

Sistemas Operacionais

Aula 01 - Introdução

Prof. Samuel Souza Brito

Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Aplicadas – ICEA
Departamento de Computação e Sistemas – DECSI

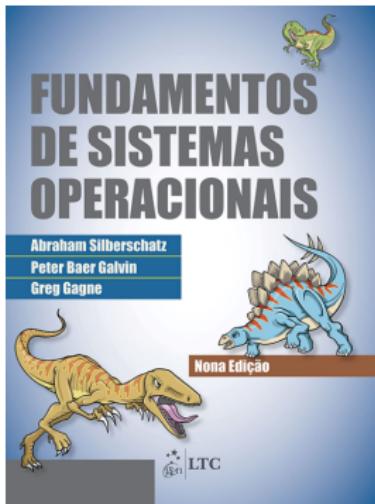


UFOP

João Monlevade-MG
2024/2

Referencial Teórico

- O material aqui apresentado (exceto a seção “História dos Sistemas Operacionais”) é baseado no Capítulo 1 do livro:
 - Silberschatz, A.; Galvin, P. B.; Gagne, G., Fundamentos de Sistemas Operacionais, Editora LTC, 9^a edição, 2015.



Introdução

O que é um Sistema Operacional?

O que é um Sistema Operacional?

- Programa que gerencia o hardware do computador.
- Oferece uma base para os programas aplicativos e atua como intermediário entre o usuário e o hardware.

Por que estudar Sistemas Operacionais?

- Entender como os computadores funcionam.
 - Melhor utilização.
- Conectam hardware e software.
- Entendimento dos SOs revelam limitações e pontos fortes.

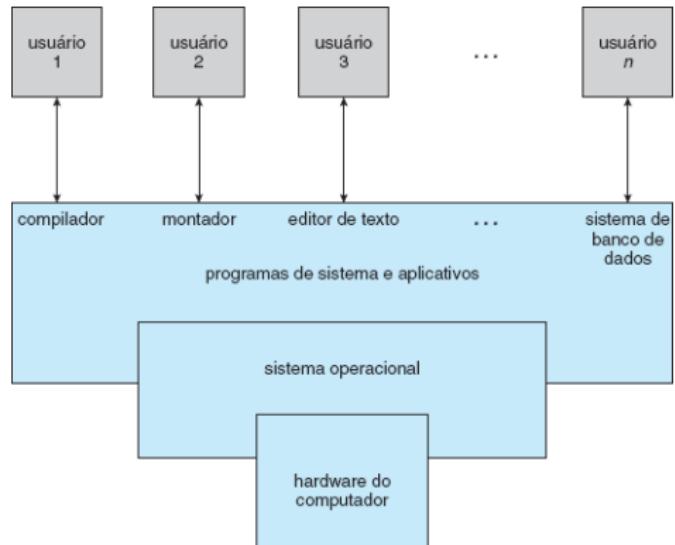
Por que estudar Sistemas Operacionais?

- Combinam conceitos de praticamente todas as áreas da computação:
 - Hardware
 - Algoritmos e Estruturas de dados
 - Linguagens e Teoria da computação
 - Otimização
 - Entre outras
- Seu conhecimento fornece uma base sólida para construção de sistemas complexos.

O que fazem os Sistemas Operacionais

O que fazem os Sistemas Operacionais

- Sistema de computação pode ser dividido em 4 componentes:
 - 1 Hardware
 - 2 Sistema Operacional
 - 3 Programas Aplicativos
 - 4 Usuários



O que fazem os Sistemas Operacionais

- Hardware (CPU, memória e dispositivos de entrada/saída):
 - Fornece os recursos básicos de computação do sistema.
- Programas aplicativos:
 - Definem as formas pelas quais esses recursos são utilizados para resolver os problemas computacionais dos usuários.
- Sistema operacional:
 - Controla o hardware e coordena seu uso pelos diversos programas aplicativos de vários usuários.

Definindo Sistemas Operacionais

- Não existe uma definição totalmente adequada de um SO.
 - Eles existem porque oferecem uma forma razoável de resolver o problema de criar um sistema de computação utilizável.
- As funções comuns de controle e alocação de recursos são reunidas em um único software:
 - **Sistema Operacional**

Definindo Sistemas Operacionais

- Também não existe uma definição universalmente aceita sobre o que compõe o SO.
- Ponto de vista **simplista**:
 - Inclui tudo que um fornecedor oferece quando você encomenda “o sistema operacional”.

Definindo Sistemas Operacionais

- Também não existe uma definição universalmente aceita sobre o que compõe o SO.
- Ponto de vista **simplista**:
 - Inclui tudo que um fornecedor oferece quando você encomenda “o sistema operacional”.
 - Recursos incluídos variam muito entre os sistemas!

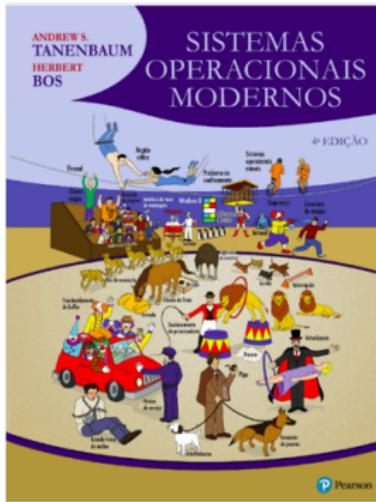
Definindo Sistemas Operacionais

- Também não existe uma definição universalmente aceita sobre o que compõe o SO.
- Ponto de vista **simplista**:
 - Inclui tudo que um fornecedor oferece quando você encomenda “o sistema operacional”.
 - Recursos incluídos variam muito entre os sistemas!
- Definição mais **comum**:
 - SO é o único programa que permanece em execução no computador durante todo o tempo.
 - *Kernel*
 - Adicionalmente, temos os programas do sistema e os programas aplicativos.

História dos Sistemas Operacionais

História dos Sistemas Operacionais

- Esta seção é baseada no capítulo 1 seção 1.2 do livro:
 - Tanenbaum, A. S., Bos, H.; Sistemas Operacionais Modernos, Editora Pearson, 4^a edição, 2016.



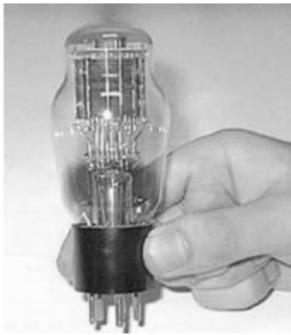
História dos Sistemas Operacionais

- Primeira Geração (1945-1955): Válvulas
- Segunda Geração (1955-1965): Transistores
- Terceira Geração (1965-1980): CIs e Multiprogramação
- Quarta Geração (1980-?): Computadores Pessoais
- Quinta Geração: Atualidade

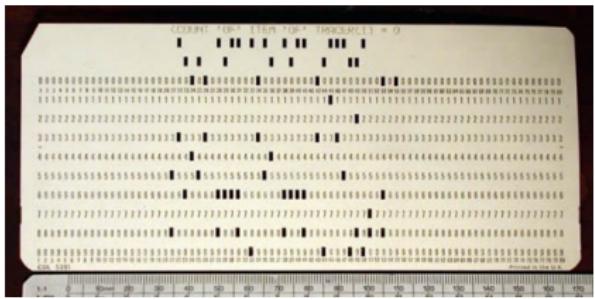
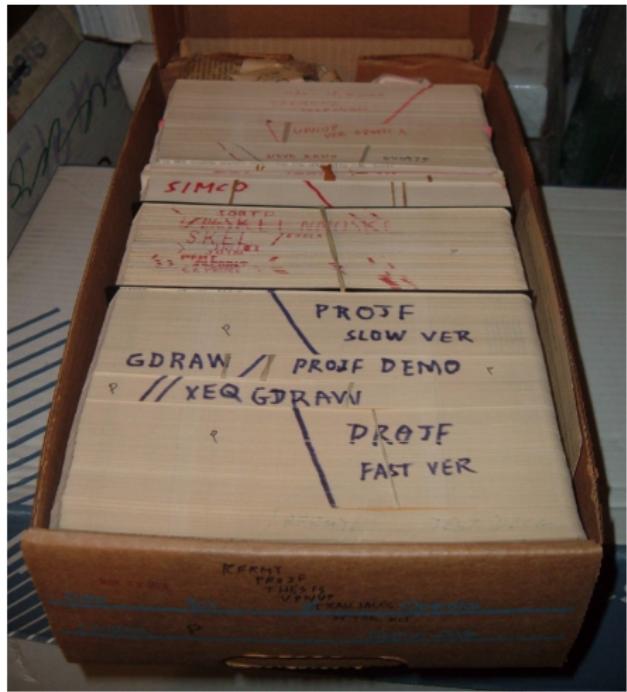
Primeira Geração (1945-1955): Válvulas

- Não existiam sistemas operacionais.
 - Programação através de chaves em um painel.
 - Código de máquina.
 - Conexão de plugs em painéis para controlar as funções da máquina.
- Operação:
 - Programador reservava tempo de máquina.
 - Inseria seu painel de programação.
 - Passava horas torcendo para que nenhuma das 20 mil válvulas queimasse durante a execução.
- No início da década de 50 os cartões perfurados foram introduzidos.
 - Operação continua a mesma.

Primeira Geração (1945-1955): Válvulas

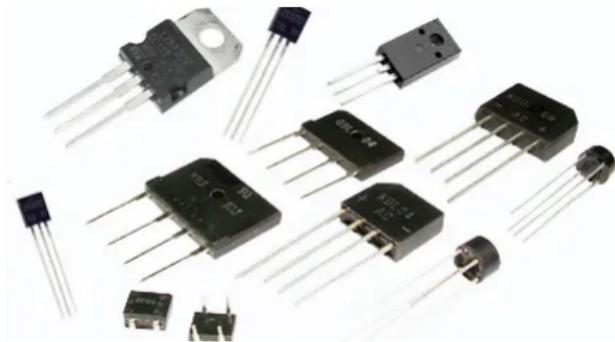


Primeira Geração (1945-1955): Válvulas



Segunda Geração (1955-1965): Transistores

- Mudança radical:
 - Computadores suficientemente confiáveis.
- Computadores de grande porte (*mainframes*):
 - Isolados em salas especiais, operadas por equipes profissionais.
 - Grandes corporações, agências governamentais ou universidades.
 - Milhões de dólares!



Mainframe

- Inicialmente, o termo era utilizado para se referir ao gabinete principal que alojava a CPU nos primeiros computadores.
 - Foram muito utilizados em guerras para cálculo balístico.
- Terminais conectados diretamente ou através de uma rede.
- Processamento de um grande volume de informações.
 - Grandes base de dados: cartões de crédito, contas bancárias, passagens aéreas, etc.



Segunda Geração (1955-1965): Transistores

■ Execução de uma tarefa:

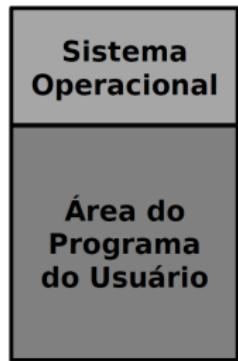
- 1 Programador escrevia no papel e depois perfurava cartões.
- 2 Entregava os cartões para um dos operadores e esperava a saída.
- 3 Operador executava cartão por cartão e caminhava até a sala da impressora para retirar a saída.

■ Problema?

- Muito tempo gasto no deslocamento dos operadores.
- Solução?

Segunda Geração (1955-1965): Transistores

- Introdução da tecnologia de disco.
 - Escalonamento de tarefas.
- **Sistema em lote (batch):**
 - Programadores entregavam as tarefas para os operadores.
 - Operadores agrupavam tarefas com necessidades semelhantes.
 - Lotes eram executados no computador (quando disponível).
 - Saída de cada lote era enviada para o respectivo programador.
 - Não existe interação com o usuário.
- Sistema operacional simples:
 - Transferia o controle de uma tarefa para a seguinte.
 - Residente na memória.



Segunda Geração (1955-1965): Transistores



Terceira Geração (1965-1980): Cls e Multiprogramação

- IBM cria o System/360:
 - Família de computadores.
 - Softwares compatíveis.
 - Computação científica e comercial.
 - Pioneiros no uso de **Circuitos Integrados** (Cls).
 - Melhor custo-benefício.
 - Sistema operacional: OS/360.
 - Enorme e extraordinariamente complexo.
 - Milhares de erros.

Terceira Geração (1965-1980): CIs e Multiprogramação



Terceira Geração (1965-1980): CIs e Multiprogramação



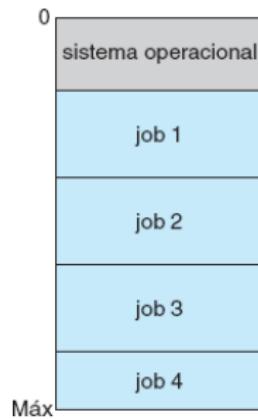
Terceira Geração (1965-1980): CIs e Multiprogramação

- Apesar dos problemas, OS/360 e sistemas operacionais similares atendiam razoavelmente bem à maioria dos clientes.
- Popularizaram técnicas fundamentais para os Sistemas Operacionais modernos:
 - **Multiprogramação e Timesharing.**

Terceira Geração (1965-1980): CIs e Multiprogramação

■ Multiprogramação:

- Memória dividida em partes, uma tarefa diferente em cada partição.
- Enquanto uma tarefa esperava que uma operação de E/S se completasse, outra poderia usar a CPU.
- CPU poderia permanecer ocupada por quase 100% do tempo.



Terceira Geração (1965-1980): CIs e Multiprogramação

■ Timesharing:

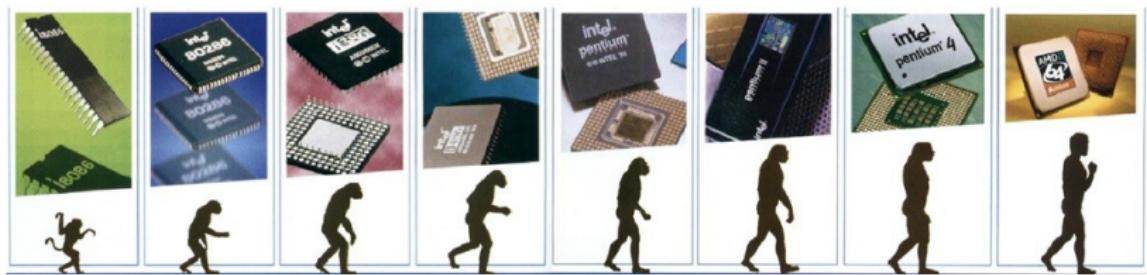
- Intervalo entre submeter uma tarefa e obter uma saída era grande.
 - Erros levariam horas para serem corrigidos.
- Timesharing surgiu como uma variante da multiprogramação:
 - Cada usuário se conectava por meio de um terminal on-line.
 - CPU executava várias tarefas, alternando entre elas.
 - Usuários podiam interagir com cada programa enquanto ele estava sendo executado.
 - Tempo de resposta curto (~ 1 segundo).
- O sistema **UNIX** surgiu nesta geração.
 - Software proprietário.

Terceira Geração (1965-1980): CIs e Multiprogramação



Quarta Geração (1980-?): Computadores Pessoais

- Circuitos integrados em larga escala (LSI):
 - Milhares de transistores em 1 cm^2 de silício.



Quarta Geração (1980-?): Computadores Pessoais

- Microcomputadores:
 - Inicialmente baratos mas com pouca potência e poder de processamento.
 - Sistemas Operacionais Simplificados:
 - CP/M, MS-DOS, etc.
 - Baseados na digitação de comandos.
- Lisa (1983):
 - Projeto de Steve Jobs.
 - Dispendioso.
 - Falhou comercialmente.
- Apple Macintosh (1984):
 - Enorme sucesso.
 - Baixo custo em relação ao projeto Lisa.
 - Amigável ao usuário.
 - Destinado a usuários que não sabiam e/ou não tinham interesse em aprender sobre computadores.

Quarta Geração (1980-?): Computadores Pessoais

- Windows (1985):
 - Baseado em interface gráfica.
 - Executado originalmente em cima do MS-DOS.
 - Ambiente gráfico sobre o MS-DOS.
- Windows 95 e Windows 98:
 - Independentes do MS-DOS.
 - Incorporaram aspectos de um sistema operacional.
- Windows NT, 2000, Millennium, XP, Vista, 7, 8, 10, 11...
- Linux, FreeBSD ...

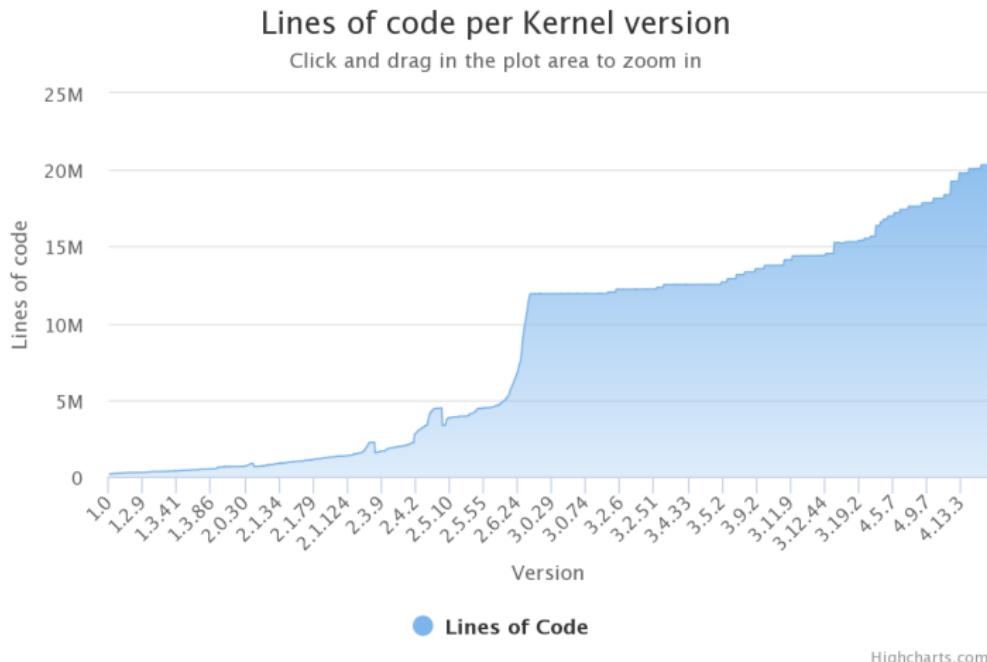
Quinta Geração: Atualidade

- Processadores com milhões de transistores
 - Arquiteturas de 64 bits
 - *Multi-cores*
- Dispositivos móveis
- Conectividade
 - Computação ubíqua e Internet das coisas (IoT)
 - Computação em nuvem
 - Computação distribuída
- Inteligência artificial

Sistemas Operacionais Modernos

- Enormes:
 - Centenas de milhares de linhas de código.
 - 100... 1000 homens-ano de desenvolvimento.
- Complexos:
 - Tipos diferentes de hardware e usuários.
 - Foco em desempenho.
- Mal compreendidos:
 - Muito grandes para serem compreendidos por um único desenvolvedor.
 - Nunca estão completamente sem erros.
 - Comportamento difícil de prever.

Sistemas Operacionais Modernos

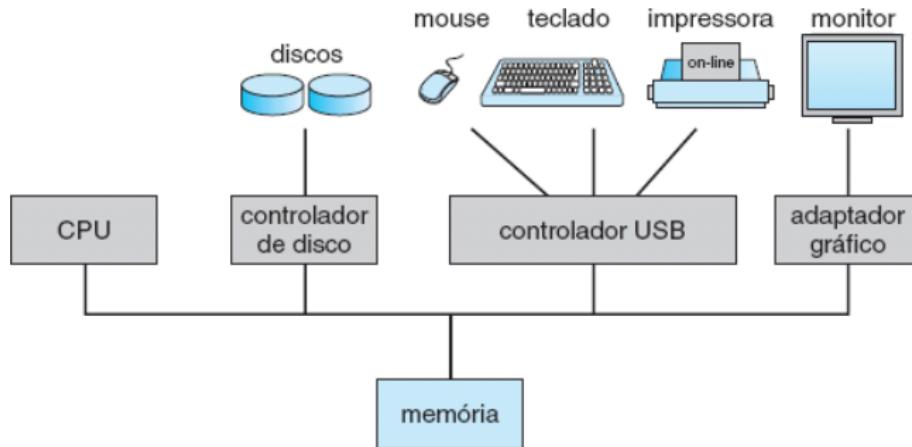


Última atualização: 2018 (kernel v. 4.15.9)

Organização do Sistema de Computação

Operação do Sistema de Computação

- Um sistema de computação de uso geral é composto por uma ou mais CPUs e vários controladores de dispositivos conectados por intermédio de um barramento comum que dá acesso à memória compartilhada.

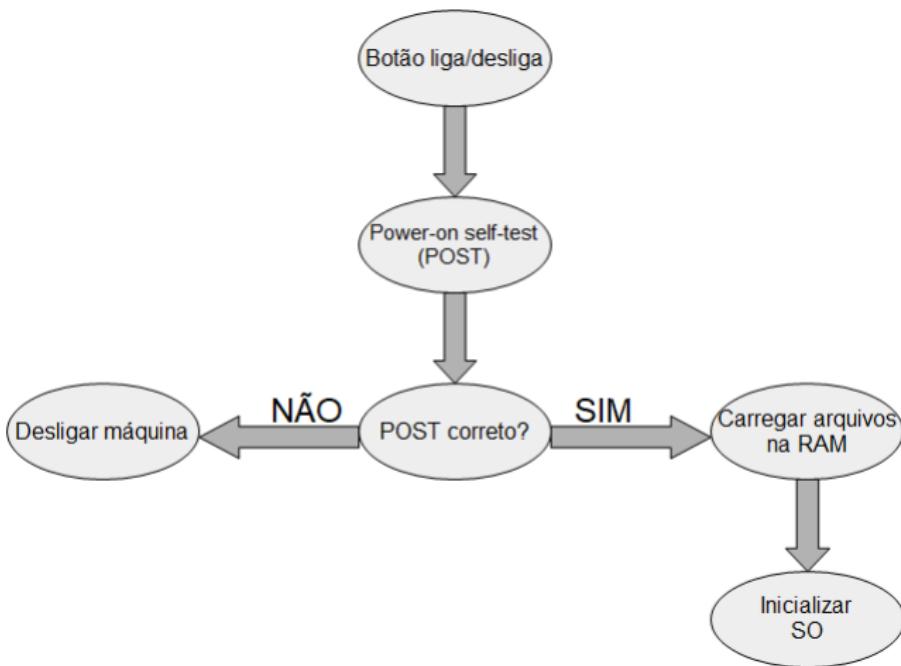


Operação do Sistema de Computação

- CPU e os controladores de dispositivos podem operar concorrentemente.
 - Competindo por ciclos de memória.
- Controlador de memória sincroniza o acesso à memória compartilhada para assegurar um acesso ordenado.

Inicialização

Inicialização



Inicialização

- Para que um computador comece a operar ele precisa ter um programa inicial para executar.
- Programa inicial ou **programa bootstrap**:
 - Tende a ser simples.
 - Armazenado dentro do hardware do computador em memória somente de leitura (ROM) ou em memória somente de leitura eletricamente apagável e programável (EEPROM).
 - **Firmware**

Inicialização

- Programa inicial ou **programa bootstrap**:
 - Inicializa todos os aspectos do sistema
 - Dos registradores da CPU aos controladores de dispositivos e conteúdos da memória.
 - Precisa saber como carregar o sistema operacional e iniciar sua execução.
 - Para alcançar esse objetivo, o programa tem que localizar e carregar na memória o kernel do sistema operacional.
 - *Basic Input/Output System (BIOS)* e *Unified Extensible Firmware Interface (UEFI)*

Inicialização

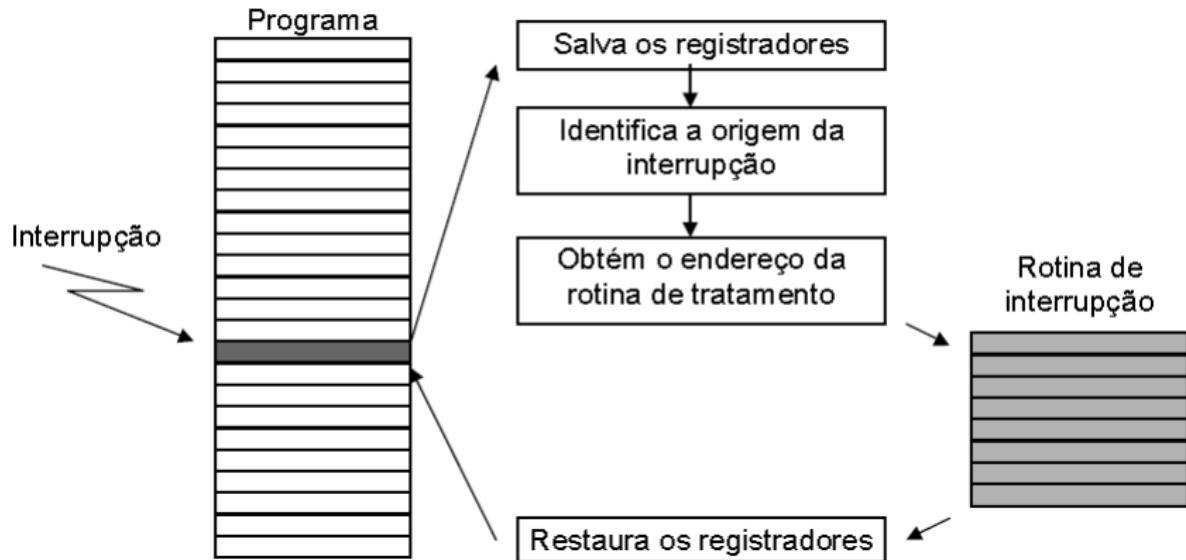
- Assim que o kernel é carregado e está em execução, ele pode começar a fornecer serviços para o sistema e seus usuários.
- Alguns serviços são fornecidos fora do kernel por programas do sistema que são carregados na memória em tempo de inicialização para se tornarem **processos do sistema** (ou **daemons do sistema**).
 - São executados durante todo o tempo em que o kernel é executado.
 - No UNIX, o primeiro processo do sistema é “init”.
 - Inicia muitos outros daemons.
- Quando essa fase é concluída, o sistema está totalmente inicializado e aguarda que algum evento ocorra.

Interrupções

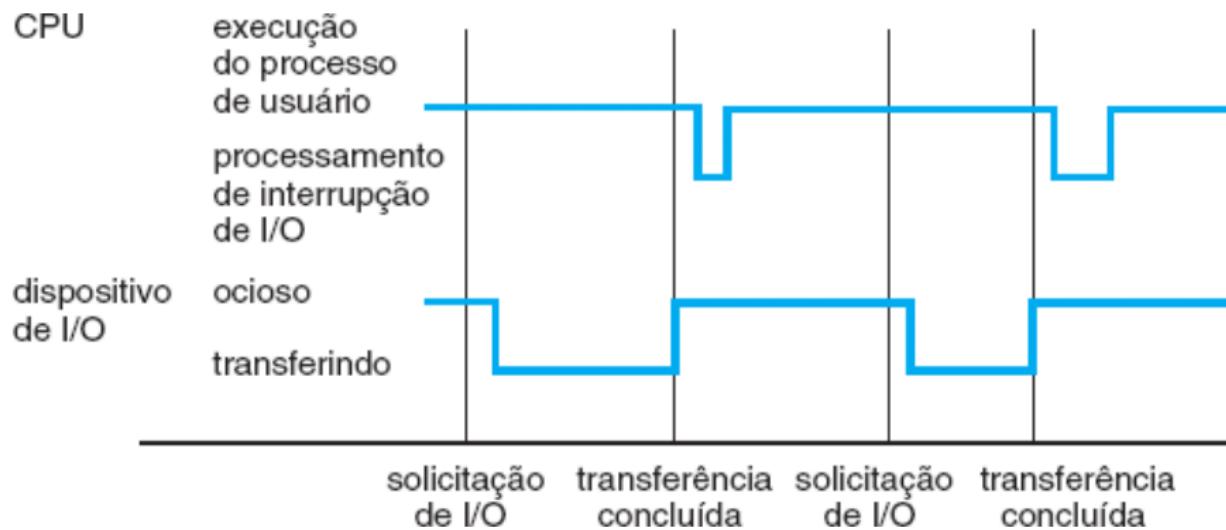
- A ocorrência de um evento é indicada por uma interrupção:
 - **Interrupção por hardware:**
 - HW envia um sinal à CPU por meio do barramento do sistema.
 - **Interrupção por software:**
 - SW dispara uma interrupção por meio de uma **chamada de sistema** (*system call*).
- Quando a CPU é interrompida, ela para o que está fazendo e transfere imediatamente a execução para uma localização fixa.
 - Essa localização fixa contém o endereço inicial no qual se encontra a rotina de serviço da interrupção.
 - A rotina de serviço da interrupção entra em operação.
 - Ao completar a execução, a CPU retoma a computação interrompida.

Interrupções

Mecanismo de interrupção



Interrupções



Interrupções

- Cada projeto de computador tem seu próprio mecanismo de interrupção.
 - Mas diversas funções são comuns a todos.
- A interrupção deve transferir o controle para a rotina de serviço de interrupção apropriada.
 - Tabela de ponteiros para rotinas de interrupção.
 - Armazenada na memória baixa.
 - Array de endereços ou **vetor de interrupções**.
 - Indexado por um único número de dispositivo, fornecido com a solicitação de interrupção.

Interrupções

- A arquitetura de interrupções também deve salvar o endereço da instrução interrompida.
 - Arquiteturas mais recentes armazenam o endereço de retorno na pilha do sistema.
- Se a rotina de interrupção precisar modificar o estado do processador ela deve salvar explicitamente o estado corrente e, então, restaurar esse estado antes de retornar.
- Após a interrupção ser atendida, o endereço de retorno salvo é carregado no contador do programa, e a computação interrompida é retomada como se a interrupção não tivesse ocorrido.

Estrutura de Armazenamento

- CPU só pode carregar instruções a partir da memória.
 - Memória principal (*random-access memory* – RAM).
 - Memória de acesso randômico dinâmica (*dynamic random-access memory* – DRAM).
- Portanto, para serem executados, os programas devem estar armazenados na memória.

Estrutura de Armazenamento

- Os computadores também usam outros tipos de memória. Alguns foram mencionados anteriormente:
 - Memória somente de leitura (ROM):
 - Não pode ser alterada.
 - Somente programas estáticos são armazenados nela.
 - A imutabilidade da ROM é útil em cartuchos de jogos.
 - Memória somente de leitura eletricamente apagável e programável (EEPROM):
 - Pode ser alterada, mas não com frequência.
 - Contém programas estáticos em sua maior parte.
 - Smartphones têm EEPROM para armazenar seus programas instalados de fábrica.

Estrutura de Armazenamento

- Todos os tipos de memória fornecem um *array* de *bytes*.
 - Cada *byte* tem seu próprio endereço.
- Interação: sequência de instruções *load* ou *store*.
- Além disso, CPU carrega automaticamente instruções para execução, a partir da memória principal.

Estrutura de Armazenamento

- O ideal é que os programas e dados residam na memória principal permanentemente.
 - Problemas?

Estrutura de Armazenamento

- O ideal é que os programas e dados residam na memória principal permanentemente.
 - Problemas?
 - Memória principal pequena.
 - Memória principal perde seus conteúdos quando a energia é desligada ou quando ela falta por outro motivo.

Estrutura de Armazenamento

- O ideal é que os programas e dados residam na memória principal permanentemente.
 - Problemas?
 - Memória principal pequena.
 - Memória principal perde seus conteúdos quando a energia é desligada ou quando ela falta por outro motivo.
 - Solução:

Estrutura de Armazenamento

- O ideal é que os programas e dados residam na memória principal permanentemente.
 - Problemas?
 - Memória principal pequena.
 - Memória principal perde seus conteúdos quando a energia é desligada ou quando ela falta por outro motivo.
 - Solução:
 - **Memória secundária** como uma extensão da memória principal.
 - Principal requisito: ser capaz de armazenar grandes quantidades de dados permanentemente.

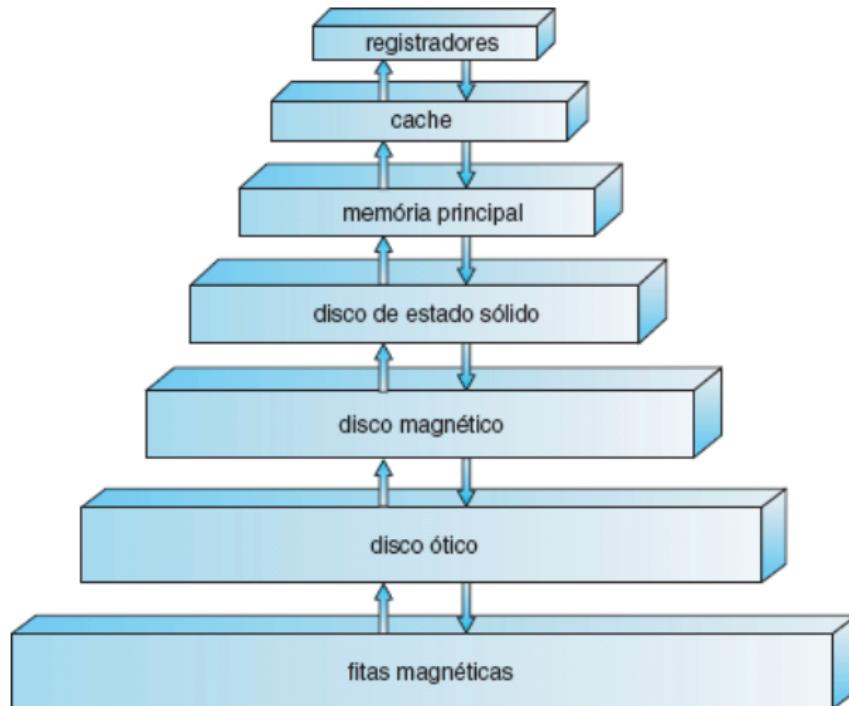
Estrutura de Armazenamento

- A maioria dos programas é armazenada em um disco até que seja carregada na memória.
 - Muitos programas utilizam o disco tanto como fonte quanto como destino de seu processamento.
- O gerenciamento apropriado do armazenamento em disco é de importância capital para um sistema de computação!

Estrutura de Armazenamento

- Outros sistemas de armazenamento possíveis:
 - Memória cache, CD-ROM, fitas magnéticas, etc.
- As principais diferenças entre os vários sistemas de armazenamento residem na velocidade, no custo, no tamanho e na volatilidade.
 - Volátil vs não volátil.

Estrutura de Armazenamento



Hierarquia dos dispositivos de armazenamento.

Estrutura de Entrada/Saída

- Armazenamento é apenas um dos muitos tipos de dispositivos de E/S.
- Grande parte do código do SO é dedicada ao gerenciamento de E/S.
 - Importância para a confiabilidade e o desempenho de um sistema.
 - Natureza variada dos dispositivos.
- Relembrando:
 - Sistema de computação: CPUs + controladores de dispositivos conectados por um barramento comum.

Estrutura de Entrada/Saída

■ Controlador de dispositivos:

- Responsável por um tipo específico de dispositivo.
 - Pode haver mais de um dispositivo conectado.
- Mantém armazenamento em buffer local e um conjunto de registradores de uso específico.
- É responsável por movimentar os dados entre os dispositivos periféricos que controla e seu buffer de armazenamento local.

Estrutura de Entrada/Saída

■ Driver de dispositivo:

- Normalmente, um para cada controlador de dispositivos.
- Entende o controlador de dispositivos e fornece uma interface uniforme entre o dispositivo e o resto do sistema operacional.
- Driver → SW
- Controlador → HW

Estrutura de Entrada/Saída

- Operação de E/S:

- 1 Driver do dispositivo carrega os registradores dentro do controlador do dispositivo.

Estrutura de Entrada/Saída

■ Operação de E/S:

- 1 Driver do dispositivo carrega os registradores dentro do controlador do dispositivo.
- 2 Controlador examina o conteúdo dos registradores para determinar que ação tomar.
 - Ler um caractere a partir do teclado, por exemplo.

Estrutura de Entrada/Saída

■ Operação de E/S:

- 1 Driver do dispositivo carrega os registradores dentro do controlador do dispositivo.
- 2 Controlador examina o conteúdo dos registradores para determinar que ação tomar.
 - Ler um caractere a partir do teclado, por exemplo.
- 3 Controlador inicia a transferência de dados do dispositivo para o seu buffer local.

Estrutura de Entrada/Saída

■ Operação de E/S:

- 1 Driver do dispositivo carrega os registradores dentro do controlador do dispositivo.
- 2 Controlador examina o conteúdo dos registradores para determinar que ação tomar.
 - Ler um caractere a partir do teclado, por exemplo.
- 3 Controlador inicia a transferência de dados do dispositivo para o seu buffer local.
- 4 Quando a transferência de dados é concluída, informa ao driver do dispositivo que terminou sua operação.
 - Interrupção!

Estrutura de Entrada/Saída

■ Operação de E/S:

- 1 Driver do dispositivo carrega os registradores dentro do controlador do dispositivo.
- 2 Controlador examina o conteúdo dos registradores para determinar que ação tomar.
 - Ler um caractere a partir do teclado, por exemplo.
- 3 Controlador inicia a transferência de dados do dispositivo para o seu buffer local.
- 4 Quando a transferência de dados é concluída, informa ao driver do dispositivo que terminou sua operação.
 - Interrupção!
- 5 O driver do dispositivo retorna o controle para o SO.
 - Retornando os dados ou um ponteiro para os dados se a operação foi uma leitura.
 - Para outras operações, retorna informações de status.

Estrutura de Entrada/Saída

- Limitações da operação de E/S baseada em interrupção?

Estrutura de Entrada/Saída

- Limitações da operação de E/S baseada em interrupção?
 - É adequada para a movimentação de pequenas quantidades de dados.
 - Pode produzir um *overhead* alto quando usado na movimentação de dados de massa como no E/S de disco.

Estrutura de Entrada/Saída

- Limitações da operação de E/S baseada em interrupção?
 - É adequada para a movimentação de pequenas quantidades de dados.
 - Pode produzir um *overhead* alto quando usado na movimentação de dados de massa como no E/S de disco.
- Solução:
 - **Acesso direto à memória (DMA).**
 - Transferência de um bloco inteiro de dados diretamente da memória para o próprio buffer do controlador ou a partir dele para a memória, sem intervenção da CPU.
 - **Uma** interrupção por bloco para informar ao driver do dispositivo que a operação foi concluída.
 - Enquanto o controlador do dispositivo está executando essas operações, a CPU está disponível para cumprir outras tarefas.

Arquitetura dos Sistemas de Computação

Sistemas Uniprocessadores

- Uma CPU principal capaz de executar um conjunto de instruções de uso geral.
- Quase todos os sistemas de processador único também têm outros processadores de uso específico.
 - Processadores de dispositivos específicos, como controladores de disco, teclado e monitor; ou processadores de uso mais genérico, como os processadores de E/S.
 - Executam um conjunto limitado de instruções.
 - Não executam processos de usuário.
 - Podem ou não ser gerenciados pelo SO.

Sistemas Multiprocessadores

- Também conhecidos como **sistemas paralelos** ou **sistemas multicore**.
- Dois ou mais processadores em estreita comunicação.
 - Compartilhando o barramento do computador e algumas vezes o relógio, a memória e os dispositivos periféricos.
- Apareceram pela primeira vez em servidores.
 - Migraram para sistemas desktop e laptop.
 - Dispositivos móveis como os smartphones e tablets.

Sistemas Multiprocessadores

- Vantagens:

- Vantagens:
 - **Aumento do throughput:**
 - Mais trabalho executado em menos tempo.
 - No entanto, a taxa de aumento de velocidade com N processadores não é N .

- Vantagens:

- **Aumento do throughput:**

- Mais trabalho executado em menos tempo.
 - No entanto, a taxa de aumento de velocidade com N processadores não é N .

- **Economia de escala:**

- Sistemas multiprocessadores podem custar menos do que múltiplos sistemas equivalentes de processador único.
 - Compartilham periféricos, memória de massa e suprimentos de energia.

- Vantagens:

- **Aumento do throughput:**

- Mais trabalho executado em menos tempo.
 - No entanto, a taxa de aumento de velocidade com N processadores não é N .

- **Economia de escala:**

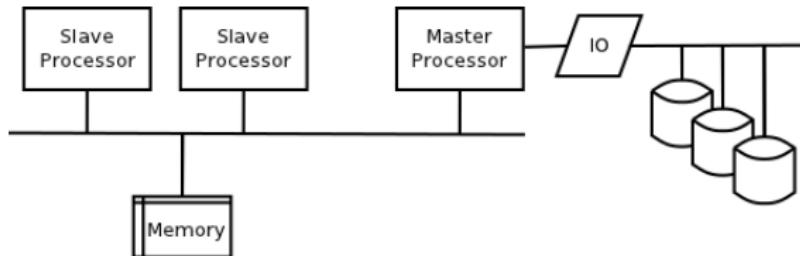
- Sistemas multiprocessadores podem custar menos do que múltiplos sistemas equivalentes de processador único.
 - Compartilham periféricos, memória de massa e suprimentos de energia.

- **Aumento da confiabilidade:**

- Se as funções puderem ser distribuídas apropriadamente entre vários processadores, a falha de um processador não interromperá o sistema.
 - Tornará apenas mais lento.

Sistemas Multiprocessadores

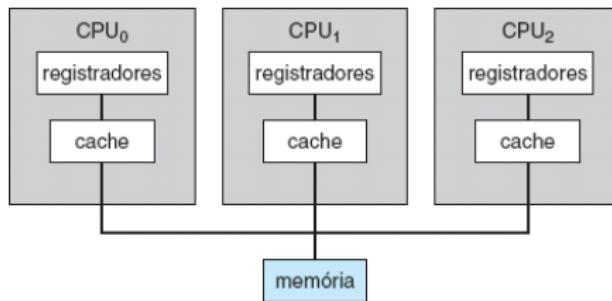
- Divididos em dois tipos: multiprocessamento assimétrico e multiprocessamento simétrico.
- **Multiprocessamento assimétrico (ASMP):**
 - Relacionamento mestre-escravo.
 - A cada processador é designada uma tarefa específica.
 - Um processador mestre controla o sistema.
 - Demais processadores consultam o mestre para instruções ou possuem tarefas predefinidas.



Sistemas Multiprocessadores

■ Multiprocessamento simétrico (SMP):

- Cada processador executa todas as tarefas do SO.
- Não existe um relacionamento mestre-escravo entre eles.
- Muitos processos podem ser executados simultaneamente sem causar uma deterioração significativa de desempenho.



Sistemas Multiprocessadores

- Tendência no projeto de CPUs:
 - Inclusão de múltiplos núcleos de computação em um único chip.
 - *Multicore*
 - Mais eficientes do que vários chips de núcleo único.

Sistemas Agrupados (Clusters)

- Reúnem várias CPUs.
- Diferem dos sistemas multiprocessadores por serem compostos de dois ou mais sistemas individuais (nós) acoplados.
- Cada nó pode ser um sistema uniprocessador ou um sistema multicore.
- Computadores em cluster compartilham memória e são estreitamente conectados por uma rede local ou de uma interconexão mais veloz.
- Usados para fornecer serviço de alta disponibilidade.
- Computação de alto desempenho.

Estrutura do Sistema Operacional

Estrutura do Sistema Operacional

- Um dos aspectos mais importantes dos SOs é sua capacidade de multiprogramar.
 - Geralmente, um único programa não pode manter a CPU ou os dispositivos de E/S ocupados o tempo todo.
- **Multiprogramação** aumenta a utilização da CPU.
 - SO mantém vários jobs na memória.
 - Inicialmente eles são mantidos em disco no **pool de jobs**.
 - O conjunto de jobs na memória é um subconjunto do pool de jobs.

Multiprogramação

- 1 SO seleciona e começa a executar um dos jobs da memória.
 - Em dado momento, o job pode ter de aguardar alguma tarefa (E/S, por exemplo).
- 2 SO passa para um novo job e o executa.
 - Quando este job tem de aguardar, a CPU é redirecionada para outro job, e assim por diante.
- 3 Por fim, o primeiro job sai da espera e retoma a CPU.
- 4 Contanto que pelo menos um job tenha de ser executado, a CPU nunca fica ociosa.

Estrutura do Sistema Operacional

- Sistemas multiprogramados fornecem um ambiente em que os diversos recursos do sistema são utilizados eficientemente.
 - Mas não possibilitam a interação do usuário com o sistema de computação.
 - Solução?

Estrutura do Sistema Operacional

- Sistemas multiprogramados fornecem um ambiente em que os diversos recursos do sistema são utilizados eficientemente.
 - Mas não possibilitam a interação do usuário com o sistema de computação.
 - Solução?
 - Tempo compartilhado (*timesharing* ou multitarefa).

Estrutura do Sistema Operacional

- Sistemas multiprogramados fornecem um ambiente em que os diversos recursos do sistema são utilizados eficientemente.
 - Mas não possibilitam a interação do usuário com o sistema de computação.
 - Solução?
 - Tempo compartilhado (*timesharing* ou multitarefa).
- **Tempo Compartilhado:**
 - Extensão lógica da multiprogramação.
 - CPU executa vários jobs alternando-se entre eles.
 - Mudanças de job ocorrem com tanta frequência que os usuários podem interagir com cada programa enquanto ele está em execução.
 - Cada usuário tem a impressão de que o sistema de computação inteiro está sendo dedicado ao seu uso.
- Um programa carregado na memória e em execução é chamado de **processo**.

Operações do Sistema Operacional

Operações do Sistema Operacional

- SOs modernos são dirigidos por interrupções.
 - Se não existem processos para executar, dispositivos de E/S para servir e usuários a quem responder: SO permanece inativo.
- Eventos são quase sempre sinalizados pela ocorrência de uma interrupção ou de uma exceção (trap).
 - **Exceção:** interrupção gerada por software causada por um erro ou por uma solicitação específica proveniente de um programa de usuário para que um serviço do sistema operacional seja executado.

Operações do Sistema Operacional

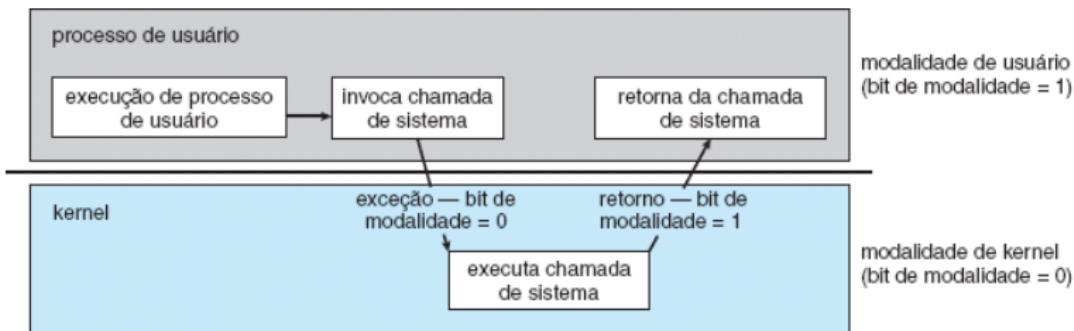
- SO e usuários compartilham recursos de HW e SW.
 - Necessário assegurar que um erro em um programa de usuário cause problemas somente para o programa em execução.
- Um SO projetado adequadamente deve assegurar que um programa incorreto (ou malicioso) não possa causar a execução incorreta de outros programas.

Modo Dual

- Para garantir a execução apropriada do SO é necessário distinguir entre a execução de código do SO e de código definido pelo usuário.
 - Solução comum: modo dual.
- **Modo dual:**
 - Modalidade de usuário e modalidade de kernel (ou modalidade de supervisor, de sistema ou privilegiada).
 - Bit de modalidade adicionado ao HW.

Modo Dual

- Quando o sistema de computação está operando em nome de uma aplicação de usuário:
 - Modalidade de usuário.
- Quando a aplicação de usuário solicita um serviço do SO:
 - Passar da modalidade de usuário para a de kernel.



Modo Dual

- Inicialização do sistema:
 - HW começa a operar em modalidade de kernel.
 - SO é carregado.
 - Inicia aplicações de usuário → modalidade de usuário.
- Sempre que uma exceção ou interrupção ocorre:
 - modalidade de usuário → modalidade de kernel.
 - Sempre que o SO obtém o controle do computador, ele está em modalidade de kernel.
- O sistema sempre passa para a modalidade de usuário antes de devolver o controle para um programa de usuário.

- Fornece os meios para a proteção do SO contra usuários errantes (e a proteção desses contra eles mesmos).
- **Instruções privilegiadas:**
 - Instruções de máquina que podem causar erro.
 - Somente em modalidade de kernel.
- Uma tentativa de executar uma instrução privilegiada em modalidade de usuário é tratada como ilegal e interceptada (por uma exceção) para o SO.
- Exemplos de instruções privilegiadas:
 - Passagem para a modalidade de kernel.
 - Controle de E/S.
 - Gerenciamento do timer.
 - Gerenciamento de interrupções.

- Exemplo de falha de proteção:
 - MS-DOS foi escrito para a arquitetura Intel 8088.
 - Não possui bit de modalidade.
 - Programa de usuário operando incorretamente pode tirar do ar o SO gravando dados sobre ele.
 - Vários programas poderiam gravar ao mesmo tempo em um dispositivo, com resultados potencialmente desastrosos.
- Conceito de modalidades pode ser estendido para além de duas modalidades (mais de um bit).

- SO deve manter o controle sobre a CPU.
 - Não permitir que um programa de usuário fique travado em um loop infinito ou não chame os serviços do sistema e nunca retorne o controle ao SO.
 - Trágico para o *Timesharing*.
- Solução: **temporizador (timer)**.
 - Configurado para interromper o computador após um período especificado.
 - Fixo (por exemplo, 1/60 segundos) ou variável (por exemplo, de 1 milissegundo a 1 segundo).
 - Interrupção gerada quando o timer é zerado.

- Quando o timer causa a interrupção, o controle é transferido automaticamente para o SO.
 - Pode tratar a interrupção como um erro fatal ou dar mais tempo ao programa.
- Instruções que modificam o conteúdo do timer são privilegiadas.
- Timer pode ser usado para impedir que um programa de usuário seja executado por muito tempo.
 - Exemplo: comando *timeout* do Linux.

Gerenciamento de Processos

- Programa vs Processo.
 - Um programa por si só não é um processo.
 - Programa é uma entidade passiva.
 - Processo é uma entidade ativa.
- Processo precisa de recursos.
 - Tempo de CPU, memória, arquivos e dispositivos de E/S.
- Recursos são fornecidos na criação do processo ou durante a sua execução.
- Quando o processo terminar, o SO “toma de volta” os recursos reutilizáveis.

Processo de um único thread

- Um contador de programa (PC).

Processo de um único thread

- Um contador de programa (PC).
 - Indica qual a próxima instrução a ser executada.

Processo de um único thread

- Um contador de programa (PC).
 - Indica qual a próxima instrução a ser executada.
- Execução sequencial.
- No máximo uma instrução é executada em nome do processo.
- Dois processos possam estar associados ao mesmo programa.
 - Duas sequências de execução separadas.

- SO é responsável por:
 - Executar o escalonamento de processos e threads nas CPUs.
 - Criar e excluir processos de usuário e do sistema.
 - Suspender e retomar processos.
 - Fornecer mecanismos de sincronização e comunicação de processos.

Gerenciamento da Memória

Gerenciamento da Memória

- Memória principal é fundamental para a operação de um computador moderno.
- Um grande conjunto de *words* ou *bytes* endereçáveis.
- Pode ser enxergada como um depósito de dados rapidamente acessíveis, compartilhados por dispositivos de E/S e pela CPU.
- Único dispositivo de armazenamento grande que a CPU pode endereçar e acessar diretamente.

- SO é responsável por:
 - Controlar quais partes da memória estão sendo correntemente utilizadas e quem as está utilizando.
 - Decidir que processos (ou partes de processos) e dados devem ser transferidos para dentro e para fora da memória.
 - Alocar e desalocar espaço de memória conforme necessário.

Gerenciamento do Armazenamento

Gerenciamento do Armazenamento

- Para tornar o sistema de computação conveniente para os usuários, o SO fornece uma visão uniforme e lógica do armazenamento de informações.
 - Abstrai, das propriedades físicas dos seus dispositivos de armazenamento, a definição de uma unidade lógica de armazenamento: o **arquivo**.
 - Conjunto de informações relacionadas definido por seu criador.
 - Programas e dados.
- SO mapeia arquivos para mídia física e acessa esses arquivos por meio dos dispositivos de armazenamento.

Gerenciamento do Sistema de Arquivos

- Um dos componentes mais visíveis de um SO.
- Computadores podem armazenar informações em diferentes tipos de mídias físicas:
 - Fitas magnéticas, discos rígidos, SSDs, etc.
- Cada tipo de dispositivo possui diferentes propriedades:
 - Velocidade de acesso, capacidade, taxa de transferência, método de acesso (sequencial ou aleatório), etc.

Gerenciamento do Sistema de Arquivos

- SO é responsável pelas seguintes atividades relacionadas com o gerenciamento de arquivos:
 - Criar e apagar arquivos.
 - Criar e apagar diretórios para organizar arquivos.
 - Suportar primitivas para a manipulação de arquivos e diretórios.
 - Mapear arquivos em memória secundária.
 - Fazer backup de arquivos em mídias de armazenamento estáveis (não voláteis).

- Memória secundária como backup da memória principal.
- SO é responsável pelas seguintes atividades relacionadas com o gerenciamento de disco:
 - Gerenciar o espaço livre.
 - Alocar espaço de armazenamento.
 - Fazer o escalonamento de disco.

- Um dos objetivos de um SO é ocultar dos usuários as peculiaridades de dispositivos de hardware específicos.
 - No UNIX as peculiaridades dos dispositivos de E/S são ocultas de grande parte do próprio SO pelo subsistema de E/S.
 - O subsistema de E/S tem vários componentes:
 - Um componente de gerenciamento de memória que inclui o uso de *buffer*, de *cache* e *spooling*.
 - Uma interface genérica para drivers de dispositivos.
 - Drivers para dispositivos de hardware específicos.

Proteção e Segurança

Proteção e Segurança

- Se um sistema de computação tem múltiplos usuários e permite a execução concorrente de múltiplos processos, então o acesso aos dados deve ser regulado.
- Mecanismos asseguram que arquivos, segmentos de memória, CPU e outros recursos possam ser operados apenas pelos processos que receberam autorização apropriada do SO.
- Exemplos:
 - O hardware de endereçamento da memória assegura que um processo só possa ser executado dentro de seu próprio espaço de endereçamento.
 - *Timer* assegura que nenhum processo possa obter o controle da CPU indefinidamente.
 - Registradores de controle de dispositivo não podem ser acessados por usuários para que a integridade dos diversos dispositivos periféricos seja protegida.

- **Proteção** é qualquer mecanismo que controle o acesso de processos ou usuários aos recursos definidos por um sistema de computação.
 - Esse mecanismo deve fornecer os meios para a especificação dos controles a serem impostos e os meios para sua imposição.
- Um sistema orientado à proteção fornece meios para a distinção entre o uso autorizado e não autorizado.

Proteção e Segurança

- Um sistema pode ter proteção adequada e, mesmo assim, estar propenso a falhas e permitir acesso inapropriado.
 - Por exemplo, um usuário cujas informações de autenticação foram roubadas.
 - Seus dados poderiam ser copiados ou apagados, mesmo que a proteção de arquivos e memória esteja funcionando.
- É responsabilidade da **segurança** a defesa de um sistema contra ataques externos e internos.
 - Vírus, *worms*, ataques de recusa de serviço, roubo de identidade e roubo de serviço.
- A prevenção contra alguns desses ataques é considerada uma função do SO em determinados sistemas, enquanto outros sistemas a deixam para a política adotada ou para um software adicional.

Dúvidas?

