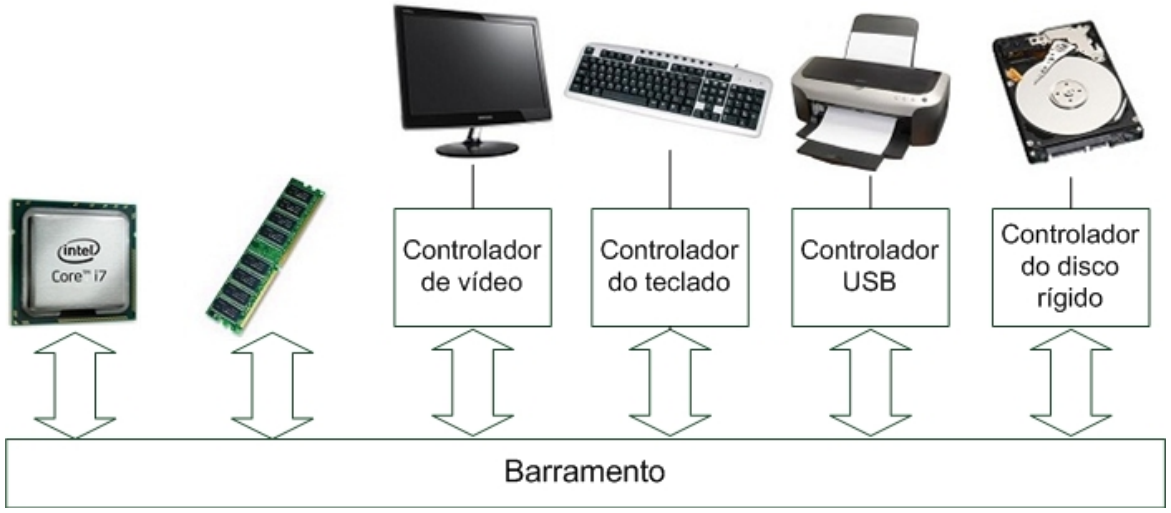
- Copiados por Steve Jobs no Apple Lisa
  - Fracasso comercial

- Melhorada no Macintosh

- Windows
  - Interface gráfica do DOS entre 1985 e 1995
  - Evolução do Windows 95 até Windows 8

Revisão sobre hardware de computadores

- Sistema operacional está completamente relacionado com o hardware
- Abstração de computador pessoal



Processadores

- Cérebro do computador
- Busca instruções na memória e as executa
- Cada família de processadores tem seu conjunto de instruções
  - x86 é incompatível com SPARC
- Registradores
  - Gerais
    - Armazenam computação temporária
  - Contador de programa - Program counter (PC)
    - Endereço de memória da próxima instrução
  - Ponteiro da pilha - Stack Pointer (SP)
    - Topo da pilha na memória
  - Registrador de estado - Program Status Word (PSW)
    - Palavra de estado do programa
    - Controla modo de execução
- Tarefa do sistema operacional
  - Interrupção de execução
    - Salvamento dos valores dos registradores
  - Retomada de execução
    - Valores dos registradores restaurados
- Arquiteturas com pipeline e superescalares
  - Complicam tarefa do sistema operacional
- Trap
  - Chamada ao sistema operacional
  - Chaveamento do modo usuário para modo núcleo
  - Execução do serviço
  - Retorno ao programa solicitante

Memória

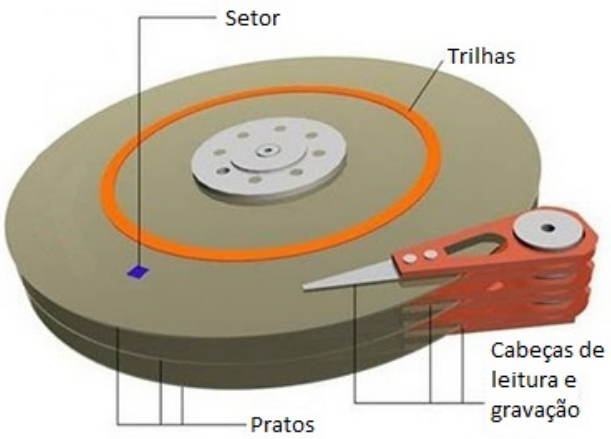
- Impossível obter memória tão rápida quanto o processador
- Hierarquia de memória

- Registradores
  - Mesmo material do processador
  - Atraso muito pequeno
- Cache
  - Divisão em níveis

    - Interna (L1) e próxima (L2) ao processador
    - Conceito utilizado em sistemas operacionais
        - Recurso divisível
        - Utilização não uniforme
        - Criar atalhos para mais utilizados
    - Termos
        - Cache hit
            - Endereço buscado está na cache
        - Cache miss
            - Endereço desejado não está na cache
- Memória principal
    - RAM
        - Armazenamento de informações para processamento
    - ROM
        - Firmware de controle de dispositivos

Discos

- Baseado em magnetismo e mecânica
- Cabeças de leitura e gravação
    - Movem sobre pratos metálicos
- Pratos
    - Sobrepostos
    - Giram em torno de eixo
- Setor
    - Unidade de armazenamento mínimo
- Trilha
    - Conjunto de setores de região circular
- Cilindro
    - Conjunto de trilhas justapostas em pratos diferentes



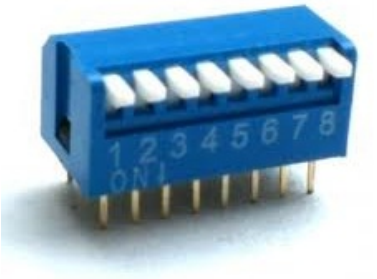
Dispositivos de entrada e saída

- Dispositivos associados a controladores
    - Recebem solicitações de leitura e escrita do sistema operacional
- Driver de dispositivo
    - Programas que fazem ligação entre controladores e sistema operacional
    - Executados em modo núcleo
    - Drivers USB e FireWire carregados dinamicamente
- Acesso pelo processador
    - Mapeamento de endereços
    - Instruções especiais
- Modos de acesso
    - Espera ocupada
        - Processador aguarda transferência
    - Interrupção
        - Dispositivo avisa término
        - Esquema de prioridades
    - Acesso direto a memória - Direct Memory Access (DMA)

        - Sem intermédio do processador durante transferência

Barramento

- Conjunto de linhas de comunição que permitem interligação entre dispositivos
- Parâmetros
    - Largura em bytes
    - Velocidade em MB/s
- Exemplos
    - ISA (Industry Standart Interface)
    - PCI (Peripheral Component Interconnect)
    - PCI Express
    - IDE (Integrated Driver Eletronics)
    - USB (Universal Serial Bus)
    - SCSI (Small Computer System Interface)
    - FireWire
- Problema
    - Dispositivos com interruções e endereços fixos
        - Possibilidade de conflito
        - Configuração manual



- Solução com Plug and Play
    - Atribuição automática de endereços e interrupções

Inicializando o computador

- Sistema Básico de Entrada e Saída - Basic Input Output System (BIOS)
    - Armazenado na placa mãe
    - Rotinas básicas
        - Leitura do teclado
        - Escrita na tela
        - Entrada e saída no disco
    - Detecta, testa e configura dispositivos ligados nos barramentos
        - Power On Self Test (POST)
    - Determina dispositivo de inicialização

O zoológico de sistemas operacionais

- Computadores de grande porte

- Servidores

- Computadores pessoais

- Dispositivos móveis

- Sistemas embarcados

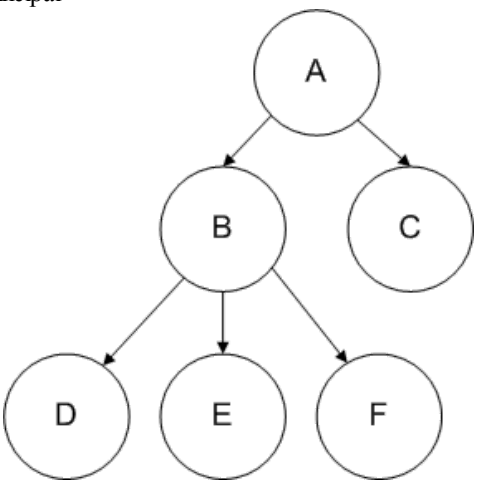
- Tempo real

Conceitos sobre sistemas operacionais

- Conceitos e abstrações comuns

Processos

- Programa em execução
- Espaço de endereçamento ou imagem do núcleo
  - Posições de memória
    - Programa + dados + pilha
- Tabela de processos
  - Registradores (PC e SP)
  - Arquivos abertos
  - Sinais de alarme
  - Processos filhos
- Processos filhos
  - Processos filhos criados de processo principal



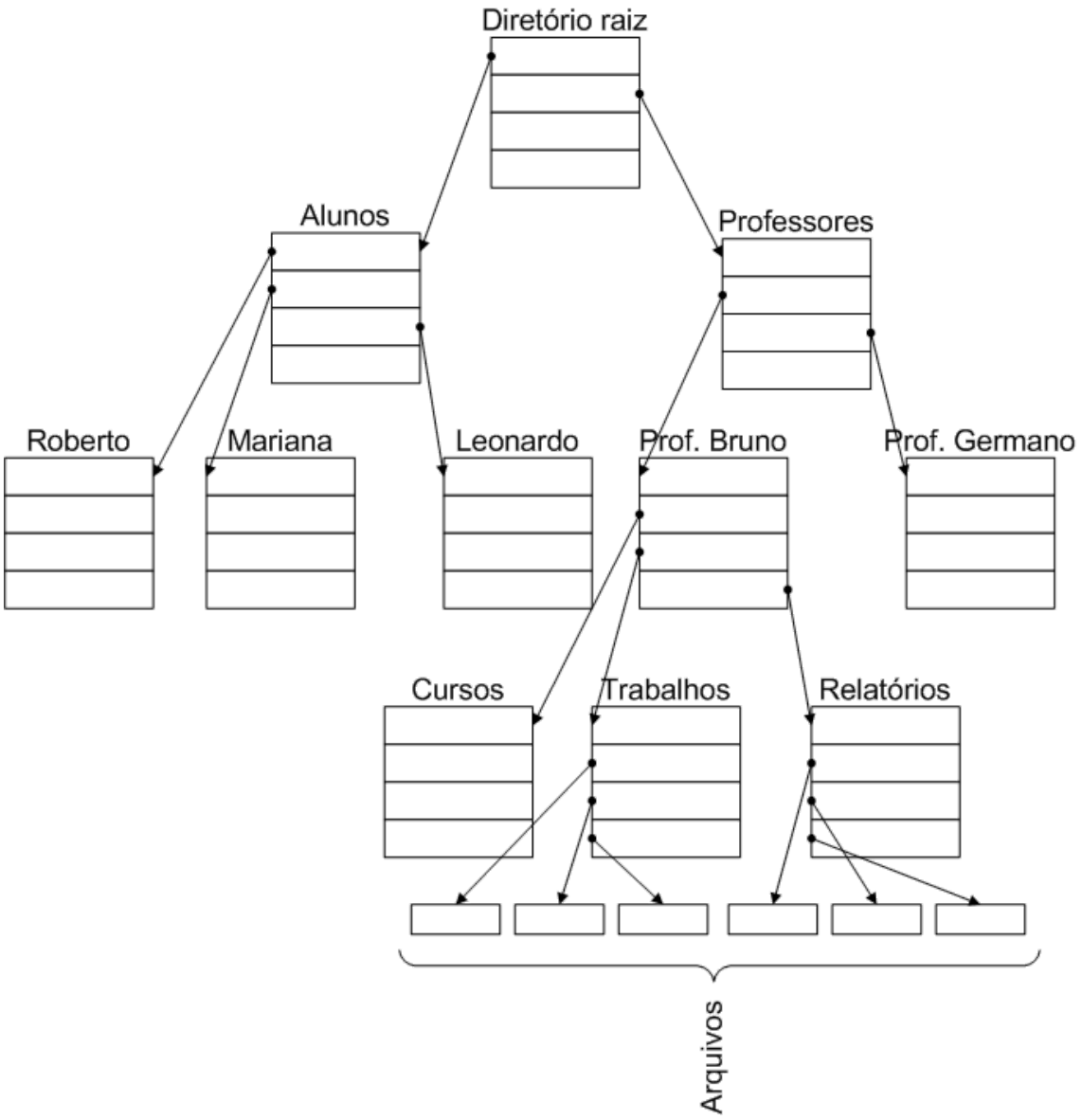
- Comunicação entre processos
  - Processos relacionados cooperando para realização de atividade
  - Troca de mensagens entre processos
- Sinal de alarme
  - Aviso do sistema operacional
- Identificação do usuário - User Identification (UID)
  - Atribuição realizada no momento da execução
  - Processos filhos têm mesma identificação do principal
- Grupo de identificação - Group Identification (GID)
  - Grupos de usuários

Espaços de endereçamento

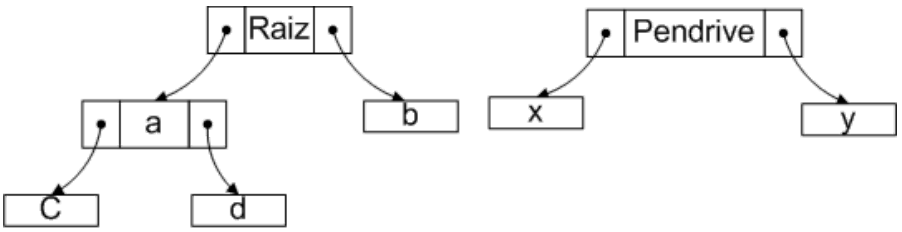
- Gerenciamento da memória
  - Não ler/escrever na região de memória de outro processo
  - Permitir endereçamento além do tamanho da memória física

Arquivos

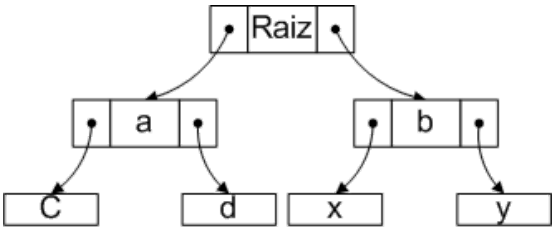
- Agrupamento de dados ou programa
- Diretório ou pasta
  - Analogia com armazenamento de cartões perfurados
- Diretório raiz
  - Topo da hierarquia
- Caminho ou path name
  - Sequência de diretórios até chegar ao alvo
  - Separador
    - UNIX é "/"
    - Windows é "\"
  - Exemplos
    - /Alunos/Leonardo
    - /Professores/Prof. Bruno/Relatórios



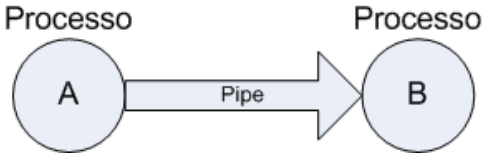
- Tarefa do sistema operacional
  - Independência de dispositivo
  - Ocultar peculiaridades
  - Fornecer interface comum
- Descritor de arquivo
  - Número identificador de arquivo
  - Fornecido no momento da abertura
- Montagem de sistema de arquivo
  - Comando mount
  - Agregação de subsistema ao sistema principal
    - Drives de CD, DVD e Bluray
    - Pendrives
  - Antes



- Depois



- Pipe
  - Pseudoarquivo utilizado para conectar dois processos



Entrada e saída

- Subsistemas
  - Comuns
  - Específicos (drivers)

Segurança

- Proteção de arquivos
  - Leitura autorizada pelo identificador do usuário
- Bits rwx
  - Esquema UNIX
  - Significado
    - Leitura (read)
    - Escrita (write)
    - Execução (execute)

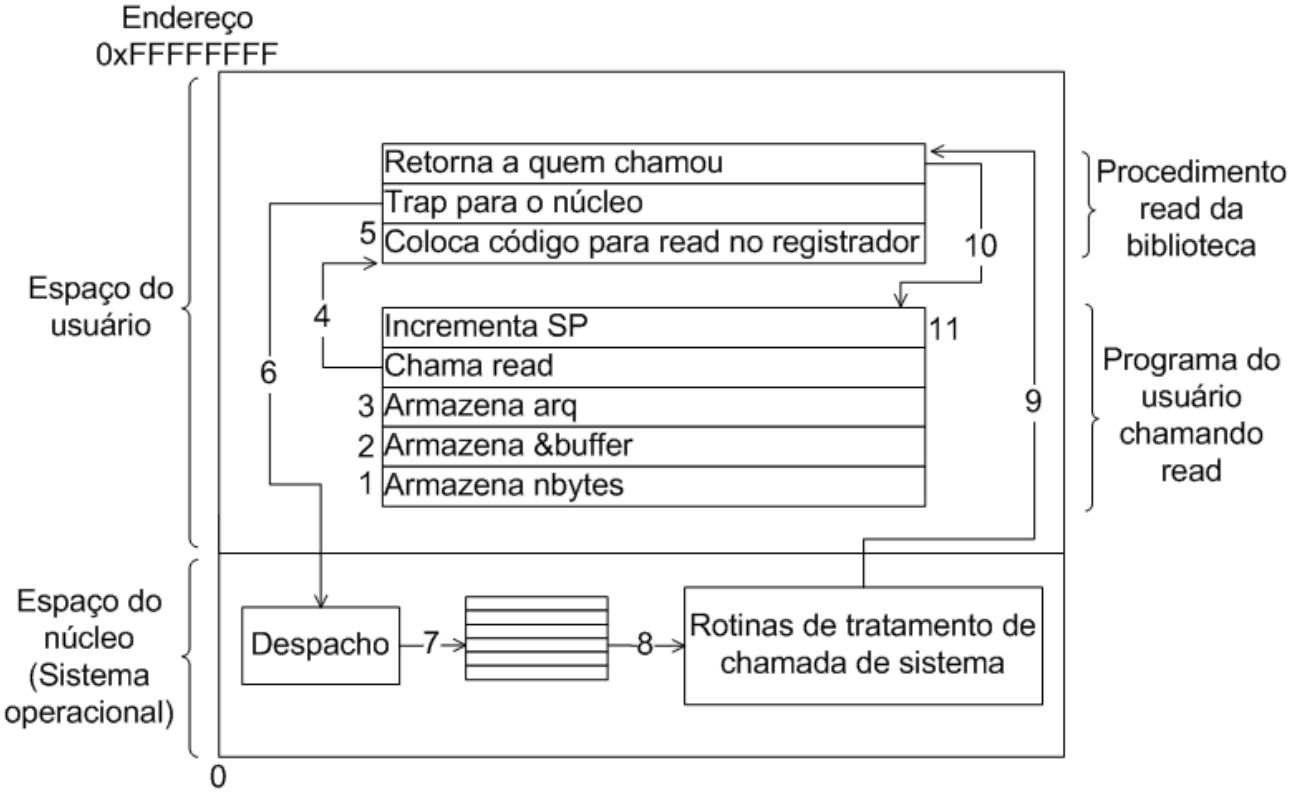
    Ordem
    - Proprietário
    - Grupo
    - Outros
  - Exemplo
    - rwxr-x--x
    - Proprietário pode ler, escrever e executar
    - Membro do grupo podem ler ou executar
    - Outros pode executar

Interpretador de comandos (shell)

- Faz uso intensivo do sistema operacional
- Shells do UNIX
  - sh
  - csh
  - ksh
  - bash
- Conexão através de pipe
  - Exemplo
    - cat arq1 arq2 arq3 | sort > /dev/lp
    - Imprime conteúdo de três arquivos
    - Passa para o processo de ordenação
    - Repassa para a impressora

Chamadas de sistema

- Gerenciamento de recursos
  - Transparente para o usuário
- Disposição de abstrações
  - Interface de chamadas disponíveis
    - Construídas em Assembly
    - Disponiveis em C
- Exemplo
  - Leitura de arquivo com comando read
    - contador = read(arq, buffer, nbytes)
      - arq é o nome do arquivo
      - buffer é referência para array onde dados serão armazenados
      - nbytes é número máximo de bytes a serem lidos
      - contador é o número de bytes realmente lidos



  - Passos
    1. Empilha quantidade de bytes a ler
    2. Empilha ponteiro para buffer
    3. Empilha nome do arquivo
    4. Chama read da biblioteca
    5. Armazena código da chamada read do sistema operacional em registrador
    6. Transfere controle para sistema operacional
    7. Encaminha através de tabela de rotinas
    8. Executa chamada solicitada
    9. Retorna a chamada da biblioteca
    10. Retorna a chamada do programa
    11. Limpa a pilha

Chamadas de sistema para gerenciamento de processos

- pid = fork()
  - Gera cópia exata do processo original
    - Descritores de arquivos e registradores
  - Processos seguem caminhos separados
  - Variáveis independentes com valores copiados
  - Retorna PID no processo pai e zero no processo filho
- pid = waitpid(pid, &statloc, options)
  - Espera fim de processo filho

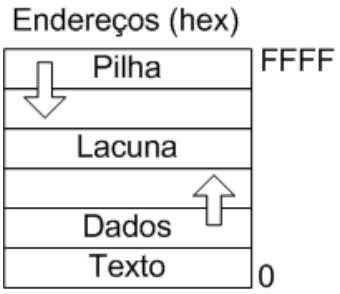
    - Especificação através do PID ou qualquer processo filho (-1)
    - Resultado da execução (término normal ou anormal) retorna em statloc
    - Outras opções em options
- s = execve(name, argv, envirop)
    - Substitui processo em execução por programa com nome name
    - Argumentos repassados através do ponteiro argv
    - Conjunto de variáveis de ambiente são repassadas através de envirop
    - Processo original não recebe retorno (porque será substituído) ou -1 na ocorrência de falhas
- exit(status)
    - Encerra execução de processo
    - Estado de saída é retorna em status
- Exemplo de intepretador

```
#define TRUE 1
while (TRUE) //repita para sempre
{   type_prompt(); //mostra prompt na tela
    read_command(command, parameters); //lê entrada do terminal
    if (command == "exit") //verifica se foi digitado "exit"
    {   exit(0); //finaliza execução do processo
    } else if (fork() != 0)
    {   //código do processo pai
        waitpid(-1, &status, 0); //aguarda processo filho acabar
    } else
    {   //código do processo filho<br>
        execve(command, parameters, 0); //executa comando<br>
    }
}
```

- Resumo pid = waitpid(pid, &statloc, options)

Chamada	Descrição
pid = fork()	Cria um processo filho idêntico ao pai
Espera que um processo seja concluído	
s = execve(name, argv, envirop)	Substitui a imagem do núcleo de um processo
exit(status)	Conclui a execução do processo e devolve status

- Segmentos de memória no UNIX
    - Pilha
        - Cresce para baixo
    - Dados
        - Variáveis
        - Cresce para cima
    - Texto
        - Código do programa



Chamadas de sistema para gerenciamento de arquivos

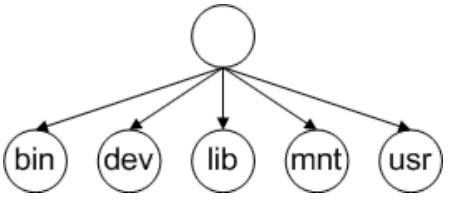
- fd = open(file, how, ...)
    - Abre um arquivo
    - Caminho absoluto ou relativo é especificado em file
    - Modo de abertura é definido por how
        - O_RDONLY para leitura
        - O_WRONLY para escrita
        - O_RDWR para leitura/escrita
        - O_CREAT para criação
    - Retorna descritor de arquivo

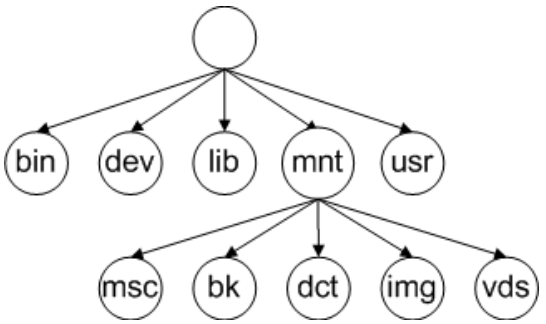
- s = close(fd)
  - Fecha arquivo definido no descritor fd
  - Retorna descritor disponível para próxima abertura
- n = read(fd, buffer, nbytes)
  - Leitura de arquivo de acordo com parâmetros vistos anteriormente
- n = write(fd, buffer, nbytes)
  - Escrita de arquivo com parâmetros idênticos a chamada read
- position = lseek(fd, offset, whence)
  - Permite leitura e escrita de forma não sequencial
  - Altera posição do cursor do arquivo fd
  - Deslocamento é definido em offset
  - Relação do deslocamento (absoluto ou relativo) é indicada em whence
- s = stat(name, &buf)
  - Fornece informações sobre arquivo
    - Tipo do arquivo
    - Tamanho
    - Última modificação
  - Nome do arquivo definido em name
  - Buffer para armazenamento de informações em buf
- Resumo

Chamada	Descrição
fd = open(file, how, ...)	Abre um arquivo para leitura, escrita ou ambos
s = close(fd)	Fecha um arquivo aberto
n = read(fd, buffer, nbytes)	Lê dados a partir de um arquivo em um buffer
n = write(fd, buffer, nbytes)	Escreve dados a partir de um buffer em arquivo
position = lseek(fd, offset, whence)	Move o ponteiro do arquivo
s = stat(name, &buf)	Obtém informações sobre um arquivo

Chamadas de sistema para gerenciamento de diretórios

- link(path1, path2)
  - Cria ligação do arquivo path2 para i-node de path1
  - i-node contém informações sobre arquivo
    - Incluíndo identificador único
  - Único arquivo com nomes e locais diferentes
- unlink(path)
  - Remove ligação do arquivo path com i-node
  - Isolamento de i-node provoca exclusão
- mount(dev, path)
  - Agrega dispostivo dev ao caminho path
  - Dispositivo passa a fazer parte da hierarquia de diretórios
  - Aplicações
    - Leitores ópticos
    - Discos rígidos externos
    - Pendrives





- unmount(path)
  - Remove dispositivo montado em path da hierarquia de diretórios
- Resumo

Outras chamadas de sistema

- chdir(path)
  - Altera diretório atual de trabalho para path
  - Evita digitar nomes de arquivos absolutos
- chmod(file, perm)
  - Possibilita alteração de permissão de arquivo file para valor perm
  - Padrão
    - rwx para proprietário, grupo e outros
- kill(pid)
  - Encerra processo com identificador pid
- s = time()
  - Retorna quantidade de segundos desde 00:00h de 01/01/1970
  - Limitado ao valor 2^{32}-1 em sistemas 32 bits
    - Bug do ano 2106

A API Win32 do Windows

- UNIX
  - Processamento e chamadas ao sistema operacional
- Windows
  - Dirigido a eventos

UNIX	Windows	Descrição
fork	CreateProcess	Cria um novo processo
waitpid	WaitForSingleObject	Espera que um processo termine
execve	(nenhuma)	CreateProcess = fork + execve
exit	ExitProcess	Conclui execução
open	CreateFile	Cria arquivo ou abre existente
close	CloseHandle	Fecha um arquivo
read	ReadFile	Lê dados a partir de um arquivo
write	WriteFile	Escreve dados em um arquivo
lseek	SetFilePointer	Move o ponteiro do arquivo
stat	GetFileAttributesEx	Obtém atributos de arquivo
mkdir	CreateDirectory	Cria um novo diretório
rmdir	RemoveDirectory	Remove um diretório vazio
link	(nenhuma)	Win32 não dá suporte a link
unlink	DeleteFile	Destrói um arquivo existente
mount	(nenhuma)	Win32 não dá suporte a mount
unmount	(nenhuma)	Win32 não dá suporte a unmount
chdir	SetCurrentDirectory	Altera diretório de trabalho atual
chmod	(nenhuma)	Win32 não dá suporte a segurança (NT suporta)
kill	(nenhuma)	Win32 não dá suporte a sinais

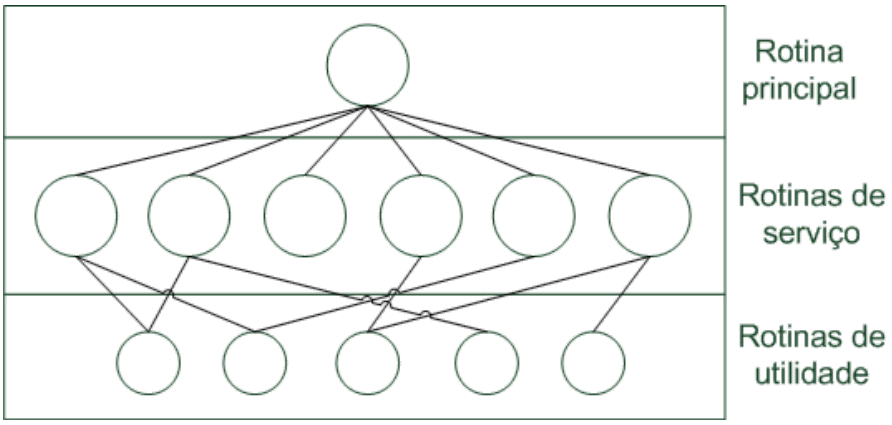
time	GetLocalTime	Obtém o tempo atual

Estruturas de sistemas operacionais

- Interface descreve sistema operacional externamente
- Estrutura indica sua organização interna

Sistemas monolíticos

- Organização mais comum
- Executado como único programa em modo núcleo
- Liberdade de chamada entre as rotinas
- Organização
  - Rotina principal
  - Rotinas de serviço
  - Rotinas de utilidade



Sistemas de camadas

- Generalização de sistema monolítico
- Cada camada é construída sobre a anterior
- Pioneiro no Technische Hogeschool Eindhoven (THE)
  - Proposto por Dijkstra
  - Nível de projeto

Camada	Função
5	Operador
4	Programas de usuário
3	Gerenciamento de entrada e saída
2	Comunição operador-processo
1	Memória e gerenciamento de tambor
0	Alocação de processo e multiprogramação

- Utilizado no MULTICS
  - Ligação entre camadas através de chamadas ao sistema operacional

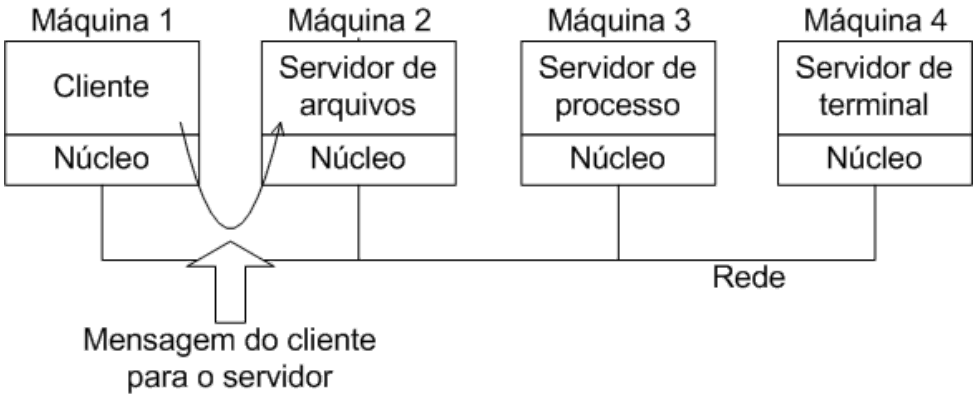
Micronúcleo

- Sistema em camada com camada especial
- Apenas micronúcleo é executado em modo núcleo
  - Restante executado com privilégio de usuário
- Alta confiabilidade
  - Falhas em módulos não comprometem micronúcleo
- Aplicações
  - Tempo real
  - Industriais
  - Aviação
  - Militares
- Exemplos

- Integrity
- K42
- L4
- PikeOS
- QNX
- Symbian
- MINIX
  - 3200 linhas em C
  - 800 linhas em Assembler
  - 35 chamadas ao núcleo

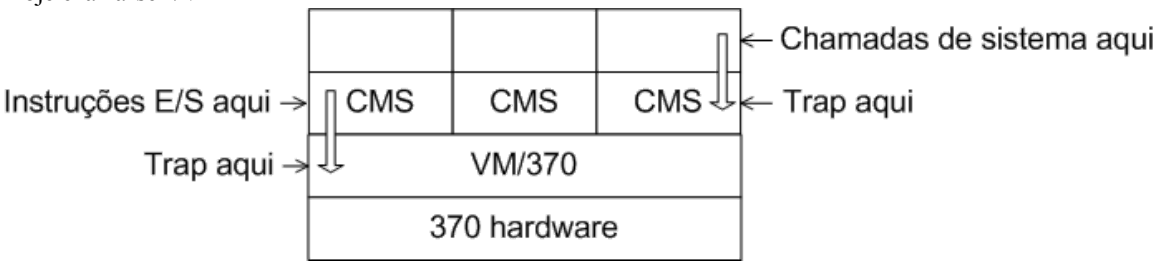
Modelo cliente-servidor

- Servidores
  - Prestam serviços
- Clientes
  - Utilizam serviços
- Modelo aplicável a única ou várias máquinas
- Típico na web



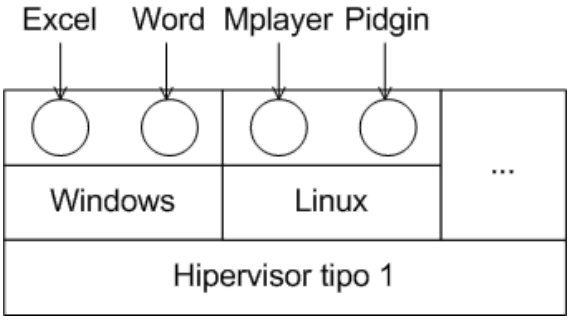
Máquinas virtuais

- Surgimento
  - Insucesso do TSS/360
    - Sistema operacional de tempo compartilhado p/ IBM 360
    - Lançado tardiamente
    - Grande e lento
  - Oportunidade para o VM/370
    - Inicialmente Control Program/Cambridge Monitor System (CP/CMS)
    - Monitor de máquina virtual
      - Fornecia cópia do hardware
    - Processamento em lotes e interativo ao mesmo tempo
    - Hoje chama-se z/VM

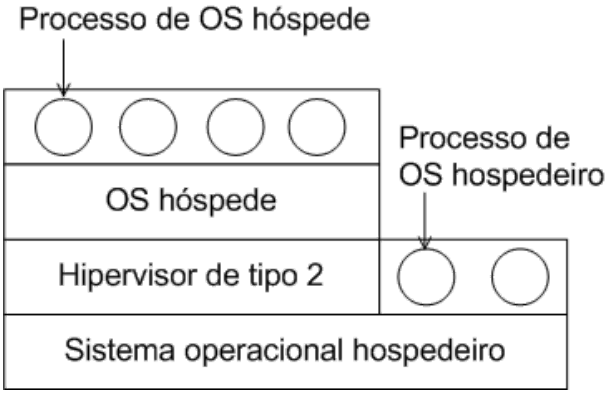


- Panorama atual
  - Migração de servidores de serviços empresariais para única máquina
    - Email
    - Web
    - FTP
  - Serviços de hospedagem
    - Hospedagem dedicada com custo de compartilhada
  - Usuário doméstico
    - Windows e Linux no mesmo equipamento

- Estratégias
  - Hipervisor tipo1
    - Não tem suporte no Pentium



  - Hipervisor tipo2
    - Presença de hospedeiro e hóspede
    - Permitiu surgimento do VMware



- A máquina virtual Java
  - Execução de programa
  - Código compilado para máquina virtual Java (JVM)
  - Interpretadores para diversos sistemas operacionais

Exonúcleo

- Divisão de recurso entre máquinas virtuais
- Exemplo
  - Blocos do 0 ao 1023 para máquina um
  - Blocos do 1024 ao 2047 para máquina dois
- Vantagem
  - Elimina mapeamento entre posições utilizadas e reais

Referências bibliográficas

- TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 3 ed. cap. 1 (Introdução).