



Revisão geral

SO 08/09



Histórico (de acordo com Tanenbaum)

- Primeira geração (1945-55)
 - Máquinas com tubo à vácuo e “plugboards”
 - Perfuradoras de cartão
 - Cálculos numéricos simples
 - Nenhuma linguagem ou SO presentes
- Segunda geração (1955-65)
 - Transistores e sistemas em lotes (batch systems)
 - Assembler ou Fortran usados como linguagem
 - Processamento em lotes deu origem a uma forma muito primitiva de SO (o primeiro ancestral 😊)



Histórico (cont.)

- Terceira geração (1965-1980)
 - Circuitos integrados e **multiprogramação**
 - IBM OS/360: primeiro a introduzir o conceito de multiprogramação
 - **Spooling (Simultaneous Peripheral Operation on Line)**
 - **Time sharing ou multitasking**
 - **MULTICS (MULTiplexed Information and Computing Service)**



Histórico (cont.)

- Quarta geração (1980-)
 - Computador pessoal
 - MS-DOS e UNIX
 - Criação de SOs “user-friendly”
 - SOs para redes de computadores
 - SOs distribuídos



Conceitos básicos

- **Sistema de Operação:** difícil definir...
- Dois conceitos dependendo da função que se quer enfatizar:
 - **Máquina estendida**
 - Estende o hardware com uma camada amigável que permite ao usuário manipular de forma mais conveniente os recursos de hardware (instruções de máquina, organização de memória, entrada e saída e estrutura dos barramentos)
 - **Gerenciador de recursos**
 - Provê uma forma ordenada, controlada e eficiente de alocação de processadores, memórias e dispositivos de entrada e saída entre os vários programas que competem para usá-los



Conceitos básicos (cont.)

- **Sistema em lotes** (batch system): programas são executados do início até o fim (podendo ser interrompidos apenas por operações de entrada e saída) e novos programas são enfileirados esperando pela CPU
- **Programa**: texto do código de um programa, ainda não em execução (estático)
- **Processo**: programa em execução (dinâmico)
- **Job**: termo utilizado para um processo de um sistema em lotes



Conceitos básicos (cont.)

- **Multiprogramação**: permite sobreposição de computação com entrada e saída. Permite que múltiplos processos estejam em memória ao mesmo tempo
- **Timesharing**: variante de multiprogramação, que define quotas de tempo para um processo utilizar a CPU. Permite interromper um processo em operações diferentes de entrada e saída
- **Tarefa** (task): termo utilizado para um processo em um sistema multiprogramado
- **Spooling**: capacidade de processar algum job assim que este chega no sistema através da sobreposição de operações muito lentas (por exemplo, dispositivos de entrada e saída) com processamento



Conceitos básicos (cont.)

- **Starvation** (inanição): processo fica na fila para sempre sem ter chance de executar
- **Deadlock** (impasse): condição em que processos ficam bloqueados esperando por algum evento que nunca vai poder acontecer
- **Buffering**: utilização de porções de memória para guardar dados que vêm de dispositivos com baixa velocidade para agilizar o processamento de algum dado
- **Caching**: guardar dados para posterior utilização evitando ter que fazer novo acesso a dispositivos lentos (pode ser implementado aproveitando buffering)



Funções de um SO atual

- Interface com o usuário (shell, janelas gráficas, linguagem de comandos, chamadas às bibliotecas do sistema)
 - Comandos básicos:
 - Manipulação de ficheiros e diretórios
 - Manipulação de processos
 - Comunicação entre processos
 - Manipulação de dispositivos de entrada e saída
 - Manipulação de data/hora
 - Consulta sobre o estado do sistema (cpu, memória, sistema de ficheiros, dispositivos de entrada e saída etc)



Funções de um SO atual (cont.)

- Gerência de recursos:
 - Memória
 - Cpu
 - Disco
 - Outros dispositivos
- Gerência de processos:
 - Usuário
 - Sistema



Estrutura de SOs

- **Monolítica:** todas as funções concentradas no kernel sem interface bem definida
- **Em camadas:** interface bem definida entre funções
- **Máquinas virtuais:** permite vários SOs sobre um mesmo hardware
- **Cliente-servidor:** funções rodam a nível de usuário como clientes. Kernel apenas estabelece a comunicação entre o cliente e o servidor
- **Micro-kernel:** funções do SO passam para espaço de usuário



Processos

- Hierarquia
- Estados: running, ready, blocked
- Tabela de processos
- Process Control Block (PCB)
 - Registradores
 - PC
 - PSW
 - Ponteiro para a pilha
 - Estado do processo
 - Hora em que o processo começou a executar
 - Tempo de cpu utilizado
 - Tempo de cpu dos filhos
 - Hora do próximo alarme
 - Ponteiros para filas de mensagens
 - Bits de sinal pendentes
 - Id do processo
 - Vários flags



Processos (cont.)

- Áreas de memória (tabela de processos)
 - Ponteiro para o segmento de texto
 - Ponteiro para o segmento de dados
 - Ponteiro para o segmento bss
 - Estado de término
 - Id do processo
 - Id do processo pai



Processos

■ Interrupções

- Associado com cada classe de dispositivo de entrada e saída há um vetor de interrupções localizado na base da memória.
- Contém o endereço do serviço de interrupção
- Serviço de interrupção:
 - Salva contexto do processo
 - **Troca de contexto**: substituição de um processo que estava na cpu por um outro processo que estava na fila de prontos (ready)



Threads

- São criadas e executam no contexto de um processo
- Troca de contexto para threads não retira o processo da cpu!
- Informação a ser guardada com uma troca de contexto de thread é menor do que a info necessária para trocar o contexto de um processo
- Criação de uma thread: usualmente, **thread_create()**



Processos

- **Criação** no programa do usuário: `fork()`
- **Comunicação entre processos**
(IPC – Inter Process Communication):
 - Pipes
 - `pipe()`
 - Sinais
 - `signal()`, `kill()`, `alarm()`, `pause()`, `sleep()` etc
 - `exit()`, `return()`, `wait()`, `waitpid()`
 - Memória compartilhada
 - `mmap()`, `shmget()`, `shmat()`
 - Troca de mensagens
 - `send()`, `receive()`
 - Sockets
 - `socket()`, `bind()`, `connect()`, `listen()`, `accept()`,
`send()`, `recv()`, `sendto()`, `recvfrom()`



Processos (cont.)

- **Condições de corrida** (race conditions): 2 ou + processos tentando escrever no mesmo recurso simultaneamente
- Como resolver: (espera ocupada ou filas)
 - **Exclusão mútua** em **seções críticas**
 - Soluções em hardware:
 - Instruções atômicas (read-modify-write)
 - test-and-set, swap, fetch-and-phi, etc
 - Soluções em software:
 - Dekker, Petterson (2 e n processos)
 - Semáforos (binários e contadores)
 - Variáveis condicionais
 - Monitores



Processos (cont.)

- Problemas clássicos de sincronização entre processos:
 - Produtor-consumidor
 - Jantar dos filósofos
 - Leitores e escritores
 - Barbeiro dorminhoco



Processos (cont.)

- **Escalonamento**: forma de impor uma ordem para execução de processos
 - Problema: 1 única CPU para todos os processos, como escolher o próximo processo?
- 2 níveis:
 - Escalonador de CPU (curto prazo)
 - Escolhe próximo processo da fila de prontos
 - Escalonador de longo prazo



Processos (cont.)

■ Variáveis:

- **Justiça**: todos os processos terão chance de executar em algum momento no futuro?
- **Eficiência**: o processador ficará ocioso por longo tempo?
- **Tempo de resposta**: quanto tempo o usuário deve esperar para obter a primeira resposta?
- **Turnaround**: quanto tempo o processo vai passar no sistema até finalizar sua execução?
- **Throughput**: quantos processos o meu sistema vai conseguir processar por unidade de tempo?



Processos (cont.)

■ Algoritmos mais comuns:

- Primeiro a chegar é o primeiro a sair (FCFS/FIFO)
- Menor job primeiro (SJF – Shortest Job First)
- Shortest Remaining Time First (SRTF): utiliza técnica de envelhecimento para evitar “starvation”
- Escalonamento baseado em prioridades
- Round-robin
- Filas de níveis múltiplos
- Filas de níveis múltiplos com realimentação



Gerência de Memória

■ Requisitos:

- Relocação de endereços
- Proteção de espaço de endereçamento de processos de usuário e de sistema
- Compartilhamento
- Organização lógica
- Organização física



Gerência de Memória (cont.)

- Técnicas básicas:
 - Partições fixas
 - Partições dinâmicas/variáveis
 - Bitmaps ou
 - listas encadeadas: first-fit, next-fit, best-fit, worst-fit, quick-fit, sistema buddy
 - Paginação
 - Segmentação
 - Combinação de ambas pode ser utilizada
- **Overlays e Memória virtual**: permitem que programas maiores do que a memória física da máquina (RAM) possam executar
- **Memória virtual** tb permite utilização mais eficiente da memória



Gerência de Memória (cont.)

- Endereços lógicos e físicos
- Tabelas de páginas
- TLB (Translation Lookaside Buffer): cache de páginas de processos
- O que fazer quando a memória está totalmente ocupada e um novo processo precisa executar?



Gerência de Memória (cont.)

- Algoritmos de substituição de páginas
 - Página ótima
 - Não usada recentemente (NRU)
 - FIFO
 - Segunda tentativa
 - Relógio
 - LRU
 - NFU
 - Working Set
 - WSclock



Sistemas de Ficheiros

■ Gerência

- Funções para operações básicas e avançadas sobre ficheiros
- Organização lógica e física
 - Acesso rápido (leituras e escritas)
 - Fácil de atualizar
 - Economia de armazenamento
 - Manutenção simples
 - confiabilidade



Sistemas de Ficheiros (cont.)

- Tipos de estrutura de ficheiros:
 - Sequencial
 - Registros
 - Árvore
- Tipo de acesso:
 - Sequencial ou aleatório
- Atributos e operações sobre ficheiros



Diretórios

- Um único nível

- Problema para separar usuários e ficheiros com mesmo nome, mas de conteúdos diferentes

- 2 níveis

- Problema para separar ficheiros com mesmo nome, mas de conteúdos diferentes

- Hierárquico

- árvores ou árvores balanceadas

- Operações sobre diretórios



Sistemas de ficheiros

- Implementações
- Layouts
- i-nodes
- Superblocks
- Compartilhamento de ficheiros e diretórios (links, formas de implementação)
- Aproveitamento de espaço
- Confiabilidade (formas de recuperação de erros)
- Exemplos de sistemas de ficheiros
- RAIDs



Entrada e saída

- Principles of I/O hardware
 - Interrupções e funcionamento do hardware de E/S
- Principles of I/O software
 - Objetivos, programmed I/O etc
 - Problemas de ordenação de pedidos (impasses)
- I/O software layers
- Disks
 - Disk scheduling (SCAN)
- Character-oriented terminals
- Graphical user interfaces
- Network terminals
- Power management



Impasses

- Detecção e prevenção de impasses (deadlocks)
 - Condições necessárias para a ocorrência de um impasse
 - Modelagem através de um grafo
 - Utilização do grafo para implementação de algoritmos de detecção e prevenção



Sockets

- Comunicação entre processos
 - AF_UNIX: na mesma máquina
 - AF_INET: entre máquinas numa rede
- Clientes e servidores
- Comunicação fiável (TCP) ou não fiável (UDP)
- Protocolo comumente utilizado: IP



Sockets (cont.)

■ Normalmente:

- Cria-se um socket
- Associa-se o socket criado à máquina (IP e porta)
- Se ativo: estabelece conexão e faz o pedido
- Se passivo: espera pedidos numa porta específica e aceita ligações na porta
- Quando termina: fecha o socket



Sockets (cont.)

- SOCK_STREAM (TCP) pode ser ativo ou passivo
- Um socket é inicialmente ativo, e se torna passivo depois de uma chamada à função **listen()**
- Somente sockets ativos podem ser referidos em chamadas à função **connect()**
- Somente sockets passivos podem ser referidos em chamadas à função **accept()**



Sockets (cont.)

- Chamadas típicas para UDP
 - `socket()`
 - `bind()`
 - `sendto()` e/ou `recvfrom()`
 - `close()`



Sockets (cont.)

- Chamadas típicas de um cliente TCP:
 - `socket()`
 - `bind()`
 - `connect()`
 - `send()` e/ou `recv()`
 - `close()`



Sockets (cont.)

- Chamadas típicas de um servidor TCP:
 - `socket()`
 - `bind()`
 - `listen()`
 - `accept()`
 - `send()` e/ou `recv()`
 - `close()`