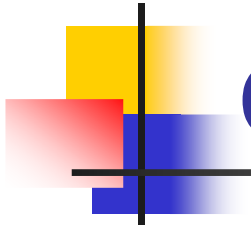
Sistemas Operacionais

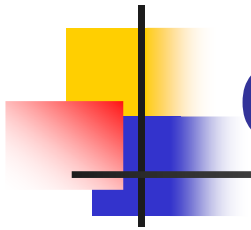
Unidade Um

Prof. Flávio Márcio de Moraes e Silva



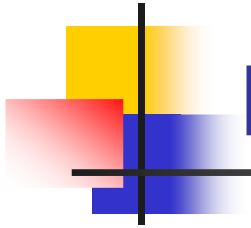
O que é um Sistema Operacional?

- O S.O. é uma camada de software colocada entre o hardware e os programas que executam tarefas para os usuários.
- É o software que controla o computador e permite a comunicação entre software e hardware.
- Um programa que age como intermediário entre o usuário e o hardware, facilitando o uso dos recursos.
- Responsável pelo acesso aos periféricos.



Objetivos do SO

- Tornar o uso do computador mais eficiente e mais conveniente.
- Eficiência – eficácia na distribuição dos recursos de hardware entre os programas. Otimizar o uso do hardware.
- Conveniência – esconder do programador detalhes relacionados ao hardware. Criando recursos de alto nível. Ex: Arquivos. Mais fácil imaginar um arquivo que um conjunto de endereços em disco.



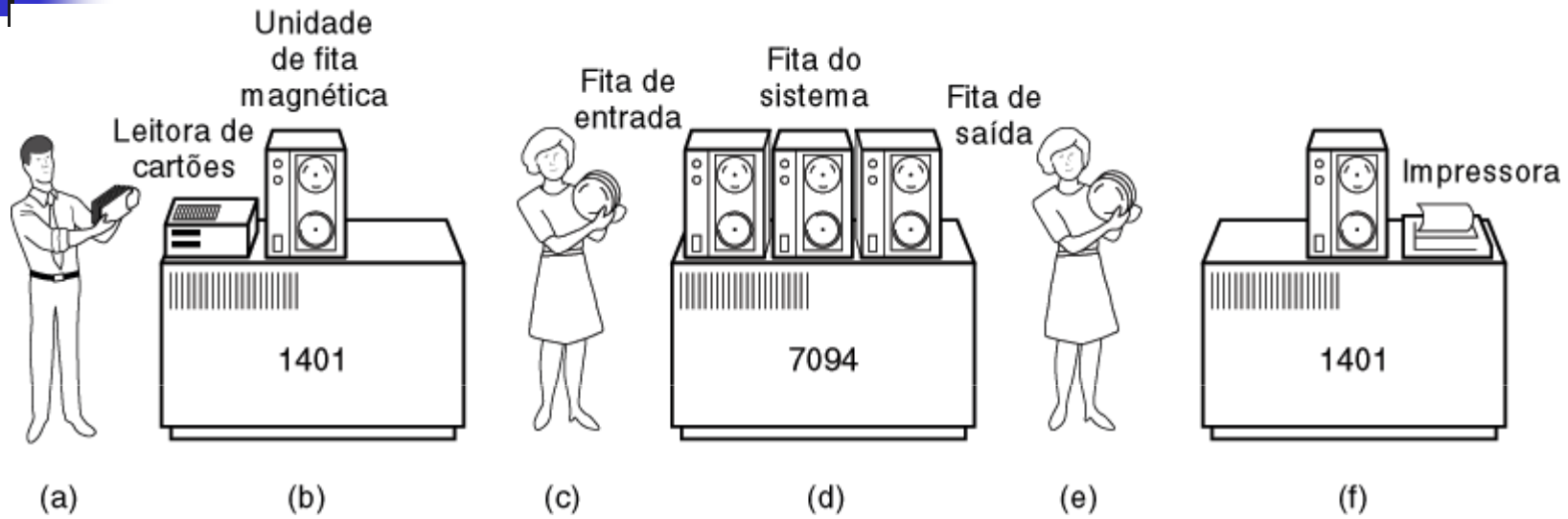
Escopo de um SO

1. Hardware – provê recursos básicos de computação (CPU, memória, dispositivos de I/O)
2. Sistema Operacional – controla e coordena o uso do hardware entre as diversas aplicações dos usuários
3. Programas Aplicativos – definem a forma com que os recursos são usados para resolverem os problemas dos usuários (compiladores, bancos de dados, editores, etc.)
4. Usuários (pessoas, dispositivos, outros computadores)

O começo da história: décadas de 1940 e 1950

- **Os sistemas operacionais passaram por diversas fases**
  - Década de 1940
    - Os primeiros computadores não dispunham de sistemas operacionais.
  - Década de 1950
    - Executavam um job (serviço) por vez.
    - Dispunham de tecnologias que facilitavam a transição de um job para outro.
    - Eram chamados de sistemas de processamento em lote de fluxo único.
    - Os programas e dados eram submetidos consecutivamente em uma fita.

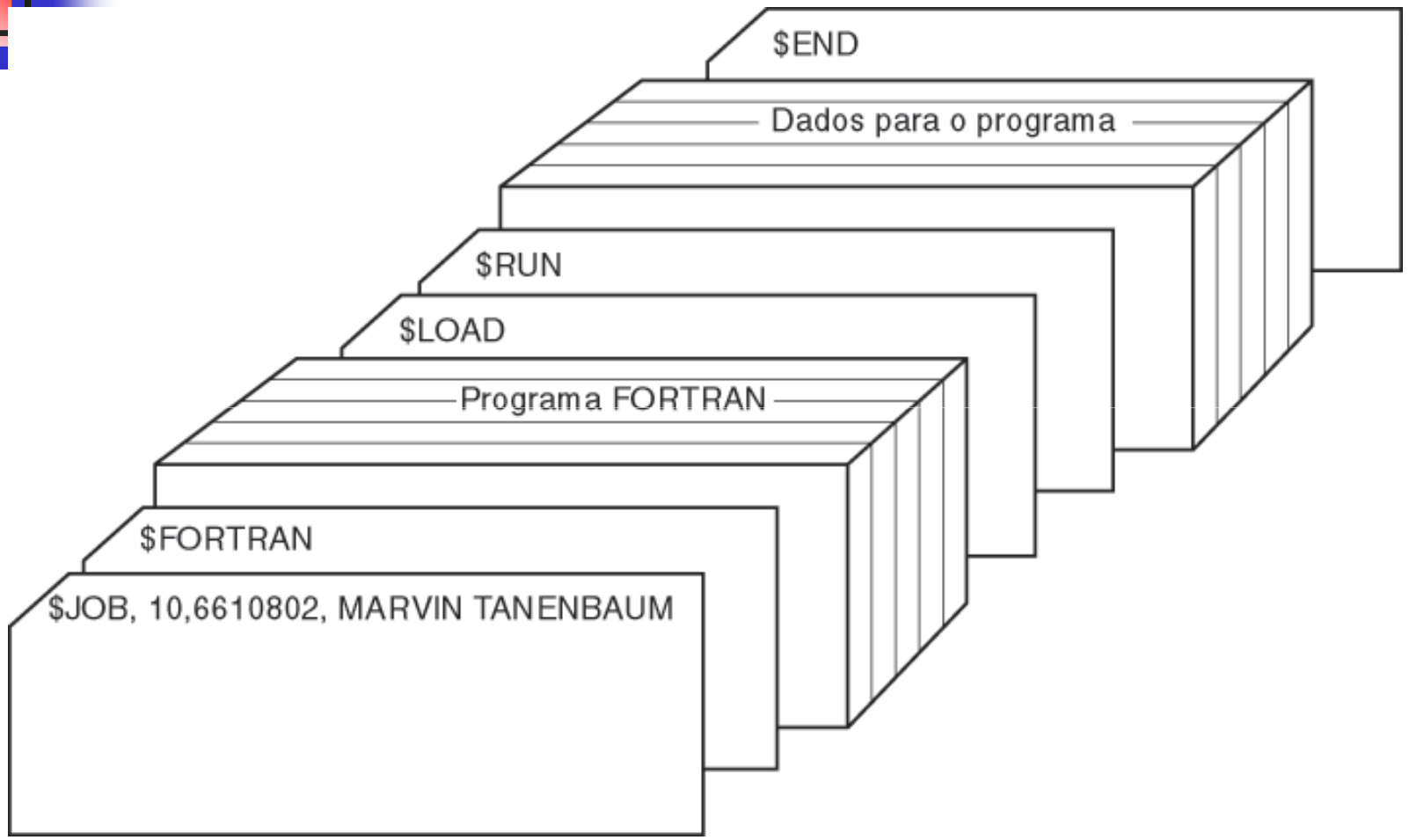
Sistema de processamento em lote



Antigo sistema em lote

- traz os cartões para o 1401
- lê os cartões para a fita
- coloca a fita no 7094 que executa o processamento
- coloca a fita no 1401 que imprime a saída

Configuração de um JOB processado em lote 1950

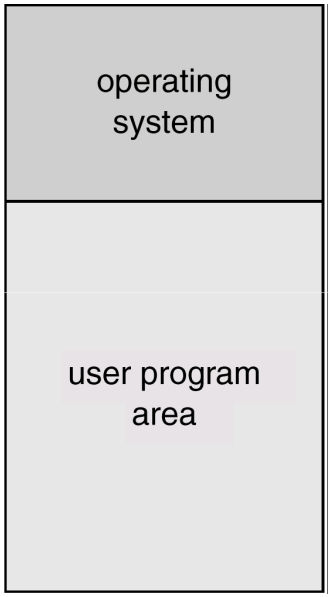


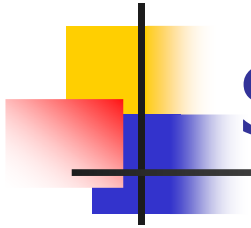
Estrutura de um job FMS típico – 2a. geração

Sistemas em Lotes

- O operador comanda a submissão de tarefas (jobs)
- Usuário ≠ operador
- Tarefa na forma de cartões perfurados
- Sequênciamento automático – controle transferido automaticamente de um job para outro. SO rudimentar.

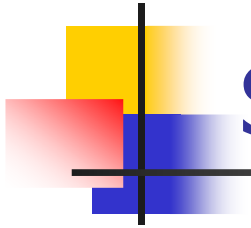
Leiaute de memória





Sistemas em Lotes

- Funcionamento
  - controle inicial
  - controle transferido para job
  - quando job termina, o controle é devolvido ao monitor
- Problemas: Baixo desempenho – operações de CPU e I/O não podiam ser sobrepostas. A CPU ficava ociosa. A leitora de cartões era muito lenta
- Solução: operação off-line – jobs carregados de unidades de fita para a memória. A leitura de cartões e tarefas de impressão eram feitas off-line

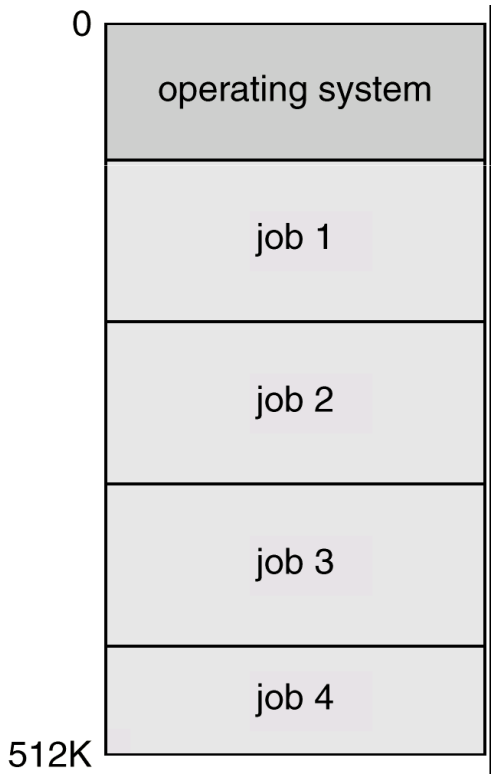


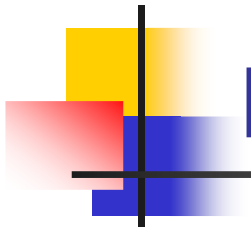
Sistemas em Lotes

- Spooling: Permite a sobreposição da computação de um job com o I/O de outro. Enquanto executa um job, o SO:
  - Lê o próximo job da leitora para o disco (fila de jobs)
  - Imprime a saída do job anterior, do disco para a impressora
- *Job pool:* estrutura de dados que permite selecionar qual job será executado a seguir, para aumentar o uso da CPU

Sistemas em *Batch* Multiprogramados

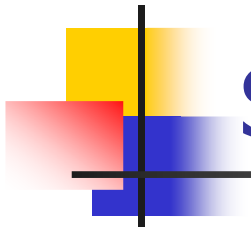
- Vários jobs são mantidos na memória principal ao mesmo tempo e a CPU é multiplexada





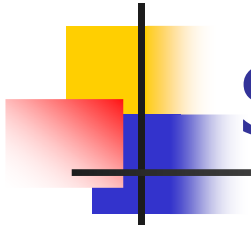
Multiprogramação

- Características do SO:
  - Rotinas de I/O provisionadas pelo sistema
  - Gerência de memória – o sistema deve alocar memória para vários jobs
  - Escalonamento de CPU – o sistema deve escolher dentre os diversos jobs prontos para serem executados
  - Alocação de dispositivos



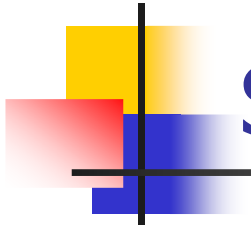
Sistemas de Tempo Compartilhado

- A CPU é multiplexada entre diversos jobs mantidos em memória
  - a CPU é alocada apenas para jobs existentes na memória
- O usuário tem a impressão que a CPU está dedicada
- Um job é transferido do disco para a memória e da memória para o disco (***swap***)
- Há Comunicação on-line entre o usuário e o sistema
  - Quando o SO termina a execução de um comando, exibe a resposta na tela.
  - Aguarda o próximo comando do teclado



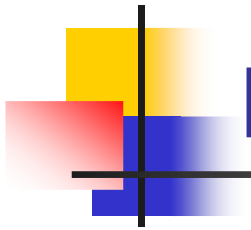
Sistemas de Computadores Pessoais

- *PC* – sistema de computação dedicado a um único usuário
- Dispositivos de I/O – teclado, mouse, vídeo, impressoras
- Praticidade e tempo de resposta
- Foram capazes de adotar tecnologia desenvolvida para sistemas de grande porte. O uso pessoal não exige sofisticação no gerenciamento da CPU nem em aspectos de proteção



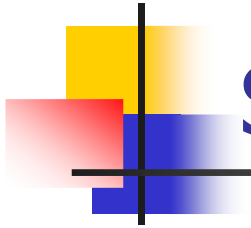
Sistemas de Computadores Pessoais

- A utilização de PCs por vários usuários e acesso à rede trazem novas necessidades de proteção e segurança
- Evolução para as estações de trabalho



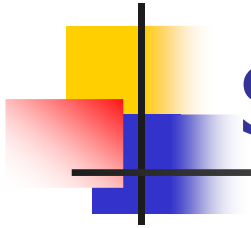
Bases de Aplicações

- API (Interface de Programação de Aplicativos): fornece as chamadas do sistema
- O DOS liberou os programadores de detalhes complicados da manipulação do hardware como manipulação de memória, I/O, comunicação etc.
  - Possibilitou o surgimento de fornecedores independentes de software
- Base de aplicação: combinação do hardware com o ambiente do sistema operacional. Ex: desenvolvimento para Windows em máquinas Cisc.



Sistemas Paralelos

- Sistemas de multiprocessadores com mais de uma CPU em comunicação ativa
  - Compartilhando barramento, *clock* e posivelmente memória e dispositivos
- *Sistemas fortemente acoplados* – processadores compartilham memória e *clock*
  - A comunicação é feita através de memória compartilhada

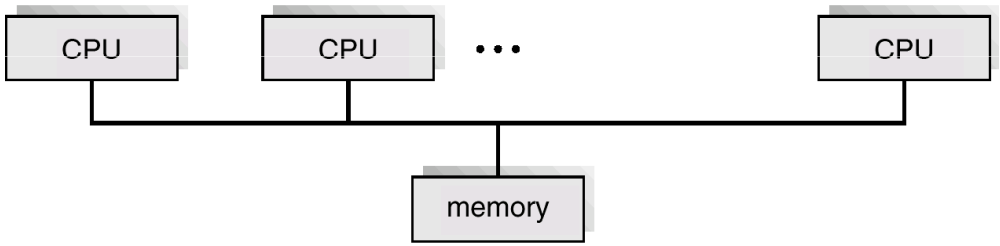


Sistemas Paralelos

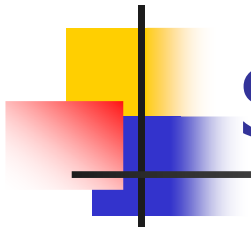
- Vantagens:
  - Maior produção (*throughput)*
  - Economia de recursos
  - Confiabilidade
    - degradação normal - a falha de um processador não interrompe o processo
    - sistemas tolerantes a falhas - duplicação de hardware e software

Sistemas Paralelos

- *Multiprocessamento simétrico*
  - Cada processador executa uma cópia idêntica do SO
  - Vários processos podem ser executados simultaneamente sem queda de desempenho

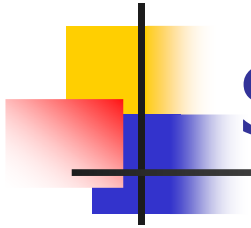


- *Multiprocessamento assimétrico*
  - cada processador é alocado a uma tarefa específica. Processadores-mestres escalonam e alocam tarefas aos demais processadores-escravos
  - Mais comuns em grandes sistemas



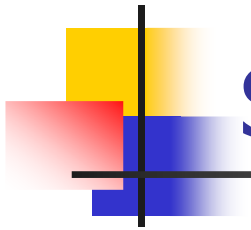
Sistemas de Tempo Real

- Usados como controle em aplicações dedicadas
  - Ex.: controle de experimentos científicos, sistemas de imagens médicas, sistemas de controle industrial, etc.
- *Sistemas de Tempo Real Críticos*
  - Armazenamento secundário limitado ou ausente. Dados armazenados em RAM ou ROM
  - Possui tratamento para conflitos
- *Sistemas de Tempo Real Não-Críticos*
  - Não indicados para aplicação industrial e robótica
  - Úteis em aplicações que requerem características avançadas (multimídia, realidade virtual)



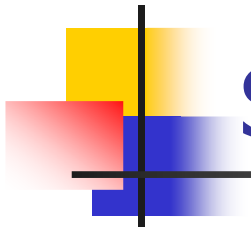
Sistemas Distribuídos

- Distribuem a computação entre vários processadores físicos
- *Sistemas fracamente acoplados* – cada processador tem sua própria memória local
  - Processadores se comunicam através de linhas telefônicas e redes de alta velocidade
- Vantagens:
  - Compartilhamento de recursos
  - Velocidade de computação
  - Confiabilidade
  - Intercomunicação



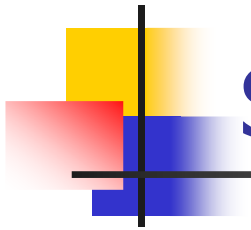
Sistemas Distribuídos

- Sistemas Operacionais de Rede
  - Proporcionam compartilhamento de arquivos
  - Gerenciam a comunicação
  - Executam de forma independente de outros computadores da rede
- Sistemas Operacionais Distribuídos
  - Menos autonomia entre computadores
  - Dão a impressão que só existe um SO controlando a rede



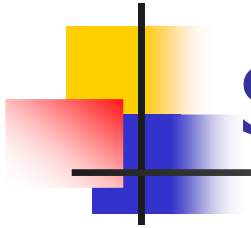
Serviços de um SO

- Serviços Básicos:
  - **Execução de programas:** meios para carregar o programa na memória principal e executá-lo
  - **Sistema de arquivos:** capacidade de ler, gravar e excluir arquivos
  - **Operações de I/O:** uma vez que os programas usuários não podem executar operações de I/O diretamente, o SO deve prover esta funcionalidade
    - Interface de acesso aos periféricos (impressoras, discos, fitas, etc)



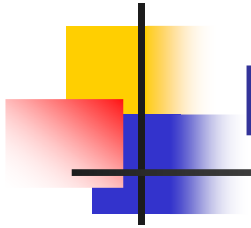
Serviços de um SO

- **Gerência de memória:** determinar quando e como a memória deve ser utilizada
- **Comunicação:** troca de informações entre processos executando na mesma máquina ou em sistemas conectados. Implementado através de memória compartilhada ou passagem de mensagens
- **Detecção de erros**: garantir a computação correta, detectando erros na CPU, hardware de memória, dispositivos de I/O e programas de usuários



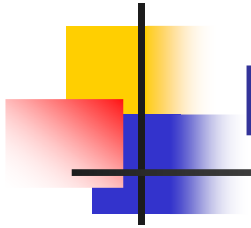
Serviços de um SO

- Depende do SO utilizado
- Serviços Avançados
  - Mecanismo de monitoração de recursos, capazes de identificar possíveis gargalos no sistema
  - Meios para armazenar e manter o estado do sistema
  - Mecanismos de compartilhamento de hardware (compartilhar o mesmo micro por usuários em rede)
    - Necessidade de algum tipo de proteção



Funções do *Kernel*

- Escalonador de processos
  - Determina quando e por quanto tempo um processo é executado em um processador
- Gerenciador de memória
  - Determina quando e como a memória é alocada aos processos
  - Determina o que fazer quando a memória principal estiver cheia
- Gerenciador de E/S
  - Atende às solicitações de entrada/saída de e para dispositivos de hardware

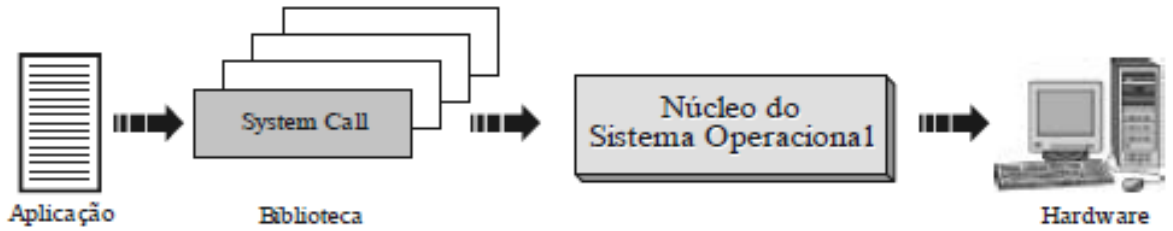


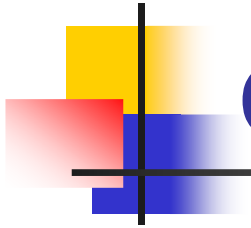
Funções do *Kernel*

- Gerenciador de comunicação interprocessos (IPC)
  - Permite que os processo se comuniquem uns com os outros
- Gerenciador de sistema de arquivos
  - Organiza coleções nomeadas de dados em dispositivos de armazenamento
  - Fornece uma interface para acessar os dados armazenados

Chamadas ao Sistema (*System Calls*)

- Programas solicitam serviços ao SO através delas.
  - Semelhantes às chamadas de "funções", a diferença é que as chamadas de sistema transferem a execução para o SO
  - Rotinas altamente dependentes do hardware
    - Normalmente disponíveis como instruções assembly
- As chamadas se diferem de SO para SO, no entanto, os conceitos relacionados às chamadas são similares independentemente do SO



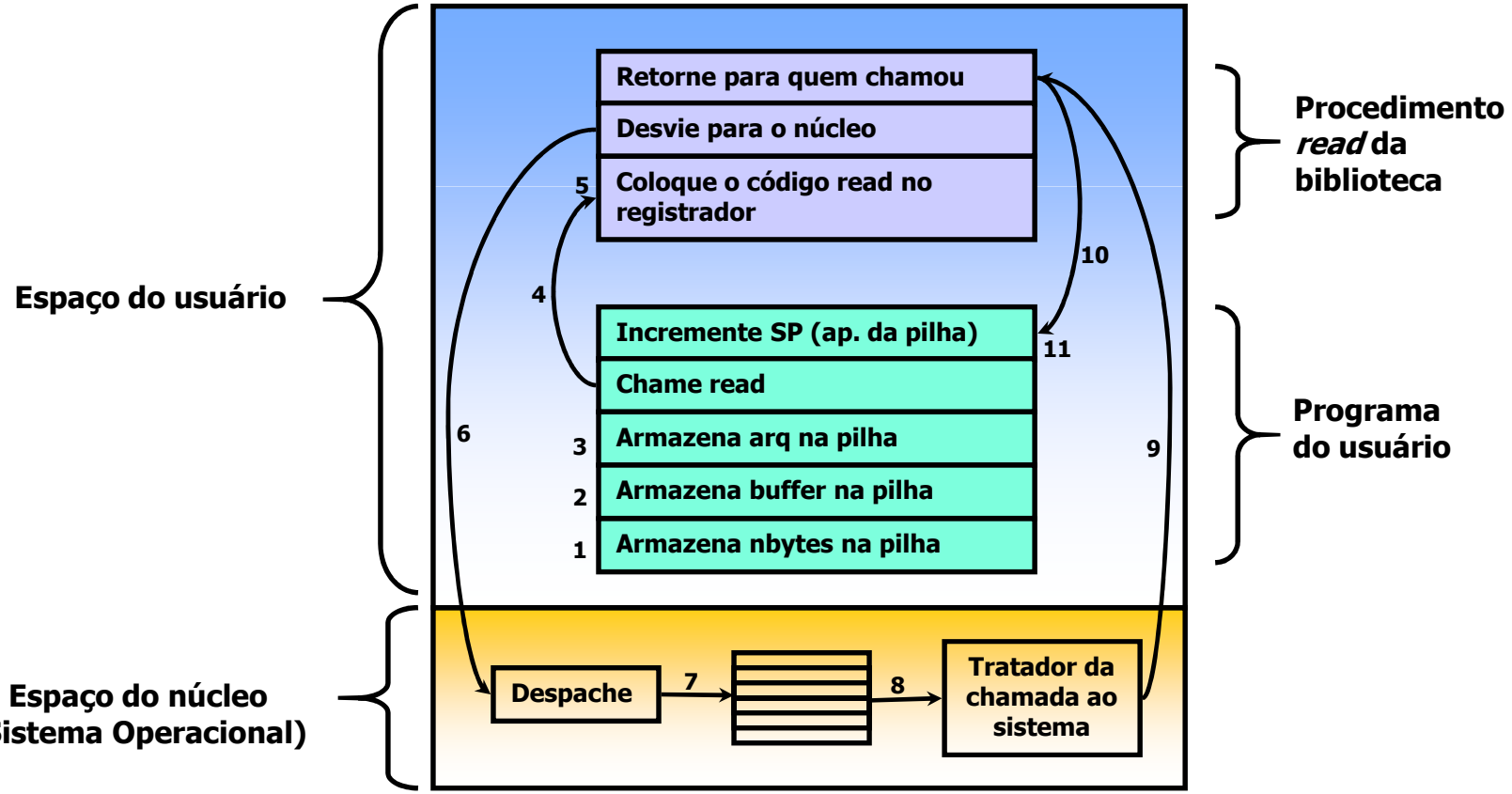


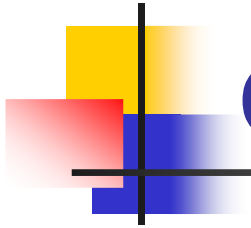
Chamadas ao Sistema

- Apenas uma chamada de sistema pode ser realizada em um instante de tempo (ciclo de relógio) pela CPU
  - Quando um aplicativo precisar ler um dado de um arquivo, terá que executar uma instrução TRAP ao SO
    - TRAP: instrução utilizada para realizar chamadas e transferir o controle ao SO
- Através de parâmetros o programa do usuário informa o que necessita.
- O retorno de uma chamada ao sistema faz com que o programa retome a execução, normalmente, do ponto em que parou.

Chamadas ao Sistema

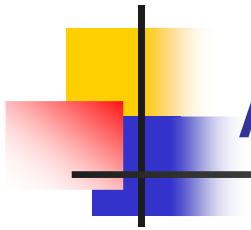
- Exemplo de uma chamada read (para ler um arquivo)
  - Contador = read(arq, buffer, nbytes);





Chamadas ao Sistema

- Em linguagens de alto nível as chamadas ao sistema ficam escondidas em bibliotecas utilizadas pelo compilador, que gera o código assembly necessário
- Via assembly geralmente, 3 métodos são utilizados para passar parâmetros ao SO:
  - Colocá-los em *registradores*
  - Guardá-los em tabelas na memória, cujo endereço fica em um registrador
  - Empilhar os parâmetros na pilha do sistema

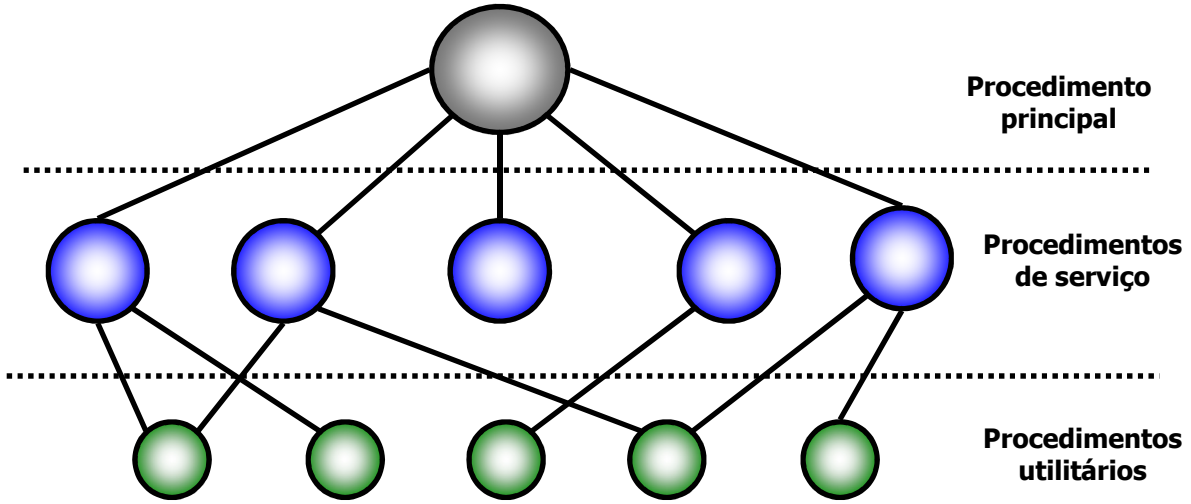


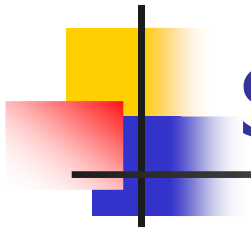
Arquitetura Monolítica

- Não há estruturação (“a grande bagunça”)
- O SO é escrito como uma coleção de procedimentos
  - Um procedimento pode chamar qualquer outro sempre que necessário
- Possuem uma interface bem definida quanto a parâmetros e resultados
- Há uma certa organização
  - Os serviços são requisitados colocando-se os parâmetros em um local bem definido (pilha) e a seguir se executa uma instrução de desvio de controle (trap)
  - O SO assume o controle, busca os parâmetros e determina qual chamada ao sistema executará

Arquitetura Monolítica

- Estrutura básica para um SO monolítico
  - Programa principal
    - Invoca o procedimento do serviço requisitado
  - Conjunto de procedimentos de serviço
    - Executam as chamadas ao sistema
  - Conjunto de procedimentos utilitários
    - Auxiliam os procedimentos de serviço



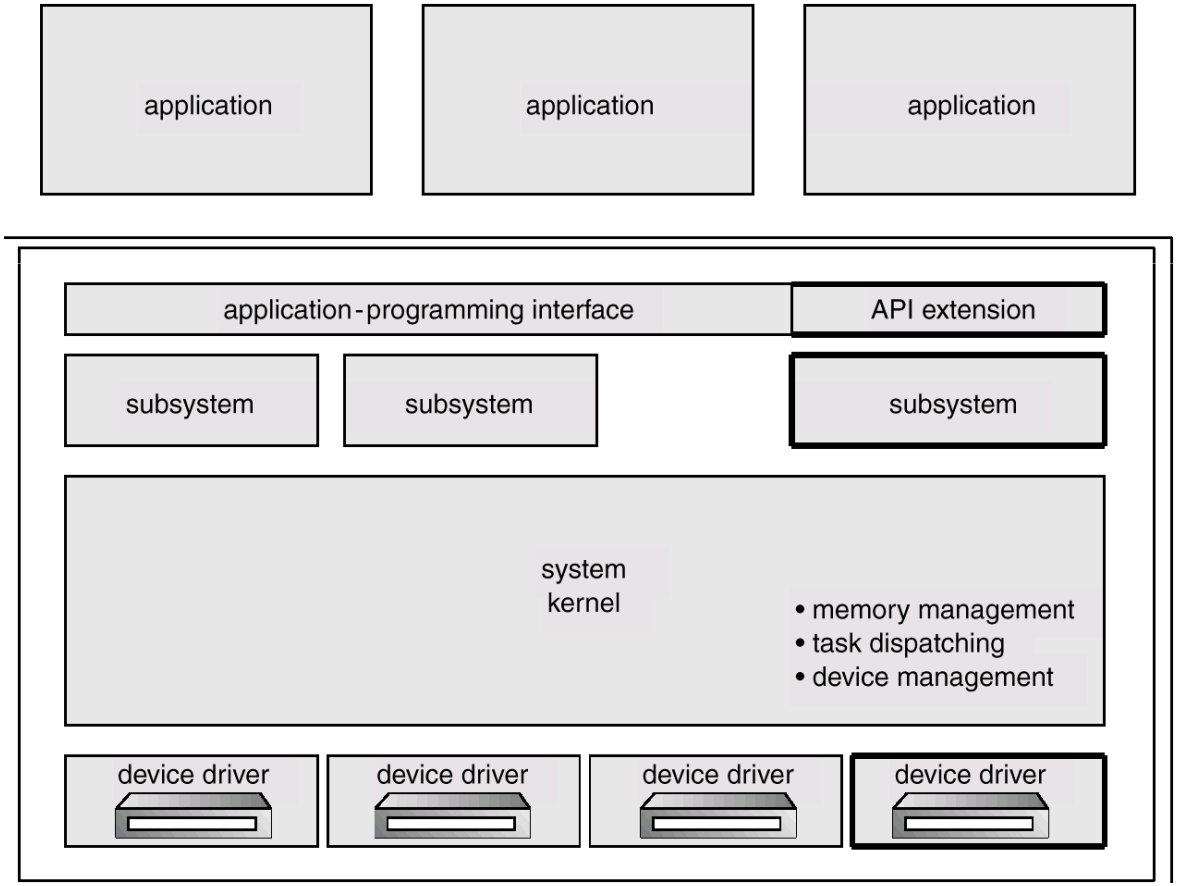


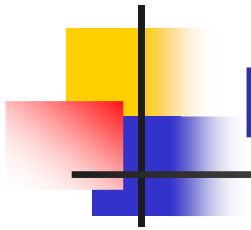
Sistemas em Camadas

- O SO é dividido (organizado) em uma hierarquia de camadas
  - A camada mais baixa (nível 0) é o hardware
  - A camada mais alta (nível N) é a interface com o usuário
- Cada camada usa funções e serviços apenas das camadas inferiores

Sistemas em Camadas

- Estrutura de Camadas do OS/2

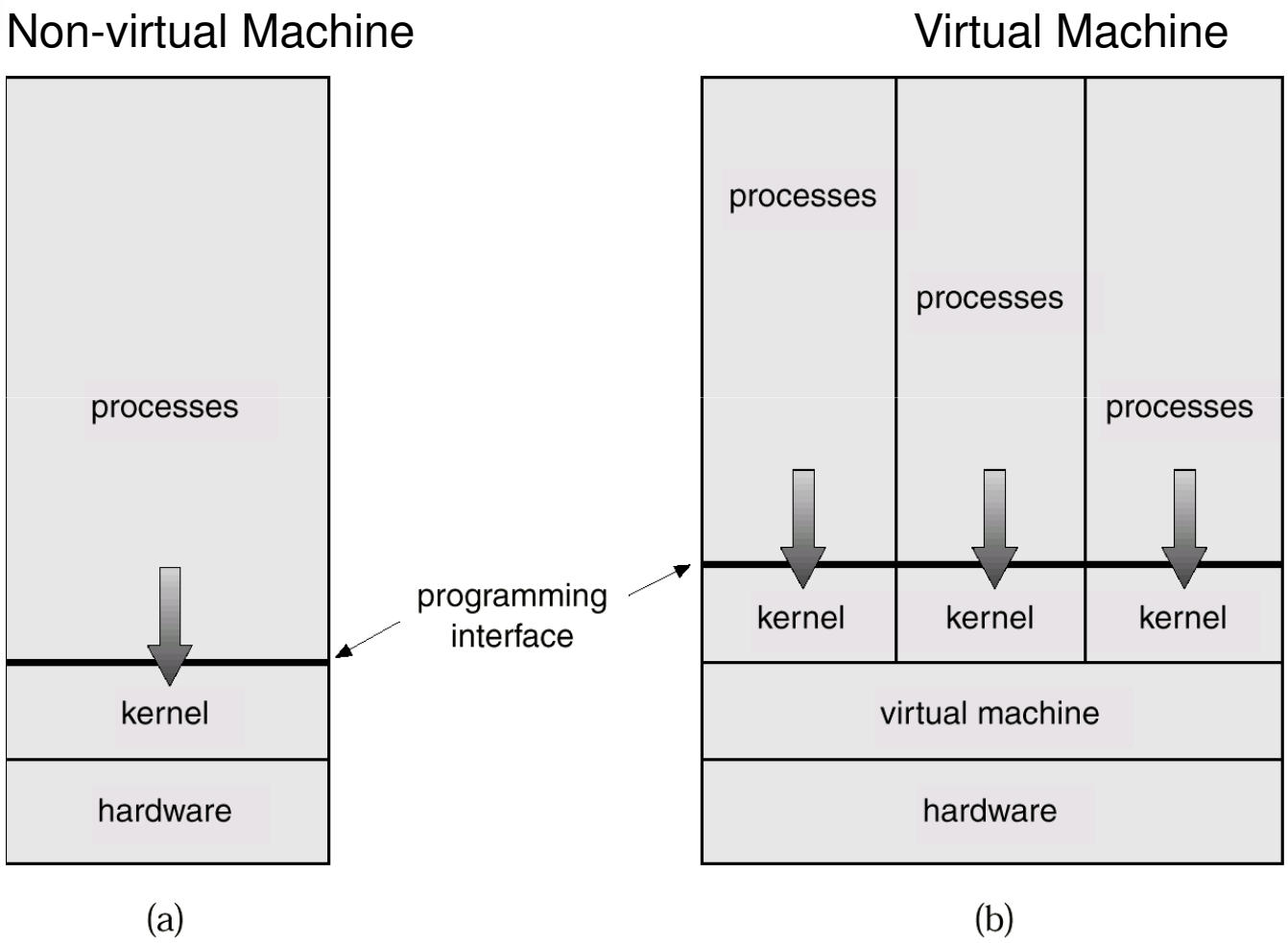


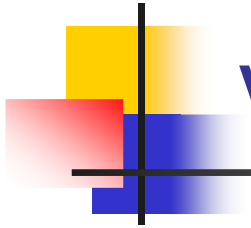


Máquinas Virtuais

- Uma *máquina virtual* se baseia na implementação de sistemas por camadas. Simula em software o hardware e o disponibiliza a um kernel de SO.
- Podem ser criadas várias máquinas virtuais (cópias).
- Cria a ilusão de múltiplos computadores, cada qual executando o seu SO em seu próprio processador, com sua própria memória (virtual).

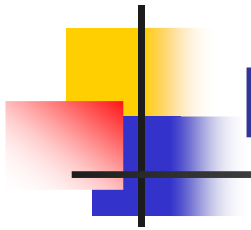
Modelos de Sistemas





Vantagens/Desvantagens de VMs

- O conceito de VM oferece completa proteção dos recursos do sistema, já que cada VM é isolada das demais
  - No entanto, impede o compartilhamento direto de recursos
- Um sistema de VMs é perfeito para desenvolvimento e pesquisa em SO já que desenvolver em uma VM não afeta as demais
- O conceito de VM é difícil de implementar devido ao esforço necessário para oferecer uma duplicata exata da máquina original



Microkernels

- Microkernels: modularização do kernel (CMU). Componentes não essenciais são implementados como programas do sistema
- Principal função do kernel é fazer a comunicação entre os programas clientes e os serviços disponíveis
- Vantagem: facilidade de expandir o SO - novos serviços são adicionados com mínima alteração do kernel
- Sistemas baseados em microkernels: Apple MacOS, Windows NT, UNIX Digital