Software Básico

Sistemas Operacionais

Tanembaum, capítulo 6

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Sistema Operacional

- Nível de abstração visto pelo usuário
- Programa gerente da máquina
- Estende a arquitetura com novos serviços (*system calls*)
- Simplifica acessos a recursos do *hardware*

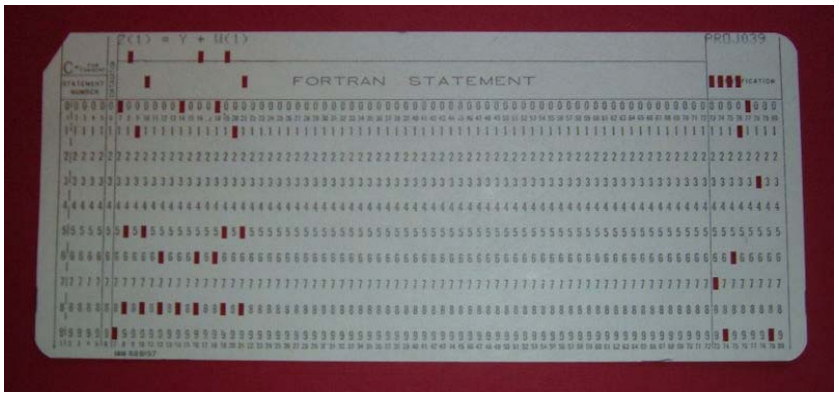
Evolução dos sistemas operacionais

- Primeiros computadores: programação direta do hardware com chaves e cabos

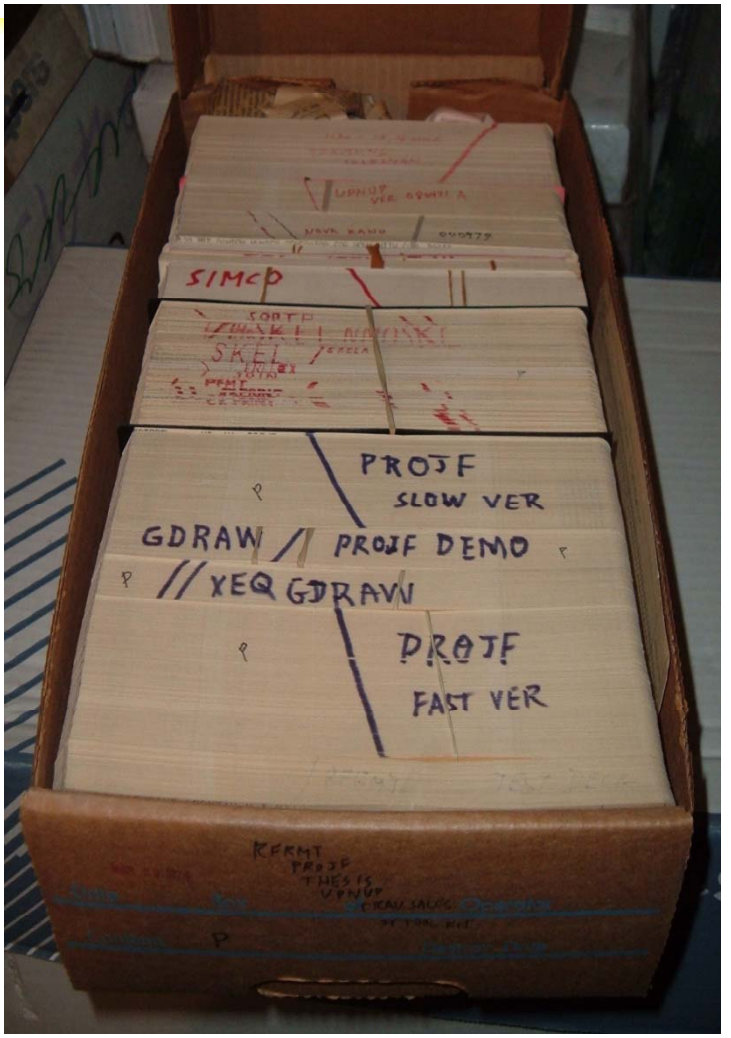
ENIAC, 1946

Evolução dos sistemas operacionais

- A programação evoluiu:
  - passou a ser com feita com cartões perfurados



- Mas o programador ainda tinha que controlar o computador diretamente
  - Monitor de carga



Evolução dos sistemas operacionais

- Processamento em *batch* ou lote
  - Programadores perderam o acesso ao computador

IBM 709, 1959

Evolução dos sistemas operacionais

- Coleta remota e batch multiprogramado
  - Computadores menores passaram a ser usados para “montar” os *batches em grupos*

IBM 1401, 1961

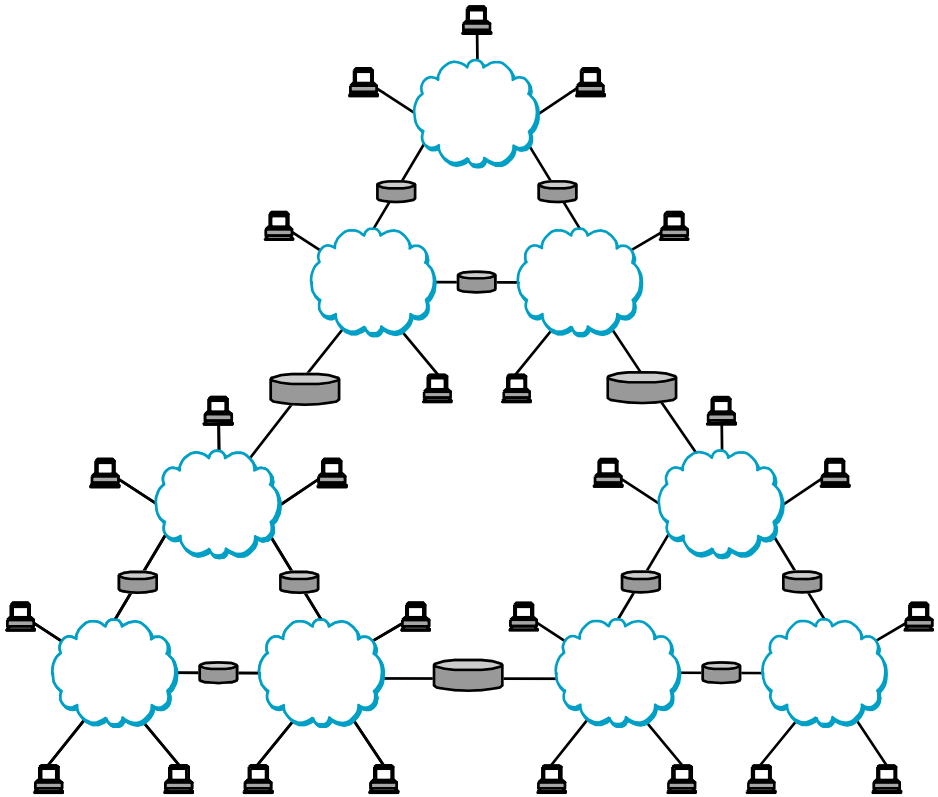
Evolução dos sistemas operacionais

- Terminal remoto/tempo compartilhado
  - O programador ganhou de novo acesso ao computador

IBM 3278, 1972

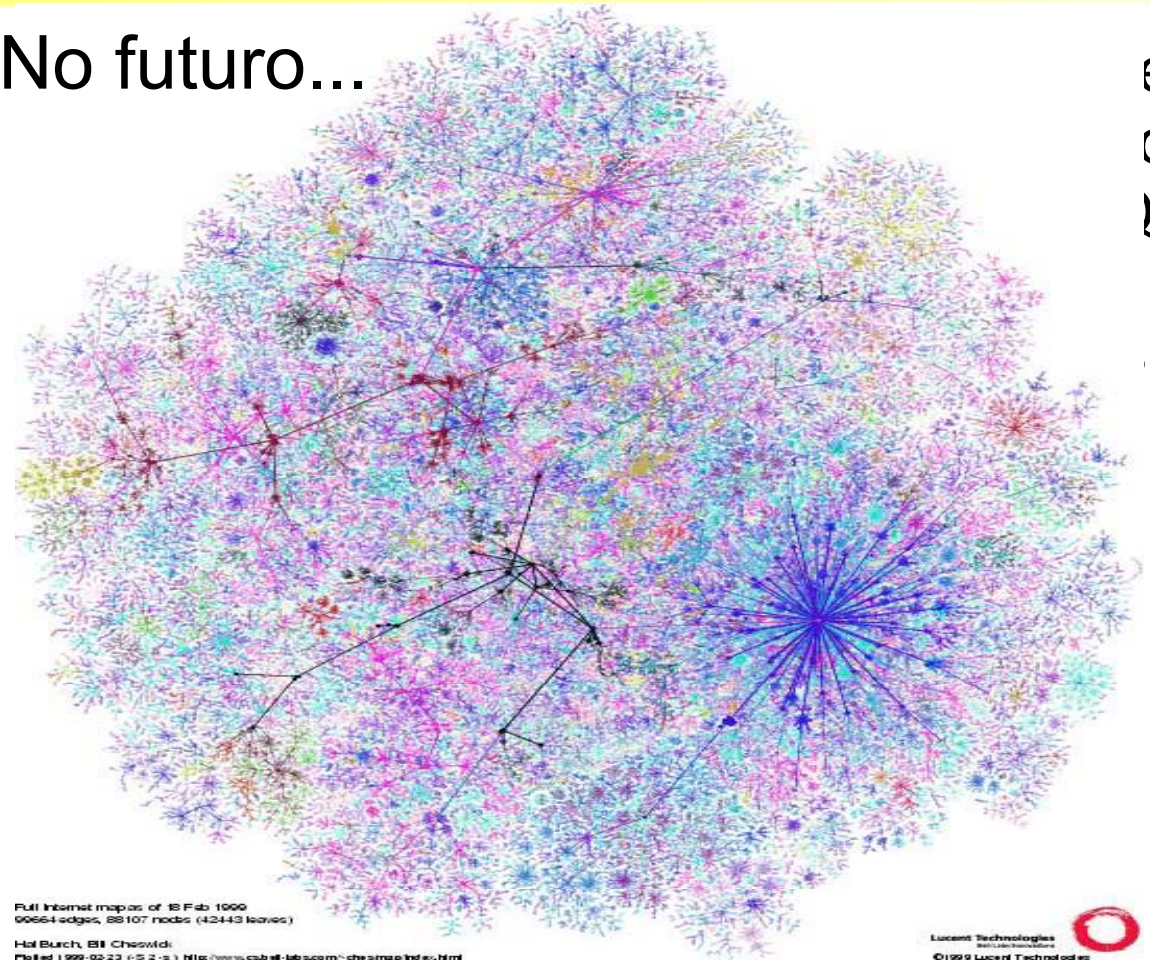
Evolução dos sistemas operacionais

- Sistemas pessoais distribuídos



Evolução dos sistemas operacionais

- No futuro...



empo real
olerância a falhas
oS
– contabilidade
egurança

http://www.geog.ucl.ac.uk/casa/martin/atlas/topology.html

Evolução dos sistemas operacionais

Moldura IP para retratos
http://www.ceiva.com/

Torradeira que imprime infos da Web no pão
http://g1.globo.com/Noticias/Tecnologia/0,,MUL758027-6174,00
TORRADEIRA+IMPRIME+IMAGENS+E+NOTICIAS+EM+PEDACO+DE+PAO.html

Estrutura do Sistema Operacional

- Gerência da memória
- Controle de processos
- Sistema de entrada e saída (E/S)
- Acesso a arquivos
- Segurança de dados
- Interpretador de comandos

Chamadas do sistema (system calls)

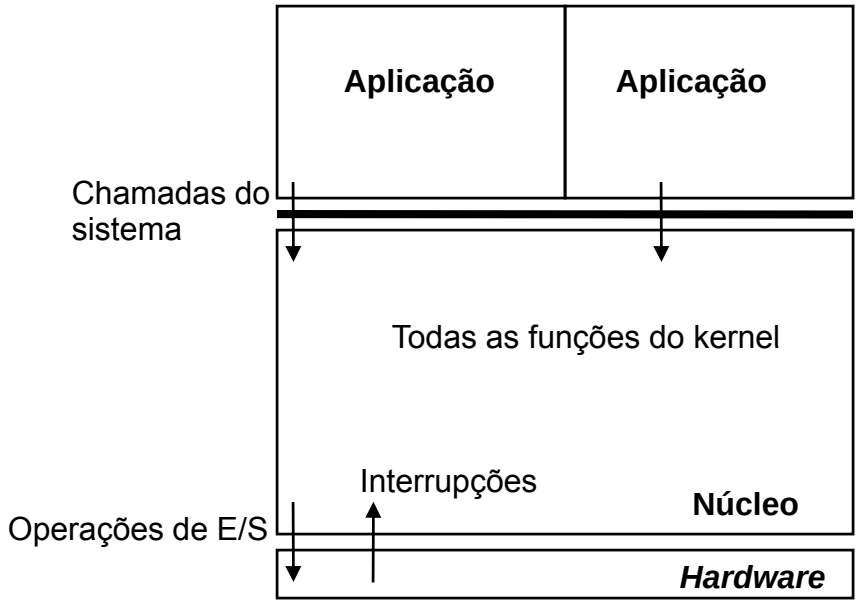
- Interface entre programas e o núcleo do S.O.
  - Controle de processos
  - Manipulação de arquivos
  - Manipulação de dispositivos
  - Manutenção de informação (tempo, atributos de processo, arquivos)
  - Comunicação
- Mudança de contexto usuário $\rightarrow$ kernel
- Implementadas como *traps*
- Parâmetros passados em uma área especial
- Tipos dependem da arquitetura do sistema

Modelos de organização

- Simples
- Em camadas
- Máquina virtual
- Micro-kernel

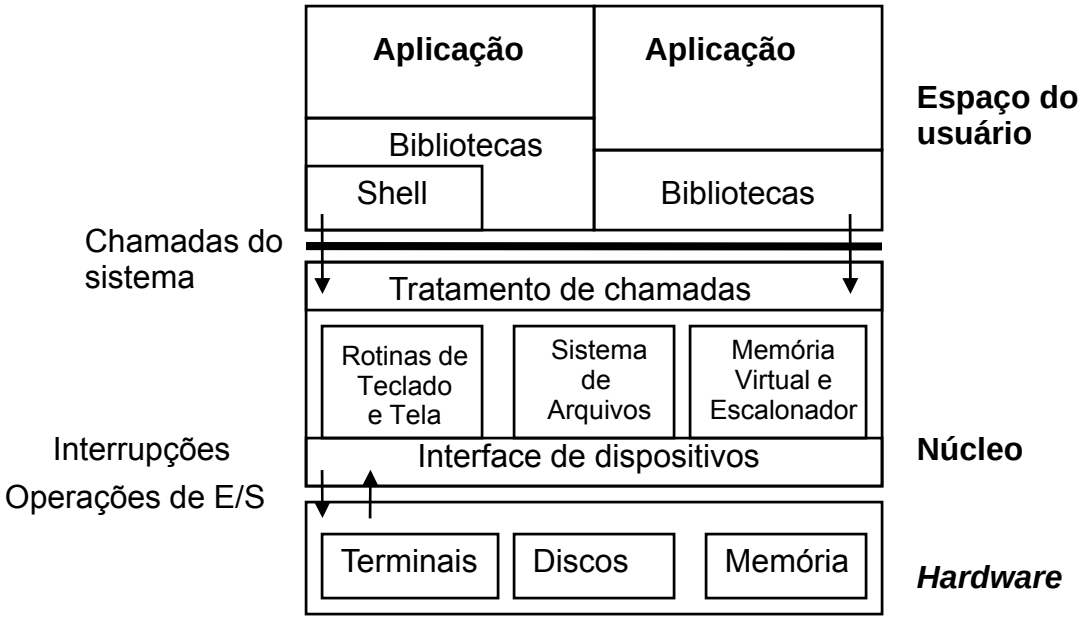
Sistema operacional simples

- Funcionalidades mal definidas
- Todos os serviços no núcleo



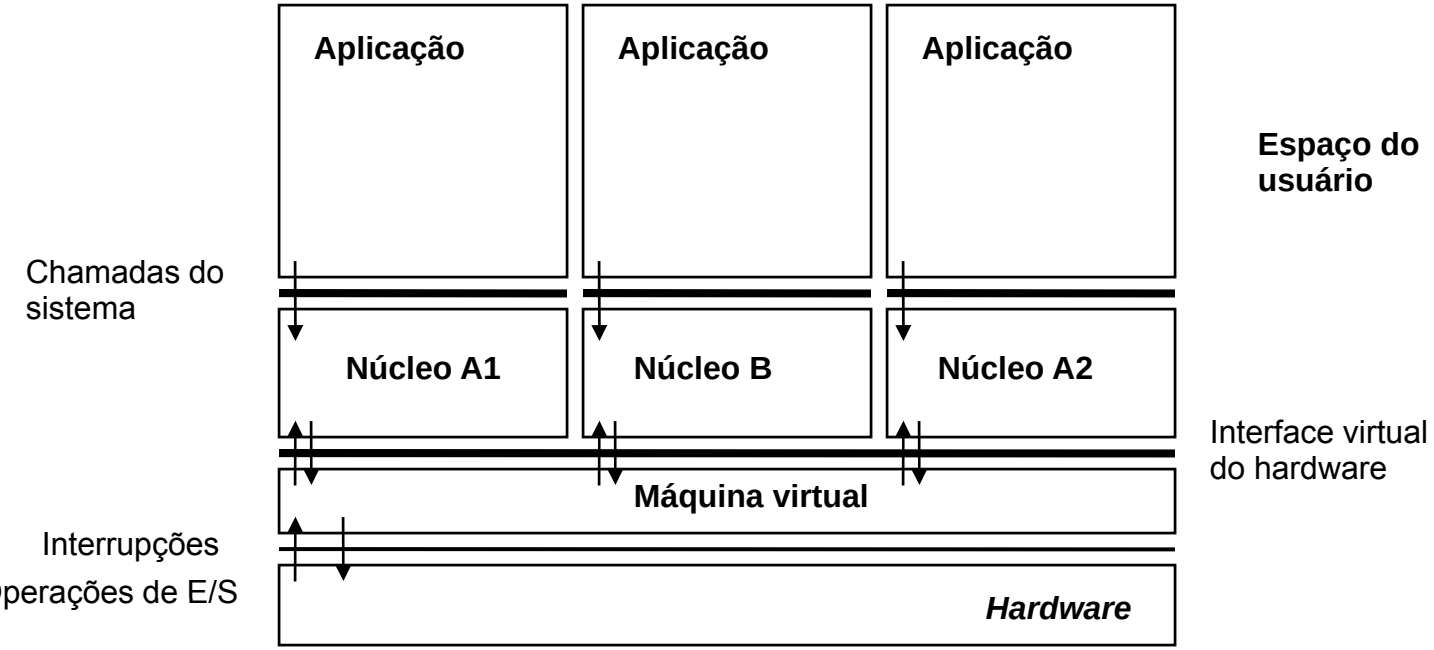
Sistema operacional em camadas

- Interfaces bem definidas
- Funcionalidades distribuídas

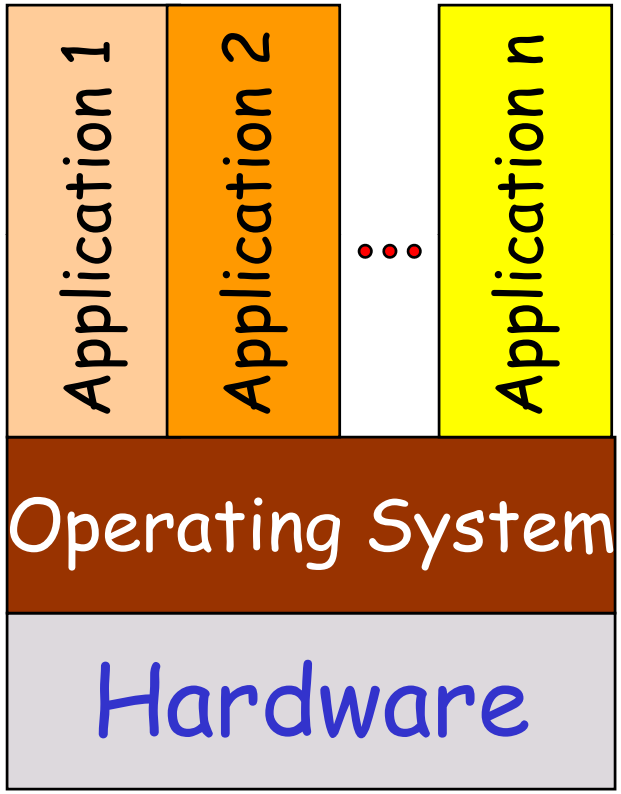


Máquinas virtuais

- Isolamento maior do hardware
- Vários sistemas podem usar a máq. virtual



Virtualization: Basic Concepts

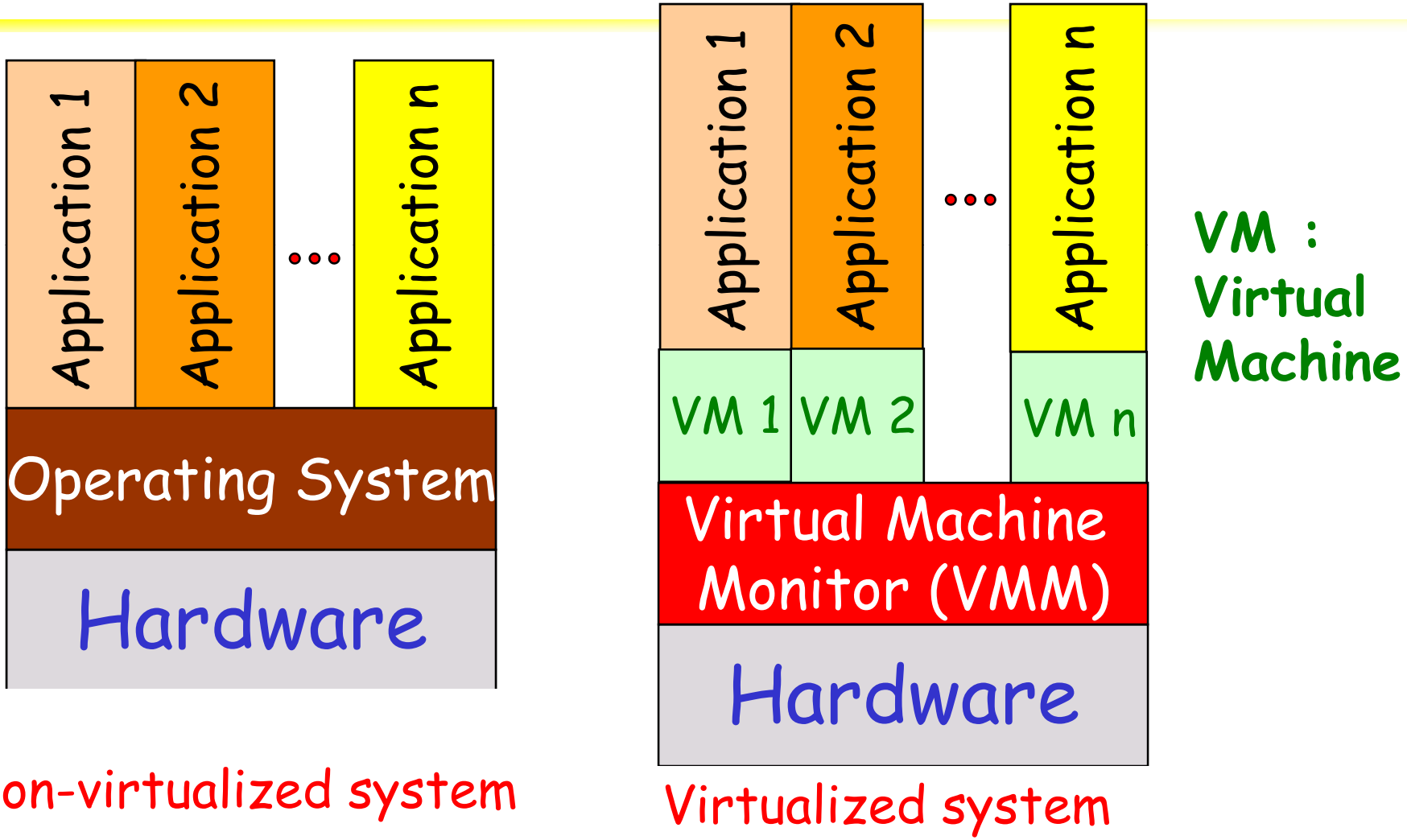


Non-virtualized system

- OS controls access to hardware
- Operation modes: user and kernel
- Only non-privileged instructions can be executed in user mode
  - do not change allocation/state of any resource
- Applications run in user mode.
- A privileged instruction causes an interruption and a switch to kernel model (OS)

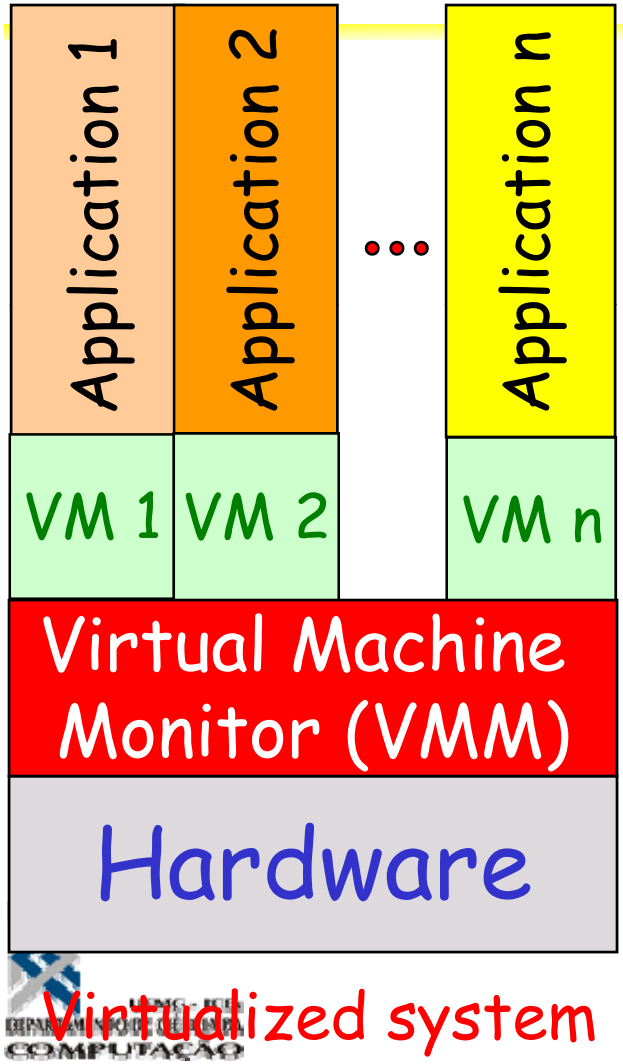
Source: Based on slides from "Virtualization: Concepts, Applications, and Performance Modeling", by Daniel A. Menasce

Virtualization: Basic Concepts



Source: Based on slides from "Virtualization: Concepts, Applications, and Performance Modeling", by Daniel A. Menasce

Virtualization: Basic Concepts

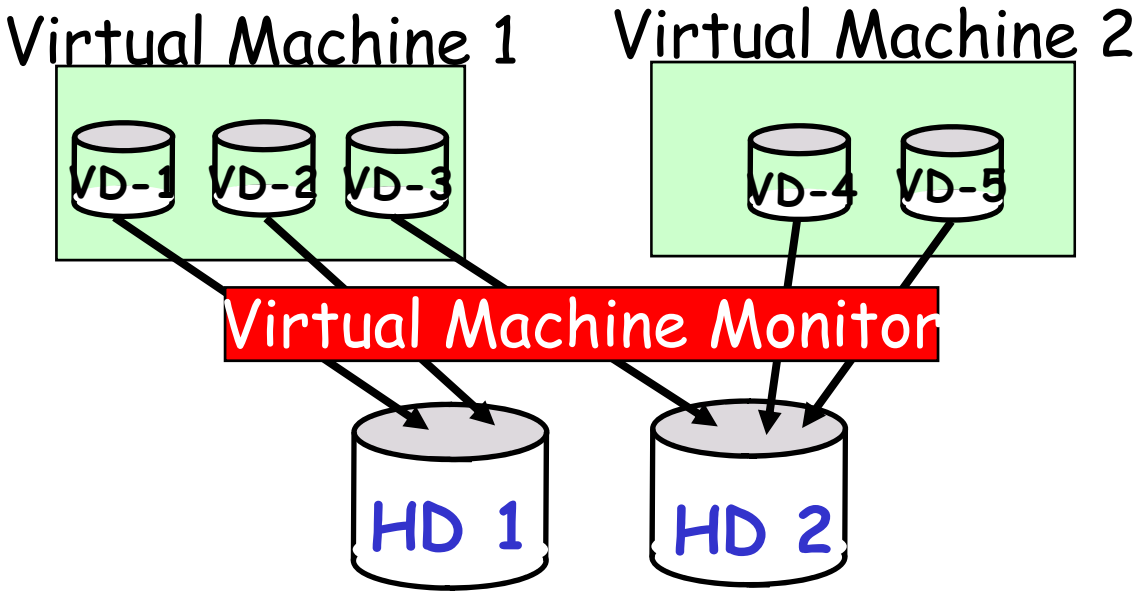


- Virtual Machines (VMs): virtual instances of physical resources, working at fraction of physical capacity
- Guest OSs, executed on VMs, run in user mode
- VMM runs in kernel mode and controls access to hardware
- All privileged instructions executed by a VM on behalf of an application are handled by VMM

Source: Based on slides from "Virtualization: Concepts, Applications, and Performance Modeling", by Daniel A. Menasce

Virtualization: Basic Concepts

- Performance isolation and service differentiation
  - Each VM is a dedicated and isolated server
- Flexible and transparent capacity expansion and contraction



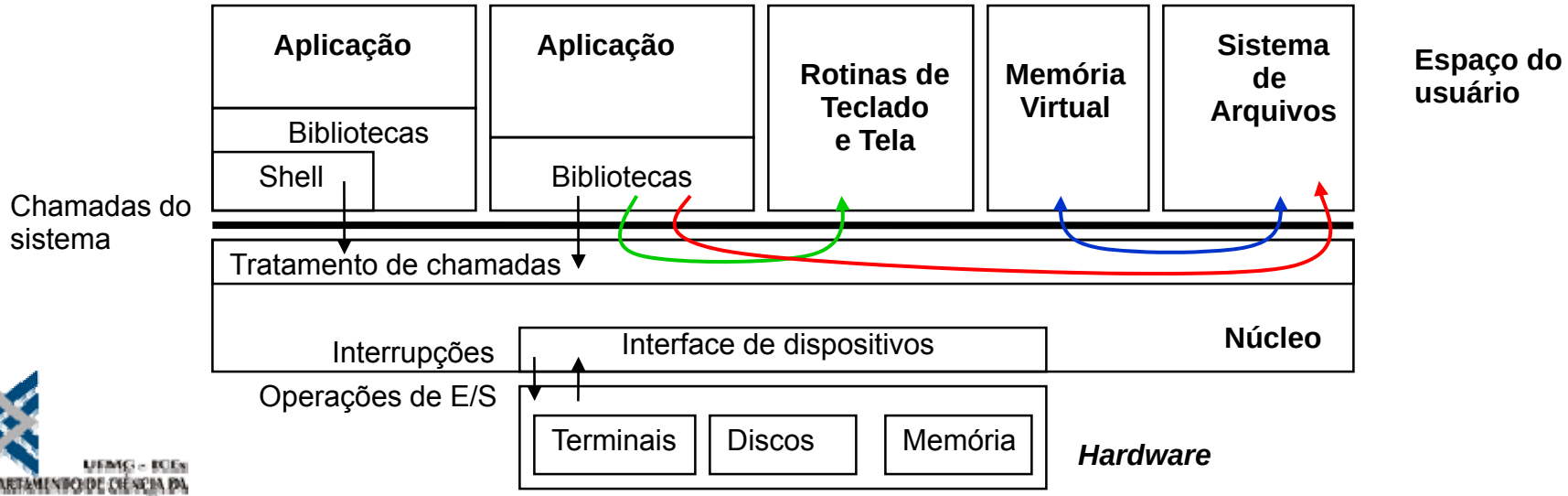
Same for all other resources in physical infrastructure
(CPU, network, etc)

Virtualization: A little of history

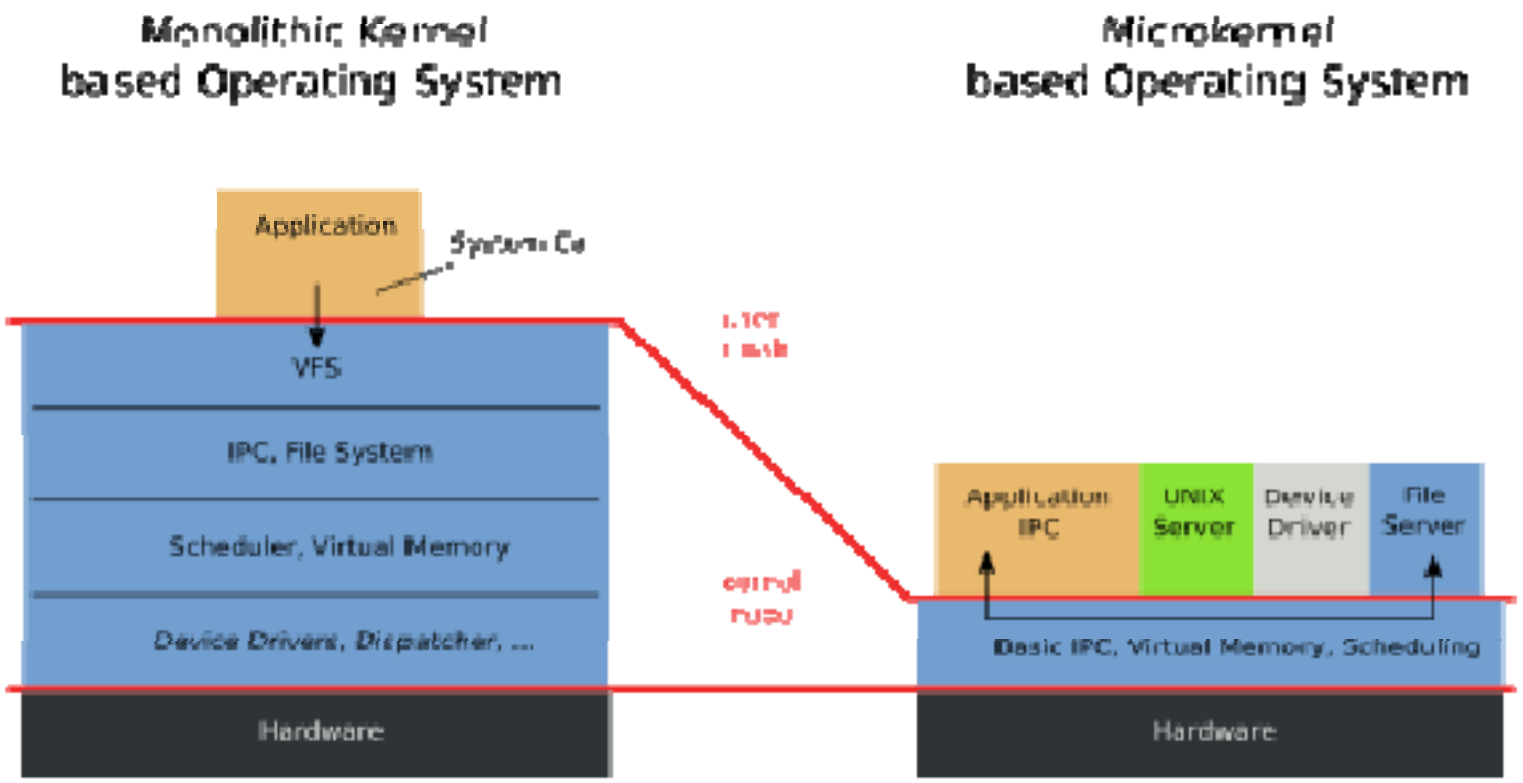
- Back in the 70's: IBM's VM/370
  - Virtualization was a hot topic for mainframes!!!
- 80's : hardware costs go down
  - Mainframes replaced by PCs
  - Virtualization faded away
- 90's: client-server and P2P networks
  - Issues: reliability, security, high costs
- Nowadays: virtualization is back in fashion!!!
  - Xen (open source), Denali, EMC's VMWare,
  - Intel's virtualization technology: hardware support
  - IBM's Logical Partitioning (LPAR), etc

Micro-kernel

- *Camadas removidas do kernel*
- *Serviços básicos entre processos e hardware*
- *Serviços do SO implementados como servidores (nível de usuário)*
- *Chamadas executadas como IPC para servidores*



Micro-kernel

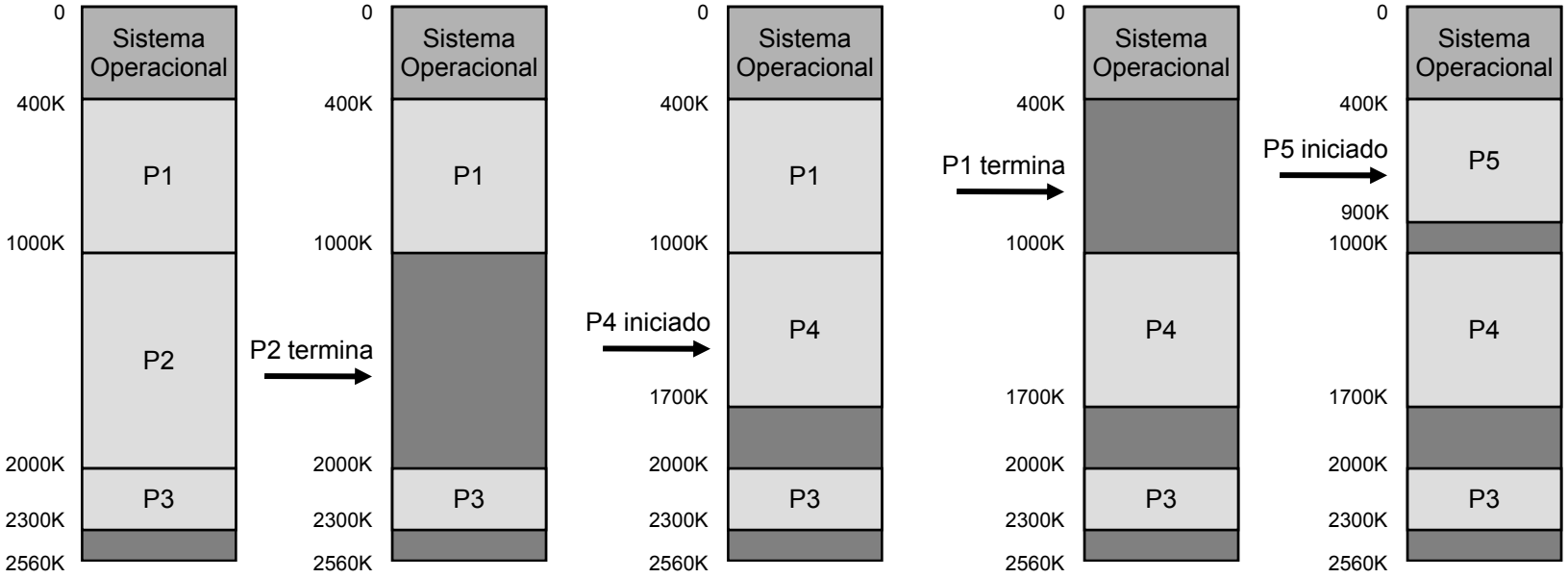


Gerência de memória

- No início, programas conheciam apenas a memória física, à qual tinham acesso total.
  - Programador responsável por particionar programa em porções (overlays) do tamanho da RAM
- Vários processos na memória
  - Necessidade de proteção entre processos
  - Determinação das posições de cada programa
- Memória virtual(Manchester, 1961): separação entre
  - Espaço de endereçamento (de cada programa)
  - Memória física (controlada pelo S.O.)

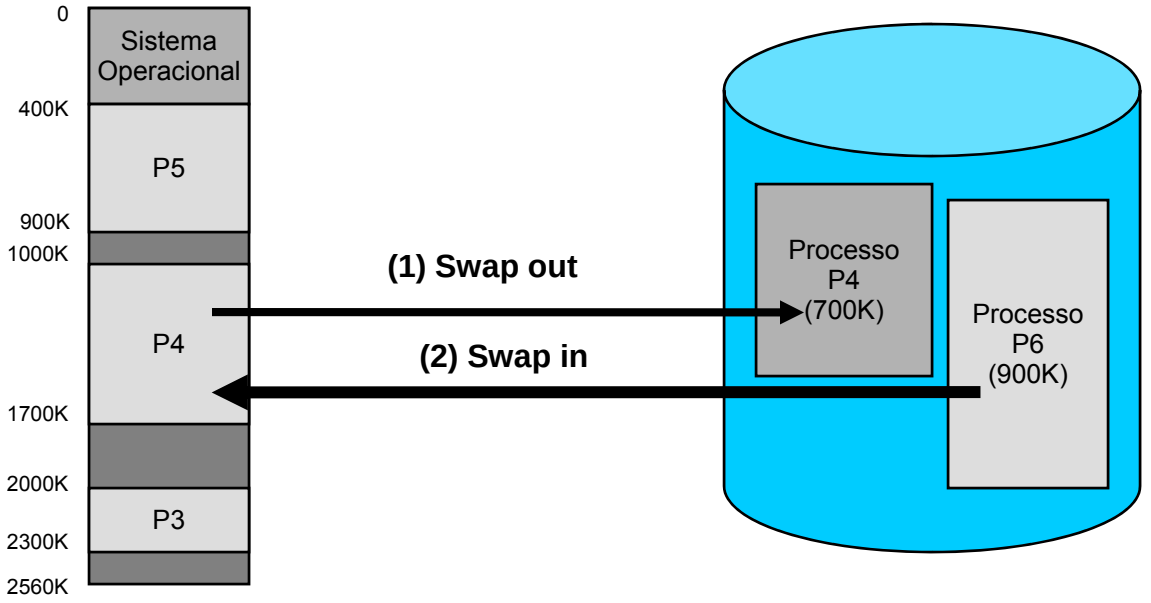
Alocação contígua

- Todo o programa em endereços contíguos
- Relocação por tabela ou registrador base



Swapping

- Processos suspensos fora da memória
- Exigem relocação ao serem recarregados



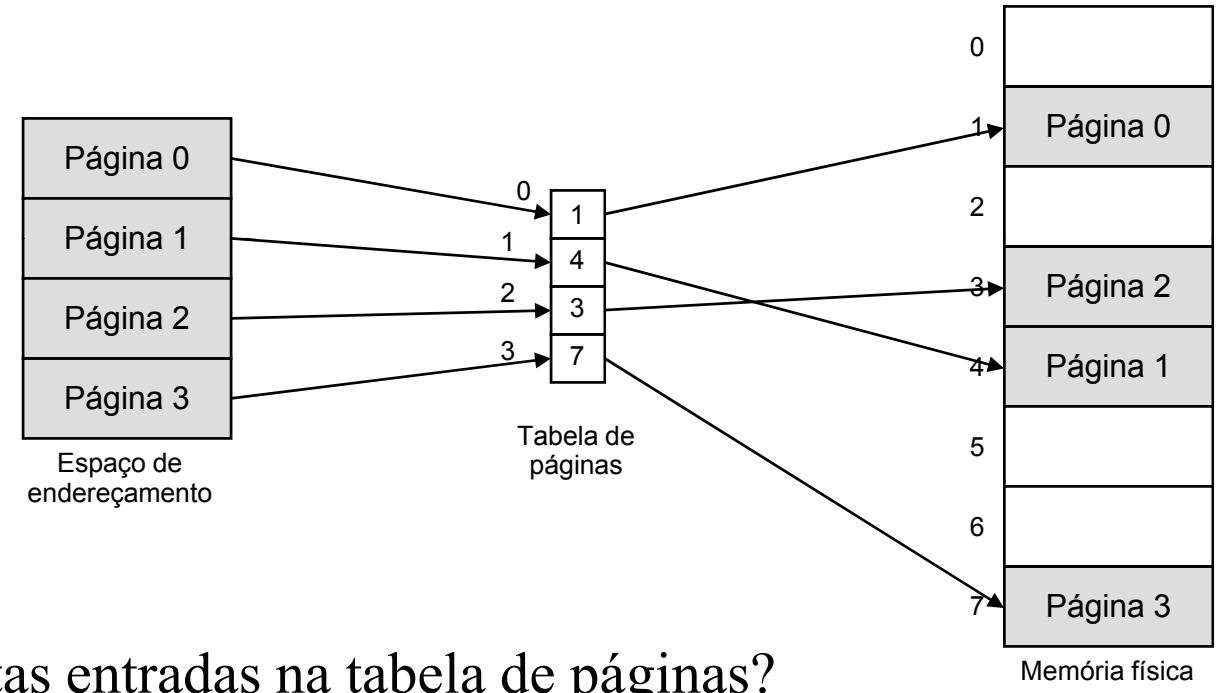
Memória virtual

- Apenas parte do programa na memória
- Requer espaço em disco para o restante
- Memória movida em páginas/segmentos
- *Hardware* faz a tradução

Paginação

- Espaço de endereçamento em páginas
  - 512 – 4MB (4KB é comum)
- Transparente ao usuário – até certo ponto
- Tabela de tradução para memória física
  - Uma tabela por processo mantida em memória principal
  - TLB – translation lookaside buffer
    - Registradores para acesso + rápido
- Página virtual x Moldura/quadro de página

Paginação



Quantas entradas na tabela de páginas?

Molduras de páginas

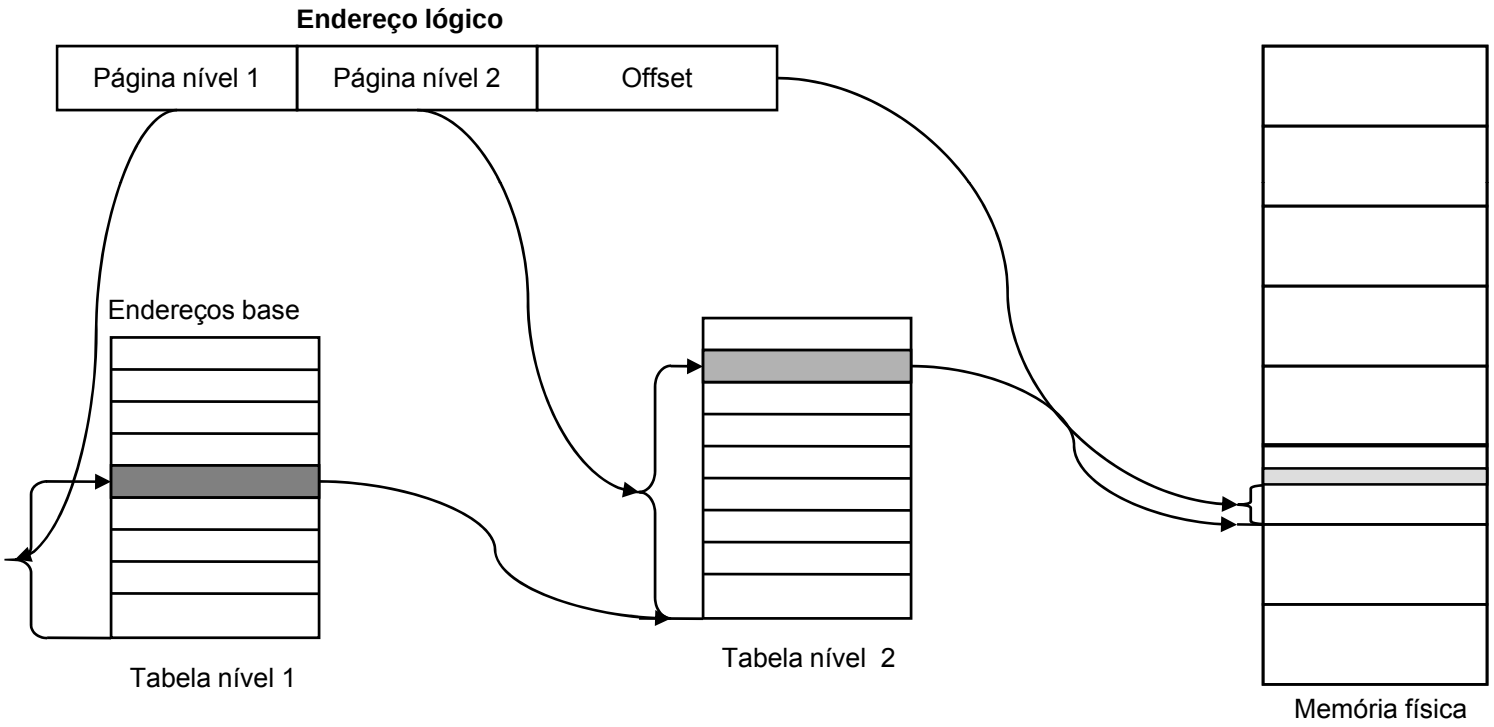
Localidade de referência

- Observação prática:
  - Programas tendem a utilizar um conjunto reduzido de páginas em cada instante
  - Páginas desse conjunto mudam lentamente
- Working set (Denning, 1968):
  - Conjunto de páginas que um processo precisa a cada instante.

Unidade de gerência de memória (MMU)

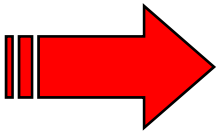
- Tradução de endereços virtuais x reais
- Verificação de permissões de acesso
- Bits de controle:
  - Página presente/ausente
  - Página acessada
  - Página alterada

Tradução de endereços



Paginação sob demanda

- Nem todas as páginas são carregadas
- Acesso a páginas ausentes causa *TRAP*
  - Page fault – falta de página
- Trap requisita a leitura da página do disco
- Tabela de páginas atualizada
- Instrução que causou a falha é reinicializada



Page Faults custam caro!

"Melhor comprar mais memória que trocar o processador..."

Custo da paginação

- Acesso na memória: 126 ns em média
- Page fault: 420 ns para gerar falha
  - 8 ms de latência de disco
  - 15 ms de *seek*
  - 1 ms de transferência de dados p/ memória
  - Total: ~ 25 ms
- Probabilidade de falha p:
  - T = (1-p) x 126 + p x 25.000.000 !!!
  - T = 250 ns => p < 0,0005 % !!!

Substituição de páginas

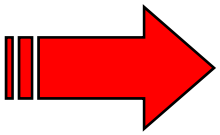
- E se os quadros estão todos ocupados?
  - Quadros que não foram acessados “recentemente”
  - Quadros menos utilizados
  - Páginas “limpas” (não precisam ser copiadas)

Políticas de substituição

- FIFO (*First In, First Out*)
  - Página carregada há mais tempo
  - Fácil, mas não considera taxa de utilização
- LFU (*Least Frequently Used*)
  - Página menos utilizada
  - Páginas novas perdem para páginas muito usadas no passado
- LRU (*Least Recently Used*)
  - Página cujo último acesso é mais antigo
  - “Se não tem sido utilizada, não o deve ser”

Se a memória não é suficiente...

- Falhas geram requisições para os discos
- Processos parados esperando
- Carga da CPU diminui
- Novos processos são iniciados
- Aumenta a disputa por quadros da memória
- Mais falhas acontecem...



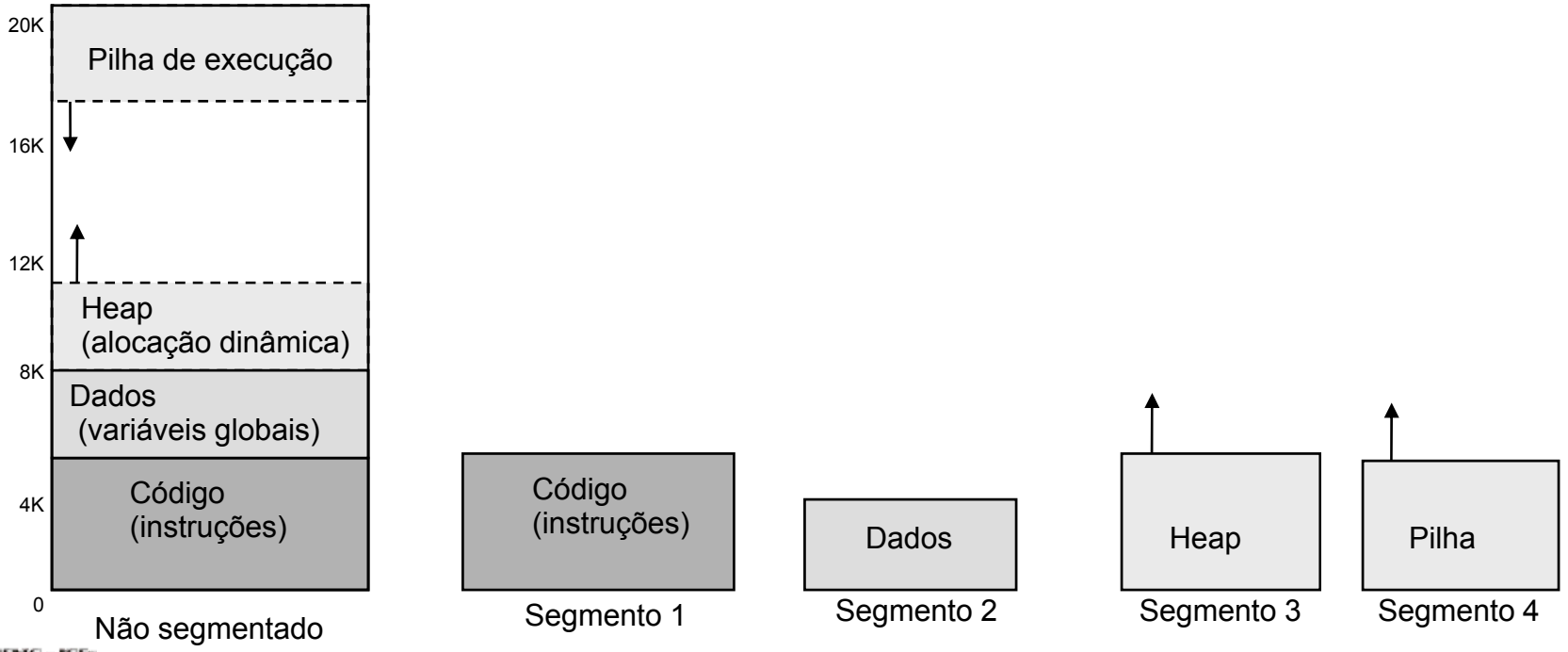
Thrashing!

Escolha do Tamanho de Página

- Vários fatores:
  - Tamanho da tabela de páginas (hardware)
  - Uso eficiente/ineficiente de largura de banda do disco
  - Fragmentação

Segmentação

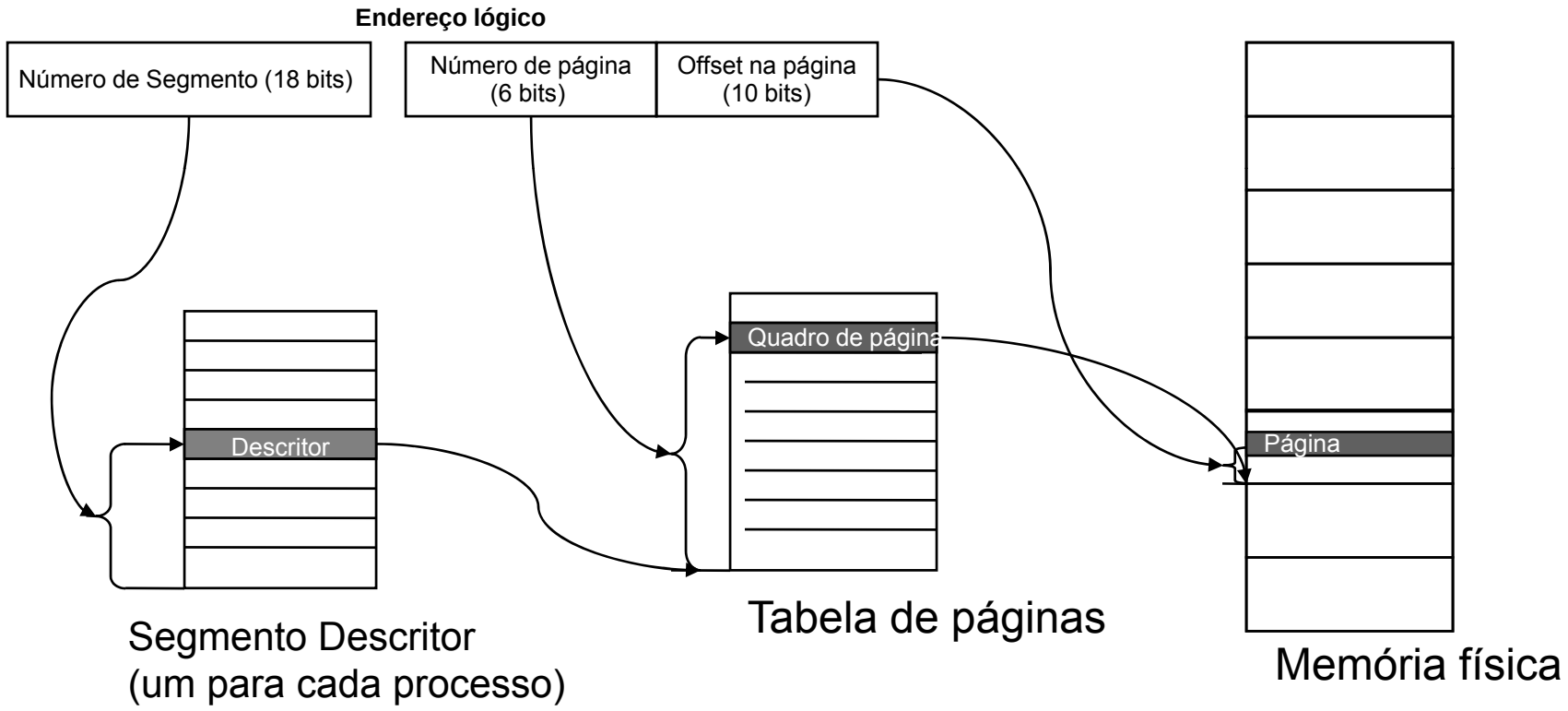
- Espaço de endereços multidimensional
  - Segmentos podem ter tamanhos diferentes (dinâmico)
- Mesmos problemas de alocação contígua



Segmentação

- Implementação via permutação
  - Segmentos inteiros transferidos entre memória e disco
- Implementação via paginação
  - Cada segmento é paginado
  - Páginas acessadas via demanda.
  - Cada segmento tem sua tabela de páginas

Segmentação Paginada: MULTICS



Fragmentação

- Diferença entre área alocada e área utilizada
- Segmentação/alocação contígua:
  - Fragmentação externa
  - Espaço entre áreas utilizadas diminui: utilidade diminui
  - Algoritmos de gerenciamento de lacunas
    - Best fit
    - First fit : em geral melhor que best fit
  - Remoção da fragmentação via
    - Compactação
    - Aglutinação de lacunas vizinhas quando segmentos são removidos

Fragmentação

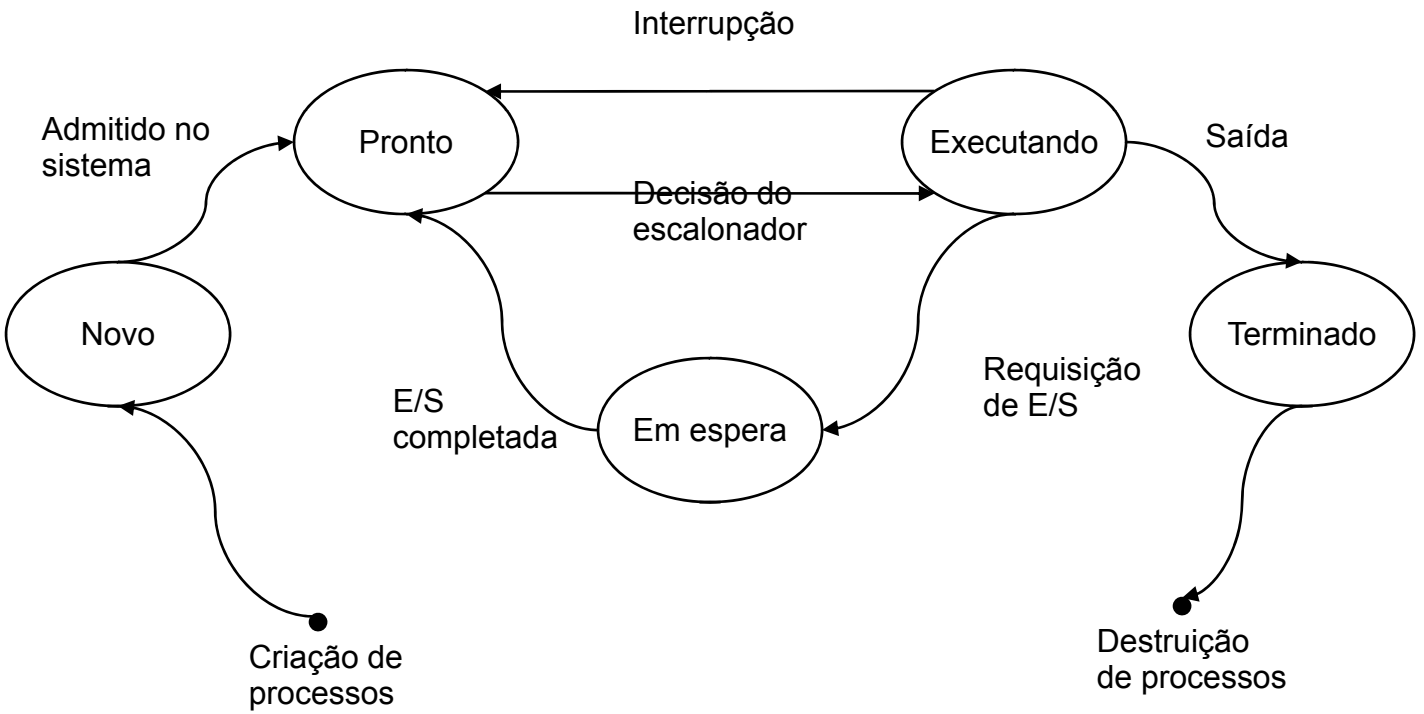
- Paginação:
  - Fragmentação interna
  - Não há espaço perdido entre páginas
  - Última página de cada processo pode ser incompleta
  - Fator adicional ao compromisso na escolha do tamanho da página

Memória Virtual no Pentium 4 e UltraSparc 3

Processos

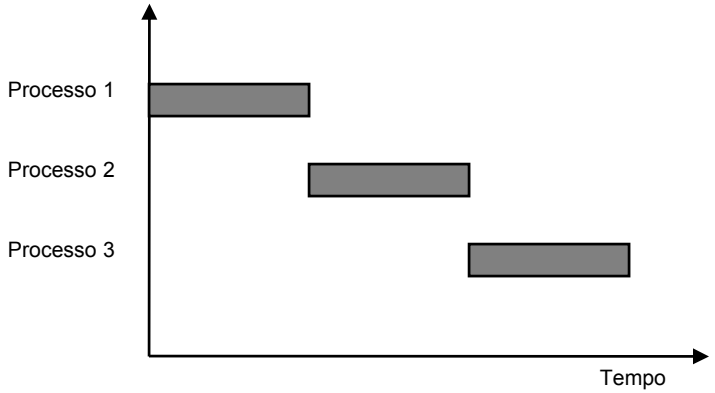
- "Programas em execução"
- *Process Control Block* (PCB)
  - Identificação do programa (código)
  - Variáveis estáticas e dinâmicas
  - Pilha de chamadas de procedimentos
  - Registradores e estado da CPU
  - Outras informações de escalonamento, memória

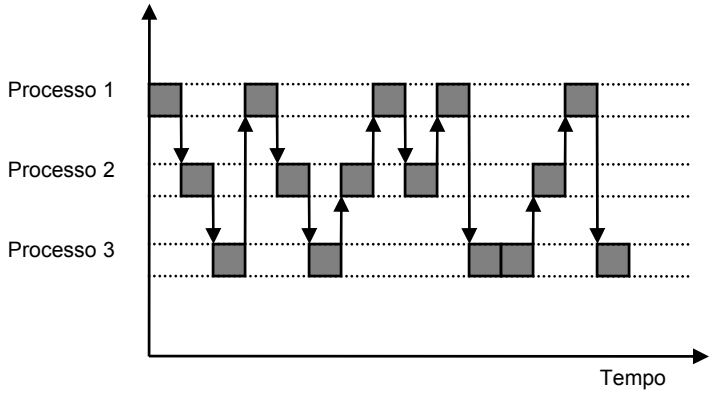
Estados de um processo



Escalonamento de processos

- Aumentar a utilização do processador
- Paralelismo aparente e transparente
- O S.O. decide quem, quando e como!





Trocas de contexto

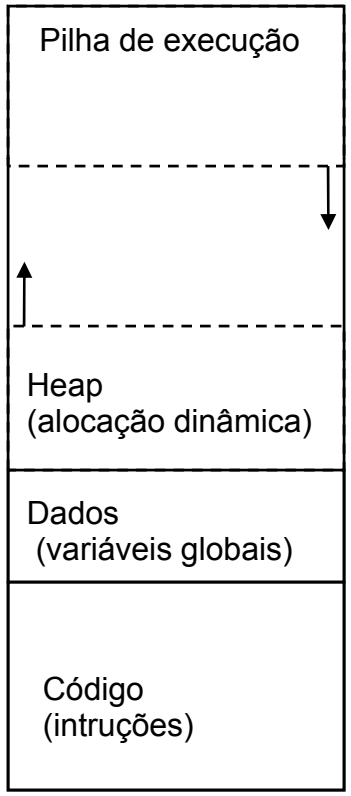
- Processos suspensos transparentemente
- Contexto deve ser recomposto ao reiniciar
- Mesmo princípio de co-rotinas
- Interrupções já fazem parte do trabalho...
  - Durante a interrupção o estado do processo é armazenado na pilha
  - Resta salvar o estado do processador e trocar a pilha que está sendo utilizada

Princípio de escalonamento

- Interrupção periódica salva contexto no PCB
- Paradas “voluntárias” utilizam *trap*
- PCBs são armazenados em filas
- Núcleo decide qual processo executar
  - Várias políticas possíveis
  - Decisão baseada nas filas de escalonamento
- PCB do processo escolhido define contexto

Etapas de um processo

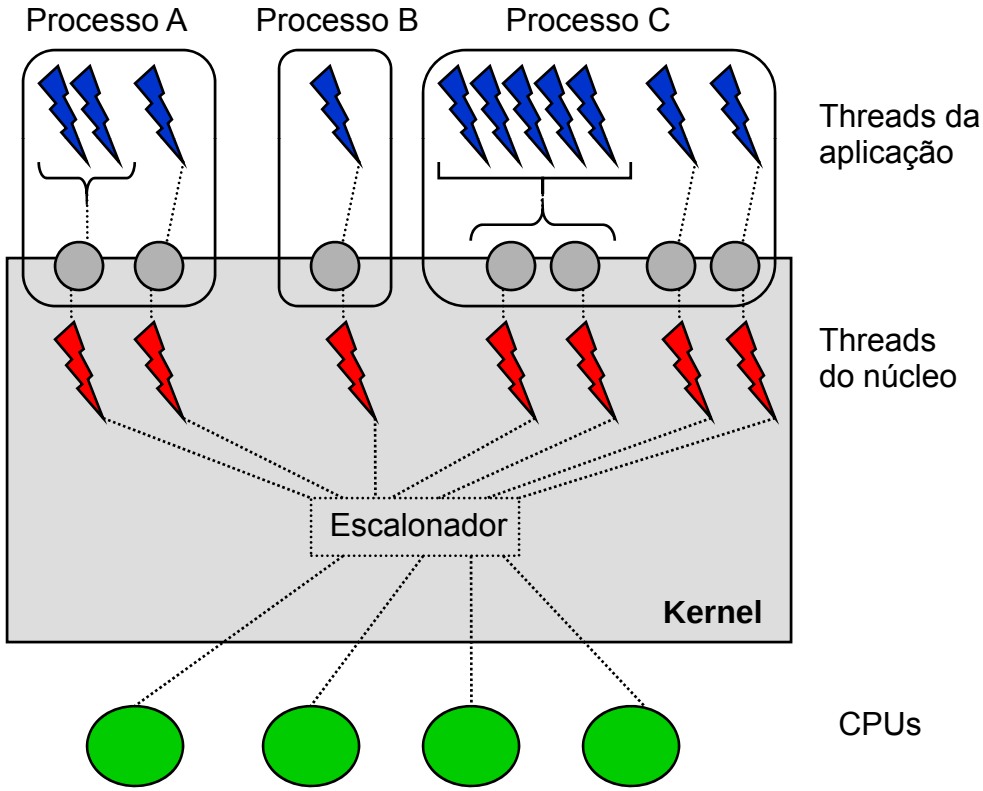
- Criação
  - Inicialização do PCB
  - Definição da imagem na memória:
- Execução
  - Escalonamento pelo núcleo
  - Comunicação entre processos
  - Acesso a dispositivos
- Terminação
  - Liberação de recursos



Diferentes entidades

- Processo:
  - Espaço de endereçamento + código
  - Sempre escalonado no núcleo
- Thread:
  - Apenas código (várias em um processo)
  - Diferentes implementações
    - No núcleo: escalonamento global
    - Na aplicação: controle do usuário

Processos e diferentes threads



Interação entre processos/threads

- É preciso controlar o acesso a recursos compartilhados (memória, arquivos, etc.)
  - P.ex.: programas acessando uma conta bancária

P1: (credita R$100)		P2: (debita R$100)	
P1.1	lê saldo atual	P2.1	lê saldo atual
P1.2	soma 100	P2.2	decrementa 100
P1.3	escreve novo saldo	P2.3	escreve novo saldo

Seqüência possível: P1.1 P1.2 P2.1 p1.3 P2.2 P2.3!!!

Interação entre processos/threads

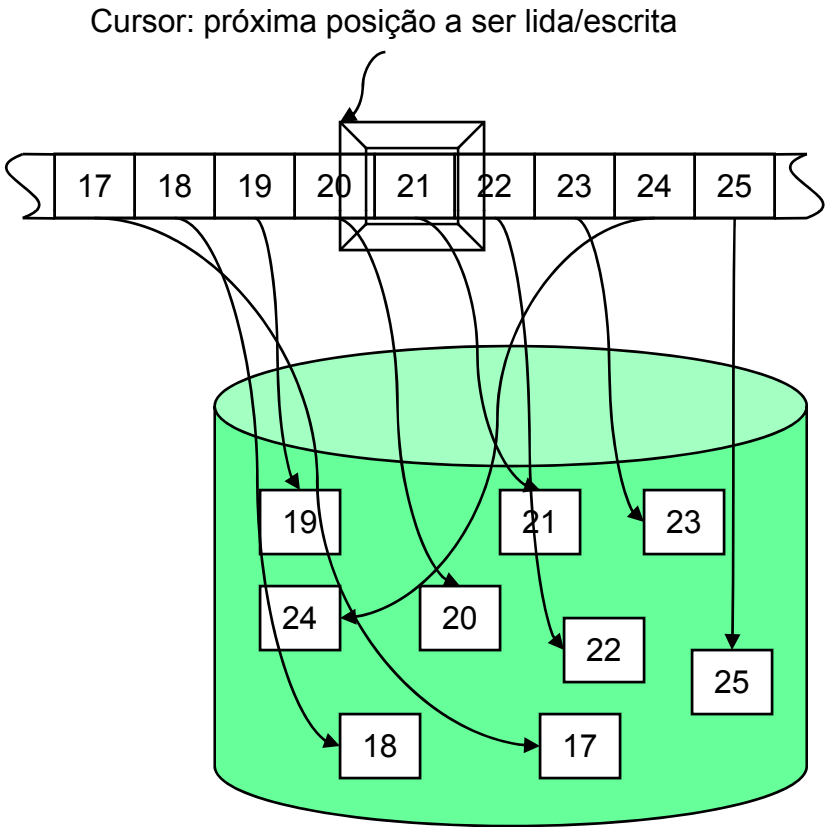
- Diversas abstrações existem para esse fim
  - Regiões de exclusão mútua
  - Variáveis de condição
  - Semáforos

Entrada e saída

- S.O. oferece abstração e controle de acesso
- Dispositivos abstratos:
  - Arquivos
  - Terminais (teclado+mouse, monitor)
  - Rede
- Complexidade modelada por *device drivers*

Arquivos

- Sequência de bytes
- Posição atual
- Atributos
  - Tamanho
  - Criador
  - Permissões
  - Data de criação
  - Data de acesso
  - Etc.



Operações sobre arquivos

- Abrir
- Ler
- Escrever
- Reposicionar o cursor (apontador)
- Truncar

Controle de acesso/proteção

- Simples: poucos recursos/garantias
- Hierárquico: controle definido por grupos
- Senha associada a cada arquivo
- Listas de controle de acesso:
  - Método mais genérico
  - Permite definição de acessos em detalhes
  - Cansativo para se administrar

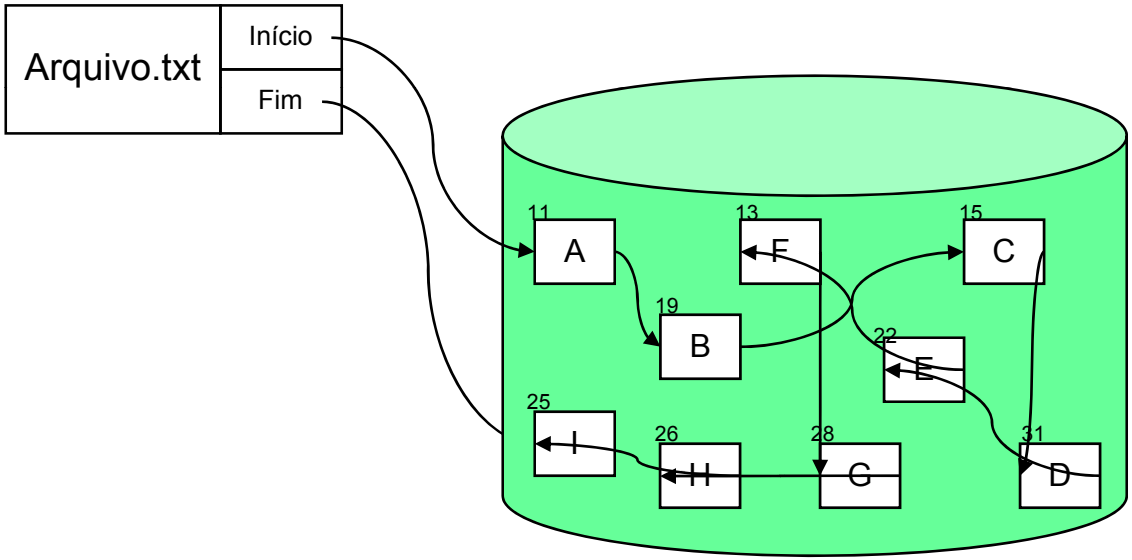
Mapeamento de arquivos em disco

- Define a relação entre blocos da sequência e setores do disco real
- Alocação sequencial:
  - Mais simples de se implementar e controlar
  - Possível apenas com arquivos fixos (CD-ROM)
- Alocação não sequencial:
  - Solução mais genérica
  - Permite lidar com arquivos de tamanho variável
  - Requer estruturas auxiliares

Técnicas de mapeamento não-sequencial

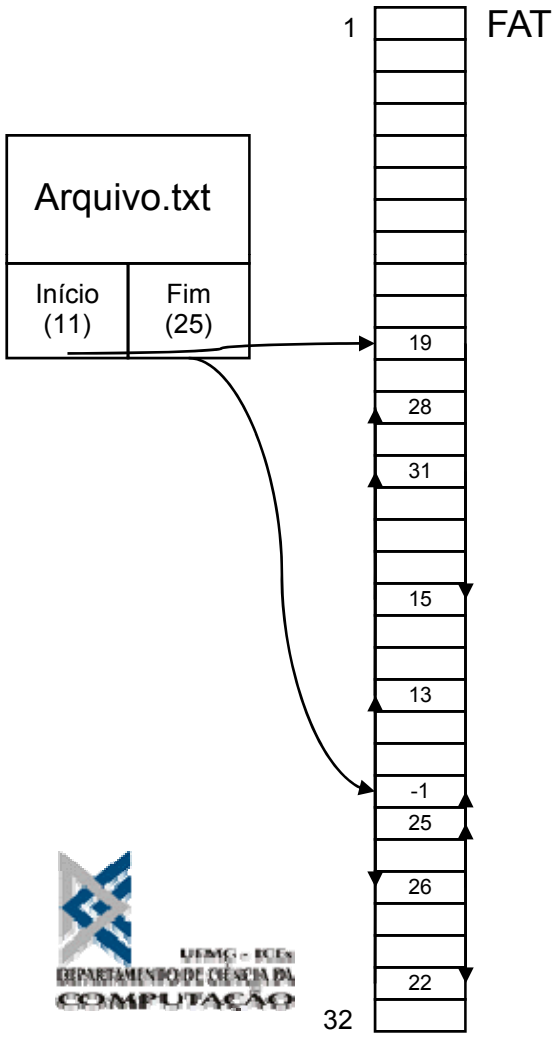
- Lista encadeada:
  - Cada bloco aponta para o seguinte: simples
  - Requer acesso sequencial para localizar blocos
- Encadeada com FAT (tabela de alocação):
  - Lista encadeada armazenada em tabela isolada
  - Reduz custo de acesso sequencial
- Arquivo indexado:
  - Estrutura de dados c/ posição de todos os blocos
  - Acesso direto a qualquer parte do arquivo

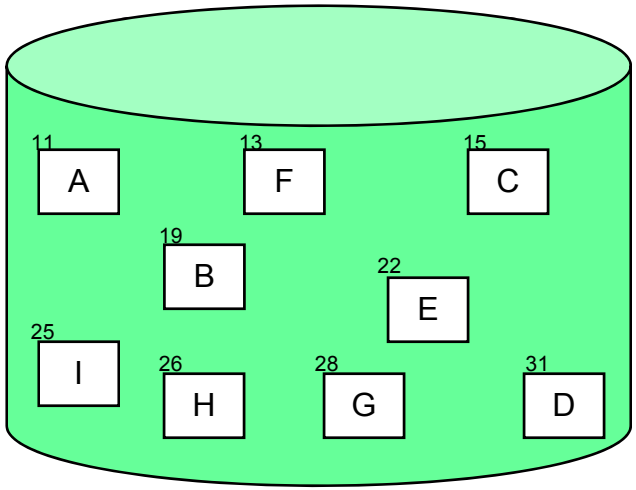
Mapeamento com lista encadeada



- Acesso sequencial:
  - simples
- Acesso direto:
  - trabalhoso

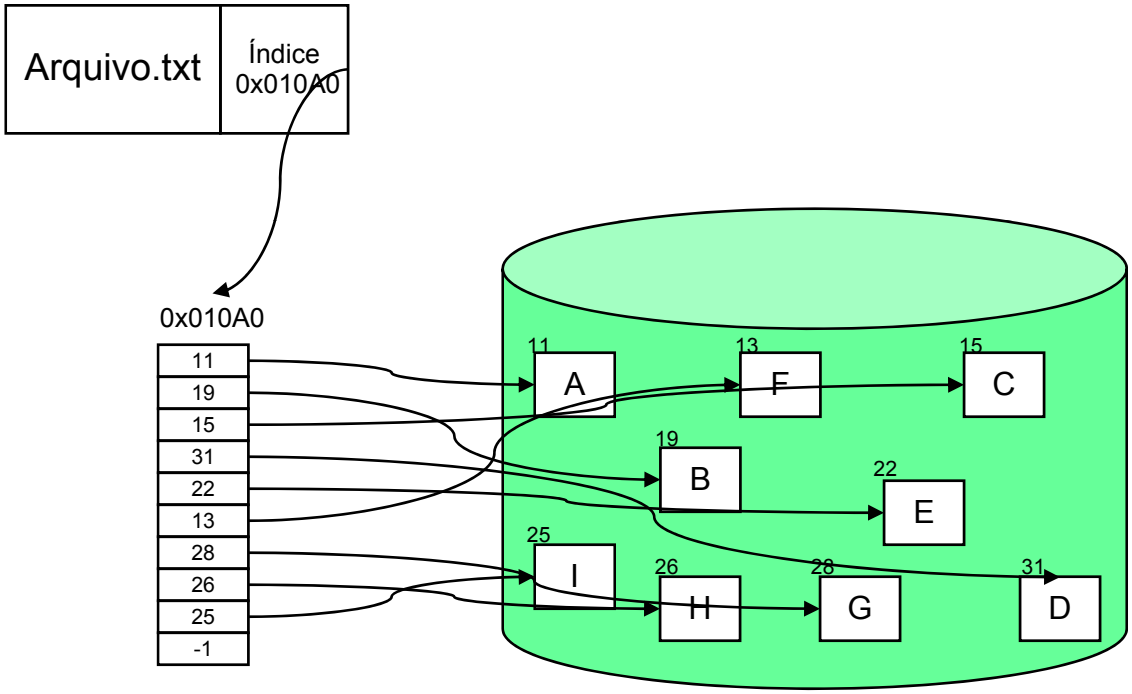
Mapeamento com tabela de alocação (FAT)





- Acesso sequencial:
  - simples
- Acesso direto:
  - mediano
- FAT na memória
- Exemplo: DOS

Mapeamento com índices



- Acesso sequencial:
  - simples
- Acesso direto:
  - simples
- Índice na memória
- Um índice por arquivo
- Índices:
  - Lista encadeada
  - Multi-nível
  - híbridos
- Exemplo: UNIX

Gerência do espaço livre

- Lista encadeada de blocos livres
  - Implementação simples
  - Solução direta no caso da FAT
- Mapa de bits dos setores do disco
  - Acesso fácil com auxílio do *hardware*
  - Requer espaço em disco e memória
- Listas com otimizações/compactação

Diretórios

- Associação de nomes com arquivos
- Atributos de arquivos
- Atributos do diretório
  - Permissões de acesso
  - Tamanho
  - Hierarquia de diretórios

Operações em diretórios

- Criar arquivos
- Procurar por um arquivo
- Remover arquivos
- Renomear um arquivo
- Percorrer o sistema de arquivos
- Criar/remover/renomear diretórios

Estrutura de diretórios

- Lista de nomes de arquivos
- Lista encadeada simples
- Lista ordenada
- Hash
- Árvores de busca (B+)

Sistemas de arquivos em redes

- Arquivos em discos de outras máquinas
- S.O. transforma acessos a disco em mensagens pela rede
- A separação entre máquinas gera problemas semânticos
  - Arquivos podem ser alterados por dois clientes
  - Alterações não são visíveis imediatamente
  - Controle de acesso deve ser distribuído

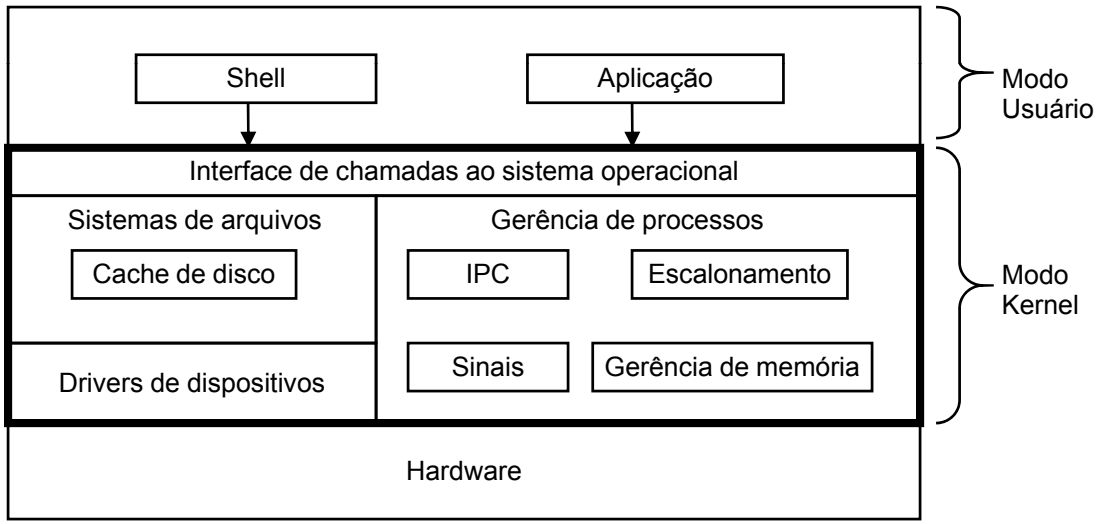
Unix: um pouco de história

- Bell Labs, Ken Thompson, Dennis Ritchie
- 1971: PDP-7, assembly
- 1972: PDP-11, C --- código fonte disponível
  - Artigo histórico sobre Unix em 1974
  - Ritchie & Thompson: ACM Turing Award 1984
- 197?: UC Berkeley:
  - Memória Virtual, TCP/IP (socket)
  - Alavanca para uso do TCP/IP na Internet
- 1980: AT&T - System V
- POSIX: Padrão de portabilidade

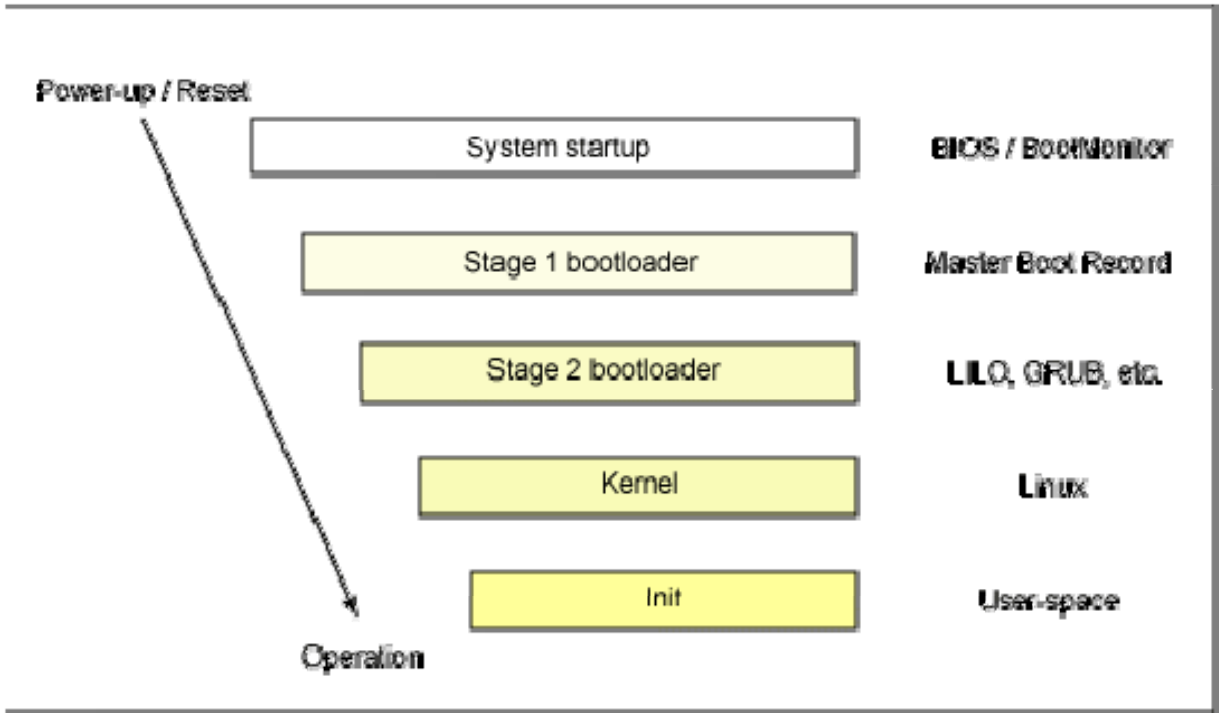
Unix: versões atuais

- Sun Solaris: um dos mais estáveis
- SCO Unix: um dos mais antigos
- Linux: um dos mais populares
- freeBSD: um dos mais eficientes
- netBSD: o com mais plataformas
- Apple Mac X: um dos mais diferentes

Unix: estrutura geral do sistema



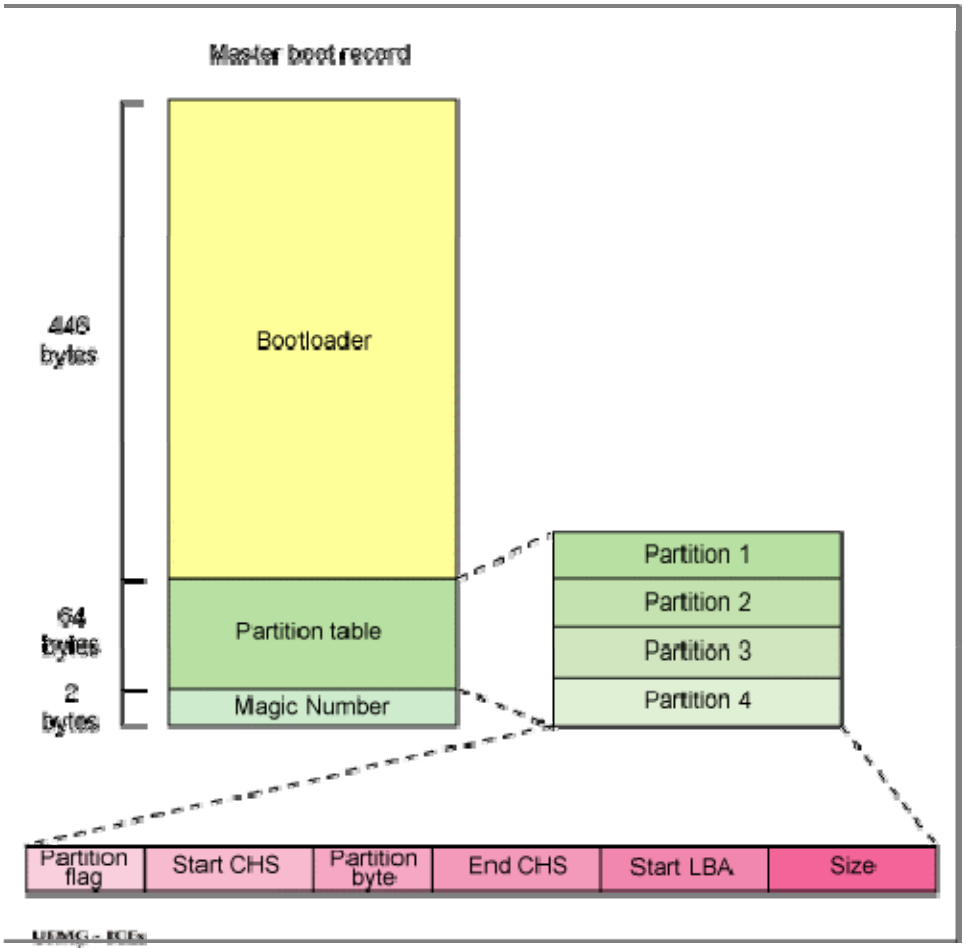
Unix: o processo de boot



Fonte: IBM

http://www.ibm.com/developerworks/linux/library/l-linuxboot/index.html

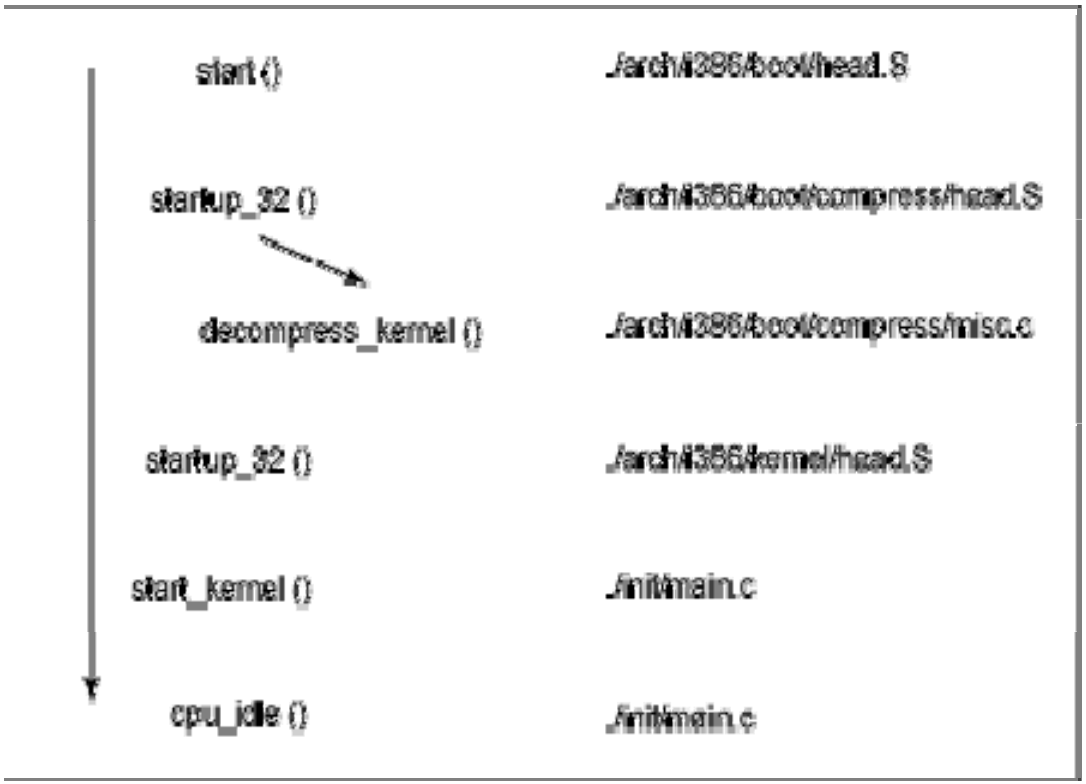
Unix: o processo de boot (bootloader)



Procura pela partição que
é bootable e carrega
o boot record dela (1º setor)

Fonte: IBM

Unix: o processo de boot no kernel



Fonte: IBM

Hardware setup

Estabelece ambiente (pilha,etc)
Descomprime kernel

Inicializa tabela de páginas e
Paginação é habilitada
Identifica CPU

Inicializações de software
(interrupção, configuração de memória)
Chama kernel_thread pra inicializar init
(1º programa em modo usuário)

idle task: escalonador toma controle

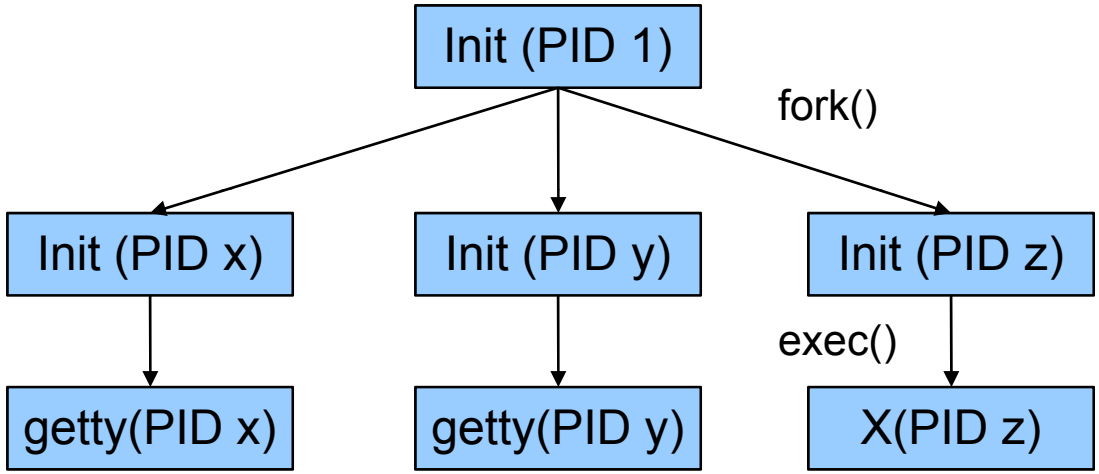
Unix: o processo de boot (init)

- Primeiro processo a executar
- Nunca termina (causa shutdown se morrer)
- Lê arquivos de configuração
- Dispara demais processos

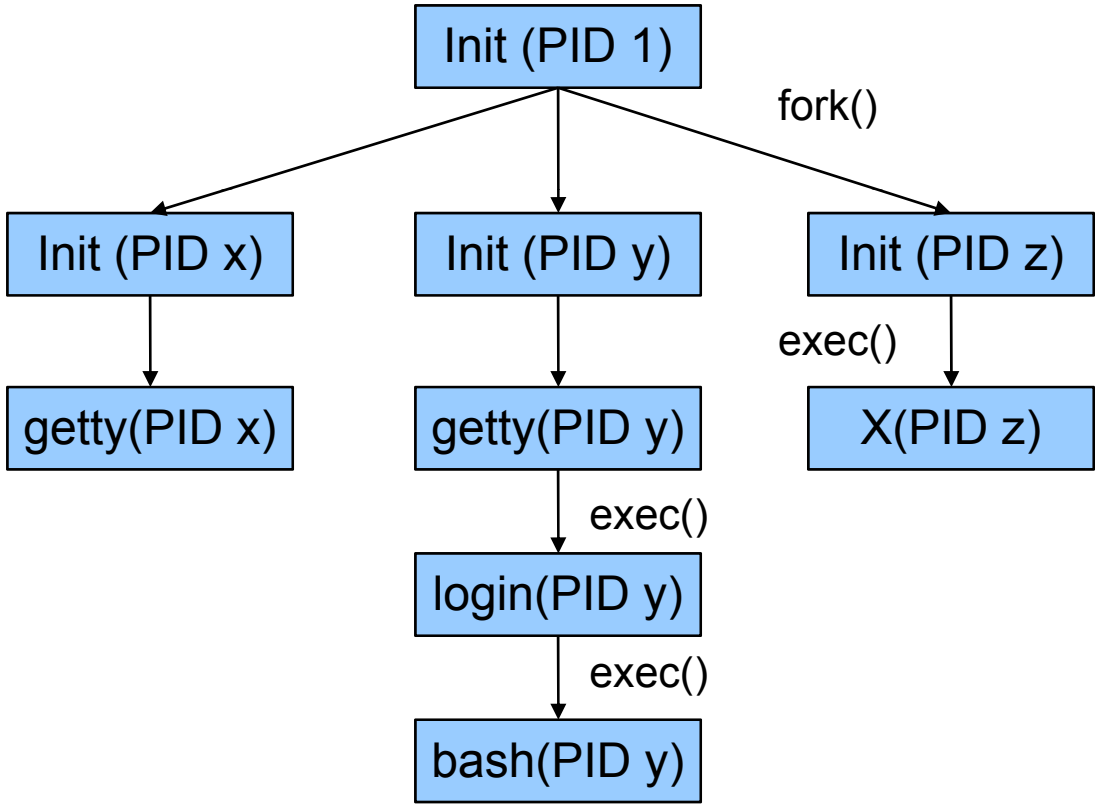
Unix: processos

- Hierarquia de processos: pais e filhos
  - **`fork()`:** cópia do processo pai
  - **`exec()`:** substitui processo por novo programa
- Várias formas de comunicação entre processos: memória, semáforos, pipes, msgs.
- Processos, *threads* de aplicação (pthreads)
- Alguns têm suporte a *threads* de núcleo (Solaris, Linux)

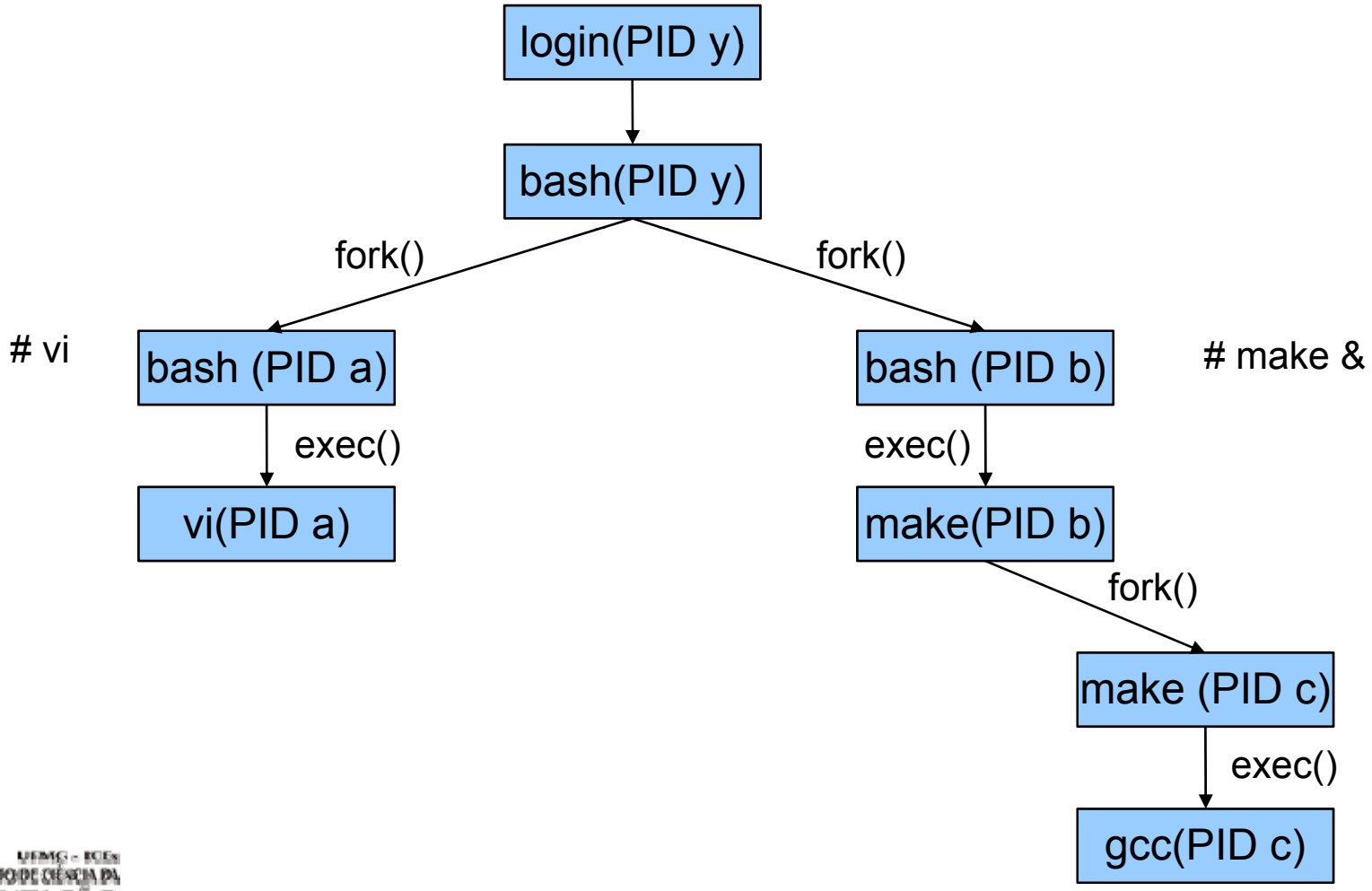
Unix: processos (a partir do init)



Unix: processos (a partir do init)



Unix: processos (a partir do login)



Unix: escalonamento de processos

- Esquema de escalonamento bastante complexo
  - Tempo de resposta rápido para processos interativos
  - Bom throughput para processos batch
  - Múltiplas prioridades + evitar inanição
- Políticas diferentes para processos em tempo real ou interativo / batch

Unix: escalonamento de processos

- Processos interativos ou batch:
  - Filas multinível de prioridades dinâmicas com time-sharing
  - Time-sharing: processos podem ter quanta diferentes
  - Prioridades dinâmicas:
    - Escalonador aumenta prioridade de processos que não executam há muito tempo
    - Visa equalizar uso do processador

Unix: escalonamento de processos

- Usuários podem reduzir prio. de processos
- Apenas root pode aumentá-las
- Classificação de processos:
  - *I/O bound* - maior parte do tempo em filas
  - *CPU bound* - usam a CPU intensivamente
  - Escalonador privilegia I/O bound: melhor tempo de resposta para aplicações interativas

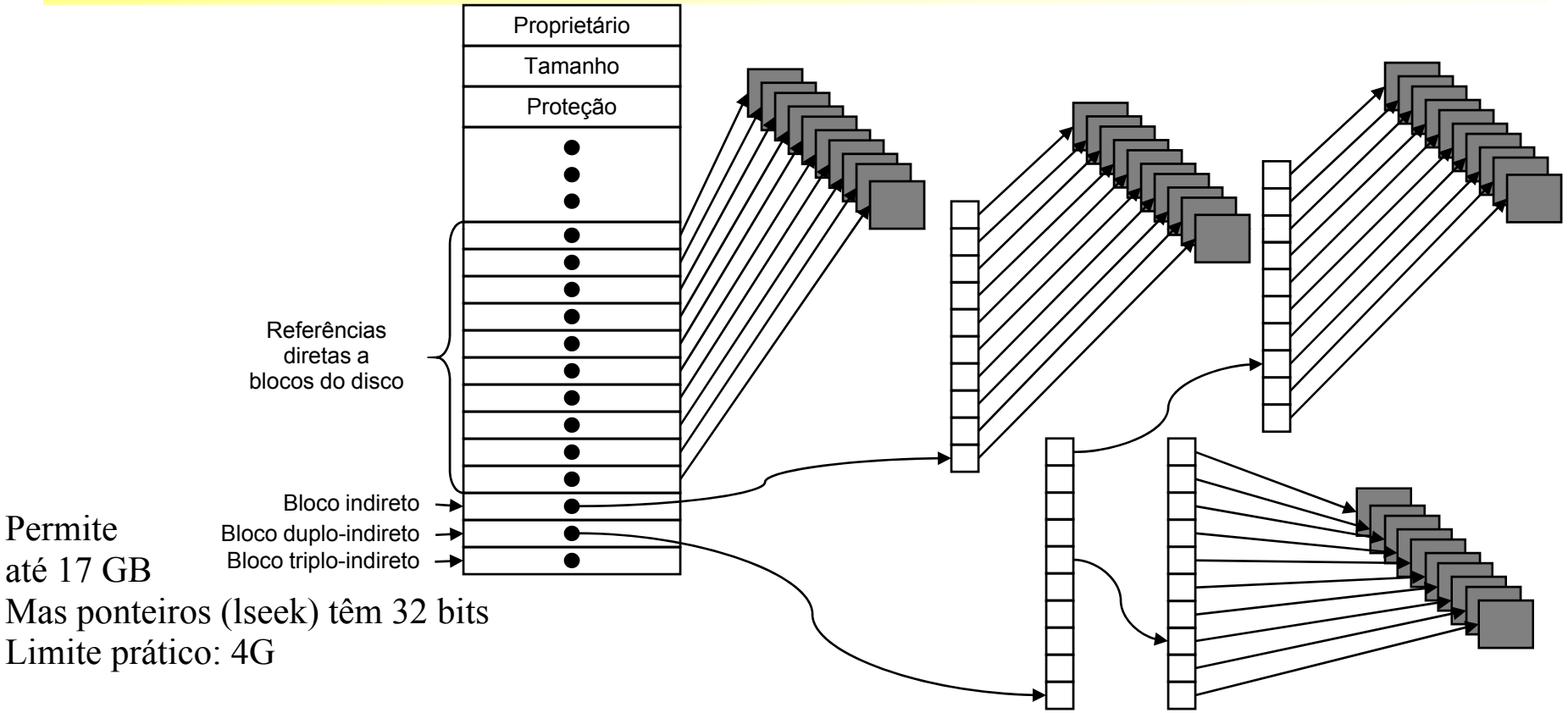
Unix: IPC

- Pipes
- Filas de mensagems (System V e Solaris)
  - msgget, msgsnd, msgrcv
- Memória Compartilhada
  - shmat, shmget
- Mapeamento de arquivos no espaço de endereçamento
  - mmap
- Semáforos
- Pthreads
  - padrão não define se de kernel ou modo usuário

Unix: E/S

- Arquivos: sequências de bytes $(2^{32} - 1)$
- Arquivos especiais para acesso a dispositivos de E/S
- Sistemas de arquivos na árvore de diretórios
- Arquivos mapeados através de *i-nodes (64 bytes)*
  - tabela de inodes localizada no início do disco, em sequência,
- Permite referências múltiplas (*links*)
- Diretórios como listas encadeadas simples
  - Cada entrada: nome de arquivo + número de inode
- Dispositivos vistos como arquivos
  - Afinal, “tudo é um arquivo”
- Dispositivos de bloco, caractere e... ???

Unix: i-node



Permite até 17 GB
Mas ponteiros (lseek) têm 32 bits
Limite prático: 4G

Contém : tipo do arquivo (ordinário, diretório, dispositivos estruturados em blocos ou não),
9 bits de proteção, número de links para o arquivo,
identidade do proprietário, grupo, tamanho (byte de mais alta ordem com valor),
última leitura/escrita/alteração no inode, 13 endereços de disco

Unix: desempenho de E/S

- Tabela de inodes mantidas em memória para acesso rápido
- Lista de blocos livres (free list)
  - ° Alocação espalhada pelo disco
- Cache de blocos (memory cache)
- Sistema lê o *próximo bloco* antes de ele ser referenciado
  - ° Prefetching

Unix: proteção

- Permissões para o dono, grupo e “outros”
- Arquivos: ler, escrever, executar
- Diretórios: listar, criar/remover arquivos, percorrer
- Processos executam com permissões do usuário exceto em casos especiais

Unix: gerência de memória

- Memória virtual paginada sob demanda
- Arquivos mapeados em memória (**`mmap()`**)
  - Arquivo visto como um vetor na memória
- Processos podem compartilhar páginas
- Cópias podem ser feitas com *copy-on-write*:
  - Nada é copiado imediatamente
  - Acessos de leitura compartilham páginas
  - Acessos de escrita forçam a criação de cópia

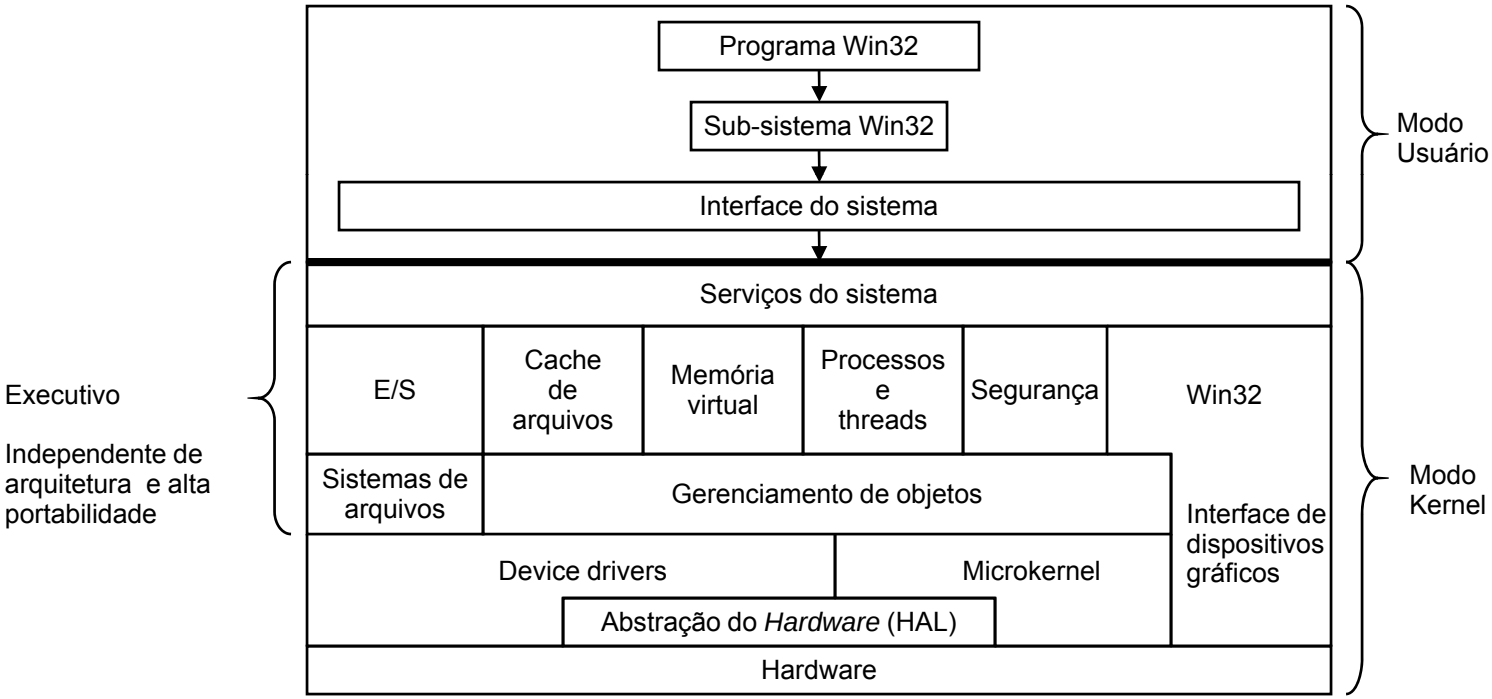
Win XP: um pouco de história

- 1981: MS DOS 1.0,
- 1983: MS DOS 2.0 (24KB)
- 1984: MS DOS 3.0 (36KB)
- 198?: MS Win (Mac), casca sobre o DOS
- 1995: Win95 - ainda casca, mais poderoso
- 1998: Win98 - mesma coisa que 95, pior!
  - Integração entre computador/Internet/TV: Anti-trust...
- 2000: Win2000 - adulto... 65000 bugs!
  - sistema operacional real, 32 bits, completamente novo
- 2001: WinXP/NT 5.1 (praticamente Win2000)

Win XP: estrutura

- Inicialmente um micro-kernel
  - Vários processos servidores excutando em modo usuário
  - Comunicação com servidores via mensagens requisição/resposta (cliente-servidor)
- NT 4.0: tudo no kernel (desempenho)
  - Só “micro-kernel” é residente e sem preempção: interrupções, escalonamento e sincronização, gerenciamento de tempo
- Sub-sistemas de ambiente: compatibilidade
  - Exporta conjunto de chamadas de função que programas usuários podem usar
- Fechado: chamadas do sistema secretas
- Win32 API é a interface documentada: procedimentos de bibliotecas que fazem o trabalho (modo usuário) ou fazem chamadas ao sistema
- Utiliza objetos (manipuladores), mas não orientado a objetos

Win XP: estrutura geral do sistema



Win XP: processos

- Todos os processos criados “iguais”
  - **`CreateProcess()`:** novo executável
  - Não tem hierarquia pai-filho
  - Mas hierarquia de processos pode ser montada:
    - CreateProcess retorna manipulador que permite controle sobre novo processo

Win XP: processos

- Processos, *threads* e *fibers (thread leve)*
  - *Threads criadas no kernel*
  - Escalonador escolhe processo E thread que vai executar
  - Threads são caras: troca de contexto incorre em passagem pelo kernel
  - Fibers: escalonada em nível de usuário pelo processo que a criou (paralelismo muito mais rápido)
- Comunicação entre processos como Unix

Win XP: E/S

- Arquivos: sequências de bytes ($2^{64} - 1$)
- Referências múltiplas para arquivos (*links*)
- Índices em uma tabela mestre
  - Master File Table: uma por volume
- Arquivos imediatos: dados no índice
- Índices como listas encadeadas para arquivos muito grandes
  - Contém ponteiros para agrupamentos com dados
- Diretórios como árvores B+
- Controle de acesso sofisticado (ACLs)

Win XP: proteção

- Usuários têm lista de credenciais
  - SID: security ID
  - Lista de grupos de segurança aos quais ele pertence
  - Privilégios especiais (se houver)
- **Objetos** têm lista de controle de acesso
  - Descritor de segurança com ACL
- Cada acesso compara as listas em ordem

Win XP: gerência de memória

- Recursos semelhantes ao Unix
  - **`mmap()`**
  - memória compartilhada entre processos
- Interface estende controle de páginas para a aplicação
- Aplicação pode “amarrar” páginas (virtuais) na memória física
  - Aplicações de tempo real
  - Precisa permissão de administrador

Mais detalhes na disciplina

SISTEMAS OPERACIONAIS