



Sistemas de Computadores

Arquitectura de um Sistema Operativo

Luis Lino Ferreira

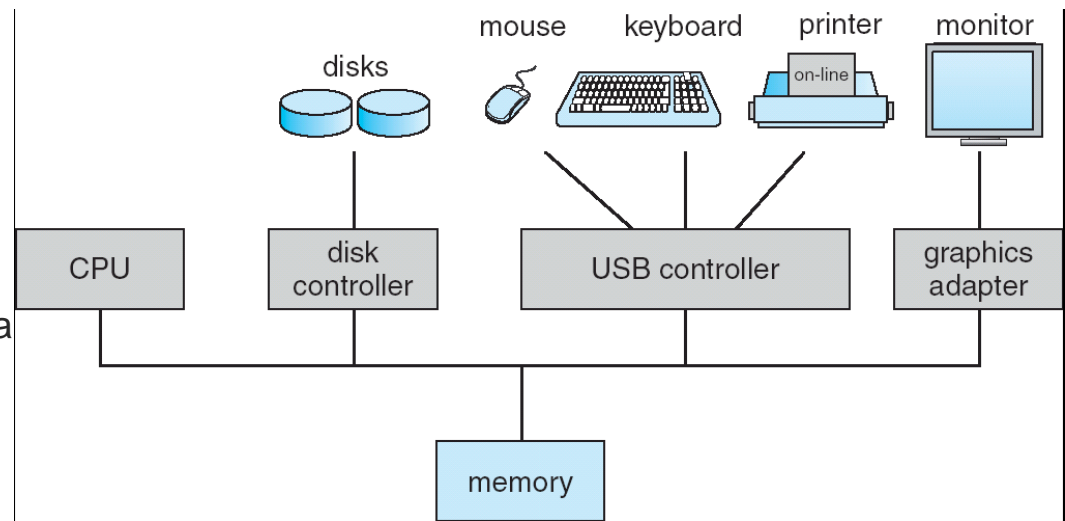
Fevereiro de 2016

[Sumário]

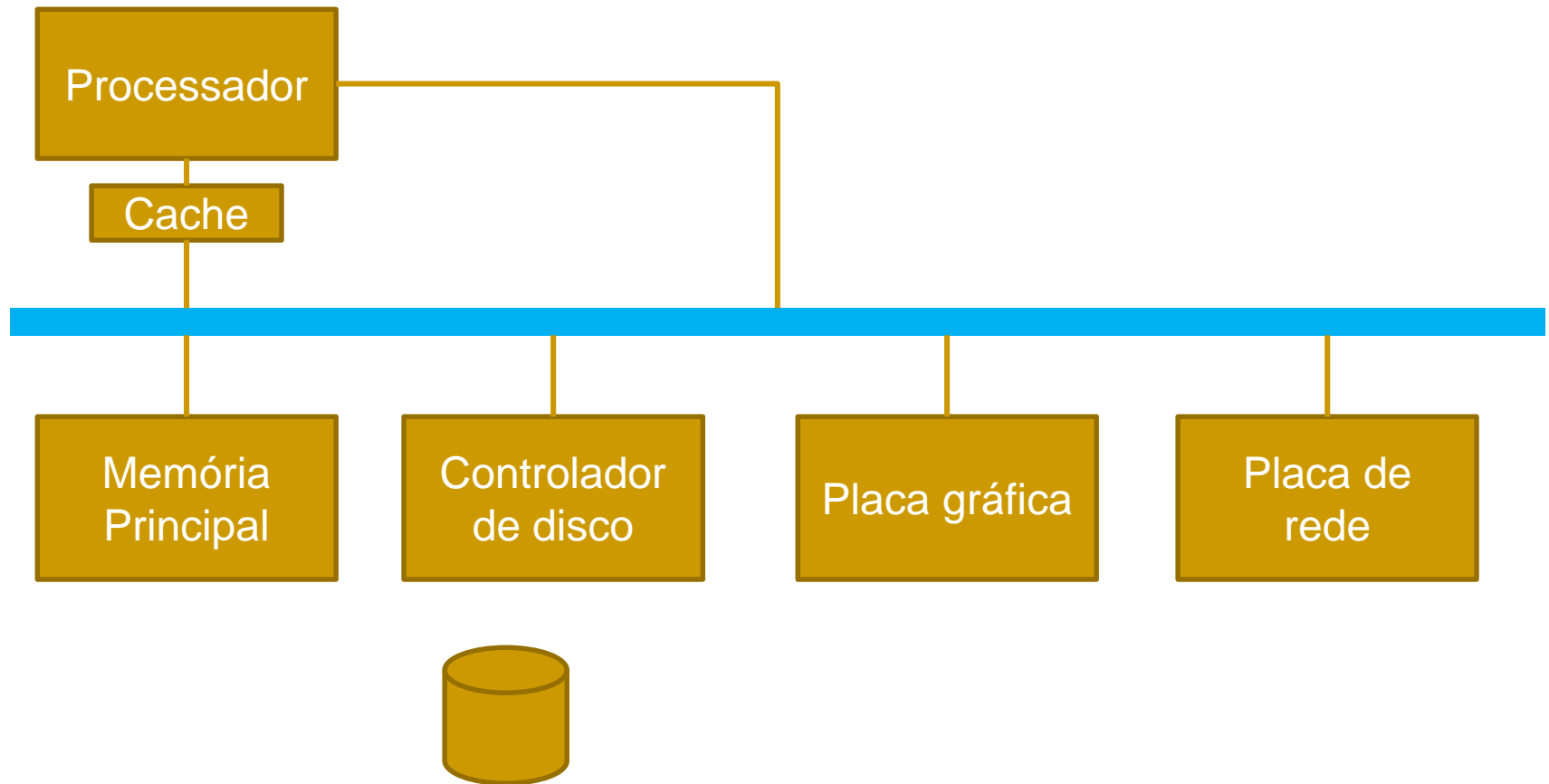
- Arquitectura de um Computador
 - Estrutura de I/O
 - Estrutura de Armazenamento
 - Protecção do Hardware
- Arquitectura de um Sistema Operativo

Arquitectura de um Computador

- Controladores de I/O e a CPU podem executar de uma forma Concorrente
- Cada controlador está encarregue de um dispositivo particular
- Cada controlador tem um buffer local
- A CPU movimenta dados da (para) memória principal para (a partir de) os buffers locais
- I/O é a partir do dispositivo para o buffer local do controlador
- O controlador informa a CPU que terminou a sua operação através de uma interrupção



[Arquitetura de um Computador]



Arquitectura de um Computador

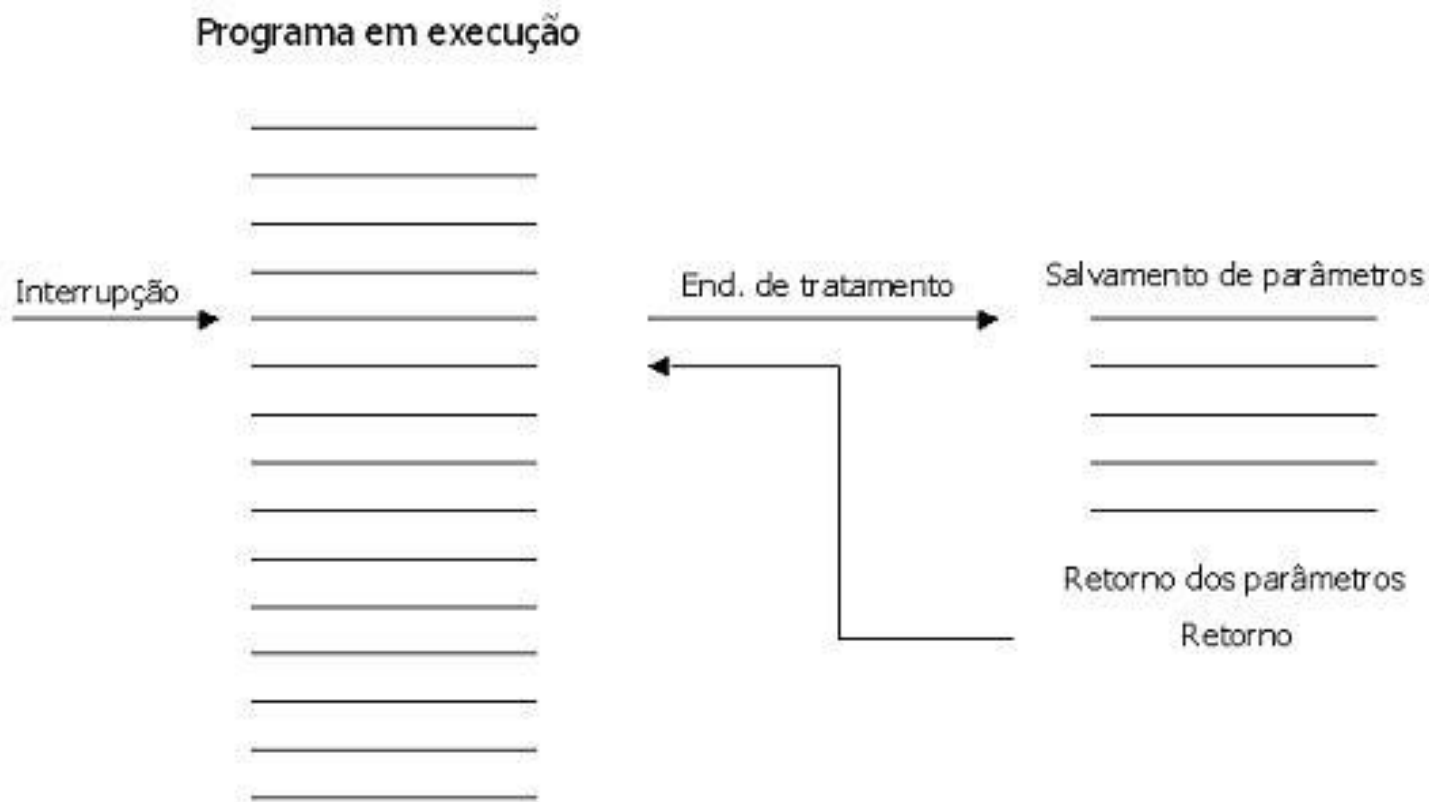
- Ligação entre componentes através de um barramento (*system bus*)
- Cada componente é controlado por um **controlador**
- A operação dos controladores é **concorrente** competindo pelo acesso à memória
- As operações de I/O são “buferizadas” pelos **controladores**
- Os controladores informam a CPU de eventos através de **interrupções**

Arquitectura de um Computador

■ Sistema de interrupções

- É um evento externo que leva a que o processador pare a execução do programa corrente e desvie a execução para um bloco de código chamado de rotina de interrupção (normalmente são decorrentes de operações de I/O)
- Portanto
 - Assinalam o acontecimento de um evento (por ex., o movimento do rato)
 - Cada evento é servido, imediatamente, por uma rotina específica, ou seja uma interrupção transfere o controlo para uma rotina de serviço
 - A arquitectura de interrupções tem de guardar o endereço da instrução interrompida
 - As interrupções entretanto chegadas são *disable* enquanto outra interrupção está a ser processada (para impedir a sua perda)

Arquitectura de um Computador



Arquitectura de um Computador

- Sistema de interrupções
 - O SO tem os meios para guardar informações sobre o estado da CPU antes da interrupção e para repor o estado anterior após o tratamento da interrupção
 - Registos
 - Contador de programa
 - Etc.
 - O SO determina qual a acção a tomar para cada tipo de interrupção
 - Transparente para o processo que é interrompido
 - Existem também interrupções geradas por software, nomeadamente para assinalar erros (por ex., divisões por 0)

Estrutura de I/O

■ Operações de I/O

- A comunicação com os dispositivos de I/O é feita através de registos endereçáveis da memória de I/O
- A sinalização de alterações de estado dos dispositivos de I/O é feita através de interrupções
- A comunicação pode ser **Síncrona** (simultâneo) ou **Assíncrona** (não simultâneo)

Estrutura de I/O

■ Operação síncrona

- Após o início duma operação de I/O, o controlo só retorna ao programa depois de terminar a operação. Pelo que o programa a executar a operação de I/O fica à espera da resposta ao seu pedido

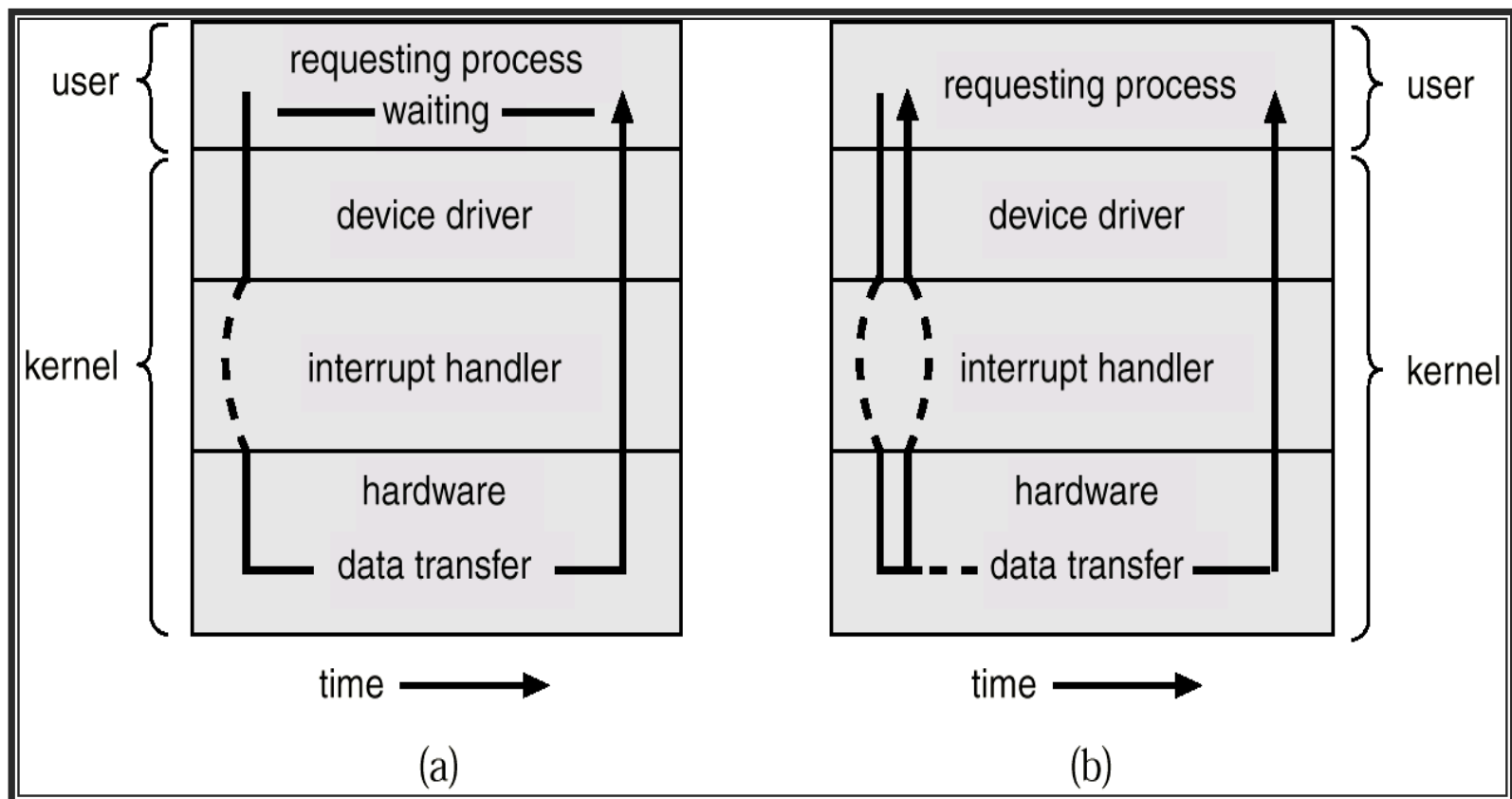
■ Operação assíncrona

- Após o início da operação de I/O o controlo retorna ao programa sem esperar pelo fim da operação. Portanto após a indicação do pedido o programa pode continuar o seu processamento
- A existência de uma resposta é assinalada através de uma interrupção

Estrutura de I/O

Operação síncrona

Operação assíncrona



Estrutura de I/O

■ Operação síncrona

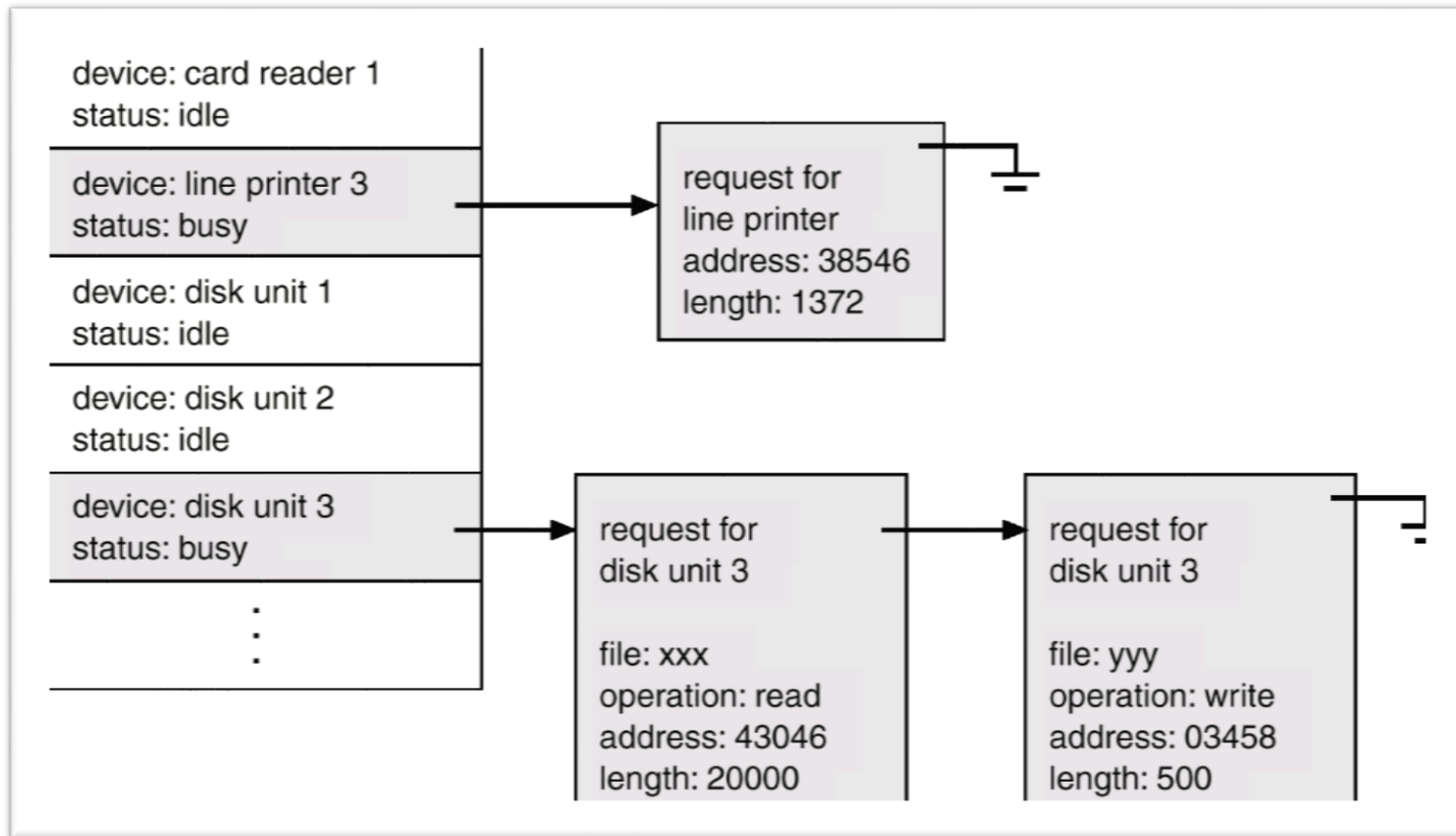
- No máximo um pedido de I/O é atendido de cada vez

■ Operação assíncrona

- permite várias operações de I/O em paralelo
- Permite processamento simultâneo
- No entanto a implementação é mais complexa
- *Device status table* (uma estrutura) contém uma entrada para cada dispositivo de I/O, indicando o seu tipo, endereço e estado
- O SO indexa a *I/O device table* para determinar o estado dum dispositivo e modificar a sua entrada na tabela de forma a reflectir a ocorrência da interrupção

Estrutura de I/O

Device Status Table



Atendimento de Interrupções

1. A tecla é premida pelo utilizador e o carácter é enviado através da porta série para o PC
2. A recepção do carácter levanta uma interrupção na CPU
3. A CPU espera a finalização da instrução corrente antes de atender a interrupção
4. O *Program Counter* é carregado com o valor da rotina de interrupção ao qual é passado o controlo
5. A rotina de interrupção grava os registos do processador
 1. Verifica se existem situações de erro
 2. Lê para um *buffer* o carácter dos registos do controlador série
 3. Ajusta o apontador do *buffer* para a próxima entrada
 4. Acciona uma *flag* no SO que permite informar outras aplicações da existência de dados no *buffer*
6. A rotina de interrupção retorna a CPU ao estado anterior, permitindo a continuação da execução transparente do programa antigo

[Atendimento de Interrupções]

- Um teclado a funcionar a 9600 *bauds* permite transferir um carácter por ms.
- Um rotina de atendimento pode ser executada em 0.002ms
- Sobram 0.998ms para o atendimento de outras interrupções ou para correr programas

Atendimento de Interrupções

- Dispositivos de alta performance não podem operar desta forma!!!!
 - Um disco a transferir um byte de cada vez, permitiria a transferência de 1kB/s – **insuficiente!!!**
 - Se aumentar-mos a taxa de transferência para 100kB/s, temos que aumentar a frequência das interrupções
 - Iremos passar a ter 100 interrupções/ms
 - Durante 1ms a CPU fica ocupada durante 0.200ms (20%) apenas para fazer a transferência de dados

[*Direct Memory Access (DMA)*]

- Usada por dispositivos que necessitem de transferir grandes quantidades de dados da/ou para a memória
- A transferência é feita, por blocos, sem qualquer intervenção da CPU
 - O controlador do dispositivo transfere blocos de dados do *buffer* directamente para a memória principal
- O dispositivo levanta uma interrupção assim que terminar a transferência de um bloco de dados
 - Só há uma interrupção gerada por cada bloco a transferir

[*Direct Memory Access (DMA)*]

- Usada em discos rígidos, leitores e gravadores de CD/DVDs, placas de som, placas de vídeo, etc.
 1. A CPU inicializa o controlador de DMA, indicando-lhe um *buffer* apropriado
 2. A operação de DMA é feita em paralelo com o funcionamento da CPU, roubando-lhe apenas alguns ciclos de relógio (em operações de acesso à memória)
 3. Assim que terminar a operação o controlador de DMA levanta uma interrupção




[Armazenamento de Dados]

- **Memória principal:** meio de armazenamento ao qual apenas a CPU pode aceder directamente
 - Acessível directamente através de um endereço de memória
 - Apenas as instruções na memória principal podem ser executadas pela CPU
- **Memória secundária:** extensão da memória principal que fornece uma grande capacidade de armazenamento não-volátil
 - Discos magnéticos
 - CDRoms, DVDs
 - Memória não volátil

[Armazenamento de Dados]

- Memória principal

- Acessível directamente pela CPU
 - Permite mapear dispositivos de I/O (por ex. placas gráficas ou de rede)
 - São necessários vários ciclos de relógio até aceder a uma posição específica de memória!
- 
- Utilização de memória *cache* de acesso rápido

[Cache]

- Memória de alta performance
 - Normalmente integrada na CPU
- Sempre que a CPU quer ler de uma posição de memória:
 - primeiro verifica se os dados se encontram na *cache*
 - Se sim, utiliza-os
 - Se não, acede directamente à memória principal e grava os respectivos dados na *cache*
- Pode também utilizar a *cache* para escrever, sendo a memória principal posteriormente actualizada

[Memória Secundária]

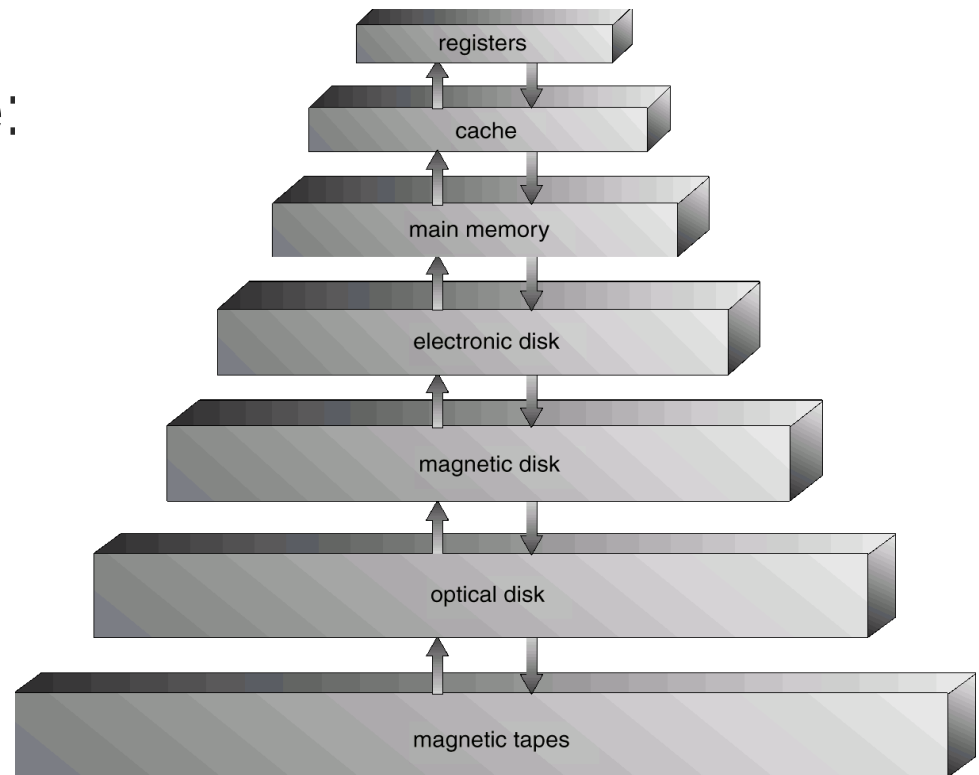
- **Discos magnéticos:** pratos de vidro ou de metal rígido revestidos de material magnético de gravação
 - A superfície do disco está logicamente dividida em pistas (*tracks*), as quais por sua vez estão divididas em sectores (*sectors*)
 - O controlador do disco determina a interacção lógica entre o dispositivo e o computador
 - **O SO deve fazer a interface entre a complexidade de utilização de um disco e o utilizador**

[Cache de Disco]

- A memória principal pode ser vista como a última *cache* para o armazenamento secundário, na memória secundária
- É necessário garantir a coerência da *cache*
 - Em sistemas multitarefa SO é responsável por esta tarefa
 - Por implementar uma política de gestão da *cache*

[Hierarquia de Armazenamento]

- Em função de:
 - Velocidade
 - Custo
 - Volatilidade



[Protecção de Hardware]

- Objectivos:
 - Proteger os diferentes programas a correr de erros cometidos por outros
 - Métodos
 - Operação em *Dual-Mode*
 - Protecção de I/O
 - Protecção de memória
 - Protecção da CPU

[Protecção de Hardware]

- A partilha de recursos do SO requer que o SO assegure que um programa incorrecto não faça com que os outros programas funcionem incorrectamente
- Ex:
 - Permite impedir a um processo o acesso aos recursos de outros processos

[Operação *Dual-Mode*]

- Fornece suporte de hardware para diferenciar pelo menos dois modos de operação, ou seja a CPU pode operar em dois modos:
 - *User-mode*: o programa é executado sob o controlo do utilizador
 - *kernel-mode (monitor-mode)*: o programa é executado sob o controlo do SO, tendo acesso a todos os recursos da máquina

Operação *Dual-Mode*

- Permite o acesso a recursos partilhados através dos serviços oferecidos pelo SO
 - Diminuindo as probabilidades de erro durante essas operações
 - Controlando o acesso concorrente aos recursos do computador
- Algumas instruções apenas podem ser executadas quando um programa está a correr em *Kernel-mode*
- Quando uma interrupção ou excepção ocorre, o hardware comuta para o *Kernel-mode*
- Qualquer erro durante a execução de um programa em *User-mode* pode ser “apanhado” pela CPU e transmitido ao SO
- É necessário assegurar que um programa do utilizador jamais pode adquirir controlo do computador em *Kernel-mode*

[Operação *Dual-Mode*]

- As chamadas ao sistema são tratadas pelo *hardware* como uma interrupção
 - A instrução de interrupção indica o serviço a ser executado
 - Os parâmetros do serviço podem ser passados através de:
 - Registos
 - *Stack*
 - Memória
 - Após a execução da chamada o programa continua do ponto em que encontrava

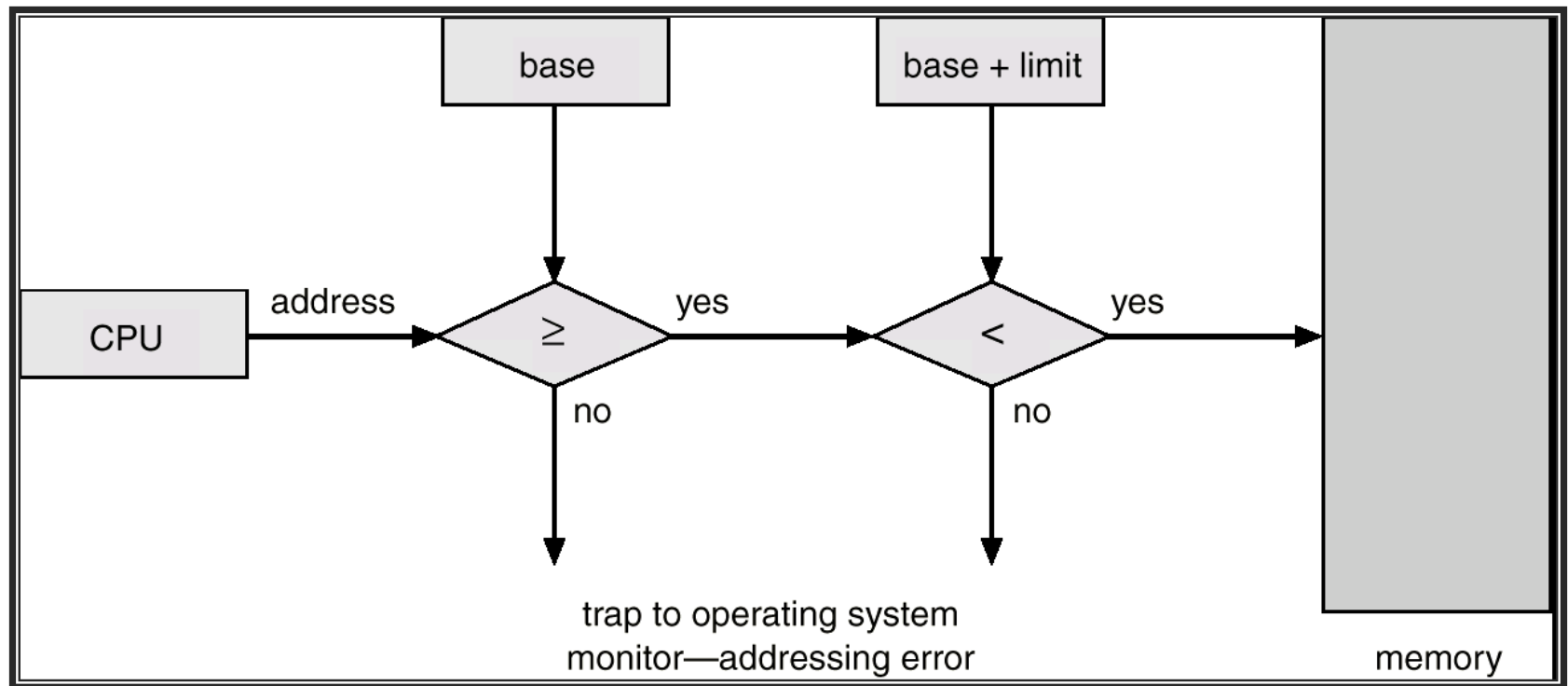
[Protecção de I/O]

- Todas as instruções I/O são instruções privilegiadas \Rightarrow um programa em *User-mode* não as pode executar
- Um programa apenas pode executar operações de I/O através do SO

[Protecção de Memória]

- A maior parte dos erros de programação devem-se a tentativas de acesso a zonas de memória erradas
- O hardware permite definir a área de memória que o programa actualmente em execução poderá aceder
 - *Base register* (início) – guarda o endereço mais baixo de memória física
 - *Limit register* (fim) – contém o tamanho da gama
- A tentativa de acesso a um posição de memória fora desta área gera um erro (*Core Dump*), tratado pelo SO

[Protecção de Memória]



[Protecção da CPU]

- Um *Timer* gera periodicamente uma interrupção ao SO
- Esta interrupção é utilizada pelo SO para permitir a entrada em funcionamento de outros programas ou de rotinas do próprio SO
- Logo:
 - Um programa não ocupa indefinidamente a CPU
 - Permite a multiprogramação (permite a execução periódica do escalonador)



Sistemas Multiprocessador

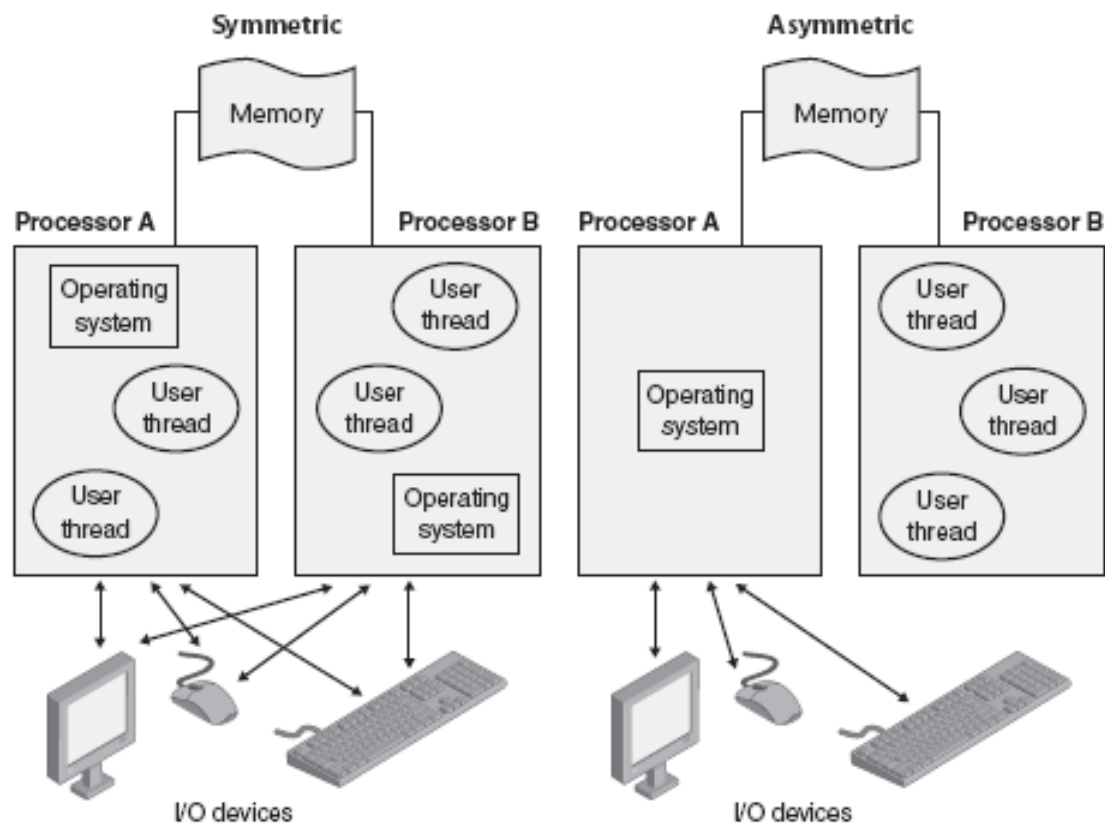
- Sistema constituídos por mais do que 1 processador
- Vantagens
 - Performance mais elevada
 - Economia de escala
 - Partilha de componentes e periféricos
 - Maior fiabilidade
 - Maior disponibilidade
 - Maior tolerância a falhas

Sistemas Multiprocessador

■ Desvantagens

- Pouco software adaptado
- A comunicação entre processadores provoca atrasos e pode ter baixa fiabilidade

Sistemas Multiprocessador



[

]

Sistemas Multiprocessador

■ Sistemas assimétricos

- As chamadas ao SO apenas correm numa das máquinas
- Os outros processadores correm os processos existentes no sistema
- Funciona bem para um número de processadores limitado
- Em sistemas de larga escala o processador que contém o SO torna-se o *bottleneck* do sistema

Sistemas Multiprocessador

- Sistemas simétricos
 - Existe uma cópia do SO na memória de cada CPU
 - Será que se podem correr chamadas ao SO concorrentemente por cada processador?
 - Suponha que quer alocar um determinado espaço de memória.

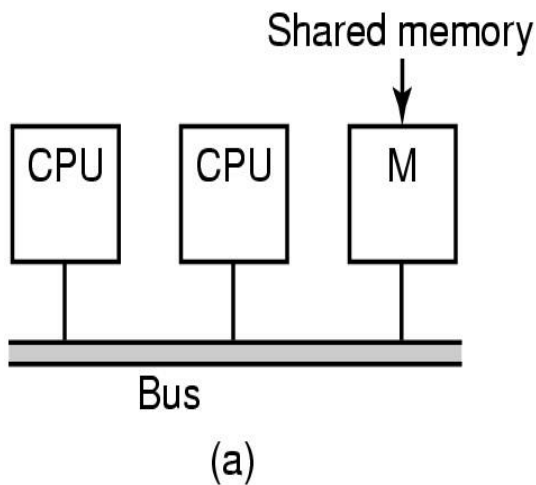
Sistemas Multiprocessador

- Sistemas simétricos

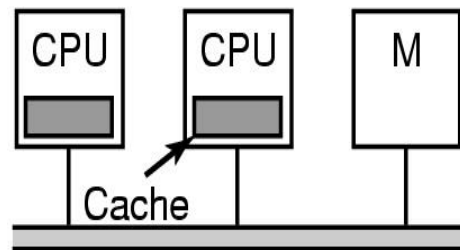
- É necessário utilizar locks que impeça o SO de correr concorrentemente quando outras CPUs querem efectuar chamadas ao SO
 - Os locks devem defender partes independentes do SO
 - Os sistema operativos devem ser desenhados de modo que os diversos módulos seja realmente independentes

Sistemas Multiprocessador

- Todos os processadores partilham a mesma memória
 - Se o número de processadores aumentar vai também aumentar a contenção no acesso ao meio



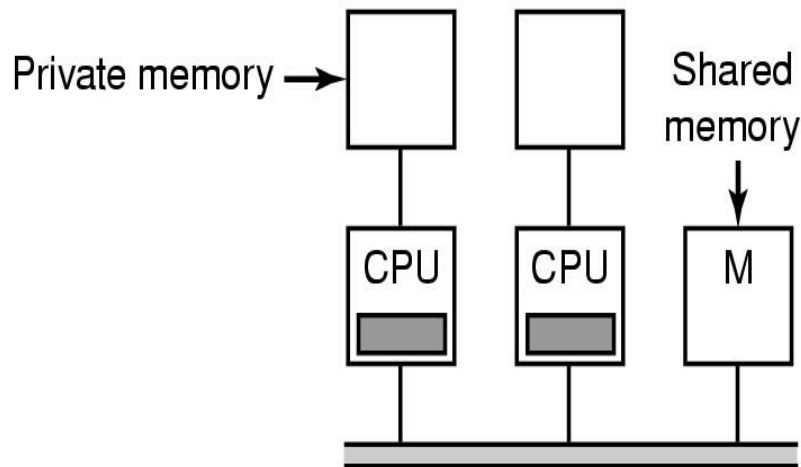
Sistemas Multiprocessador



(b)

- Cada processador tem a sua cache
 - Diminuição de leituras através do barramento
 - read-only
 - read-write
 - Quando existe uma escrita, a alteração deve ser propagada para todas as outras caches e memória principal
 - A MMU de cada processador informa os outros quais os blocos que foram alterados
 - O acesso a um desses blocos obriga à leitura de todo o bloco

Sistemas Multiprocessador



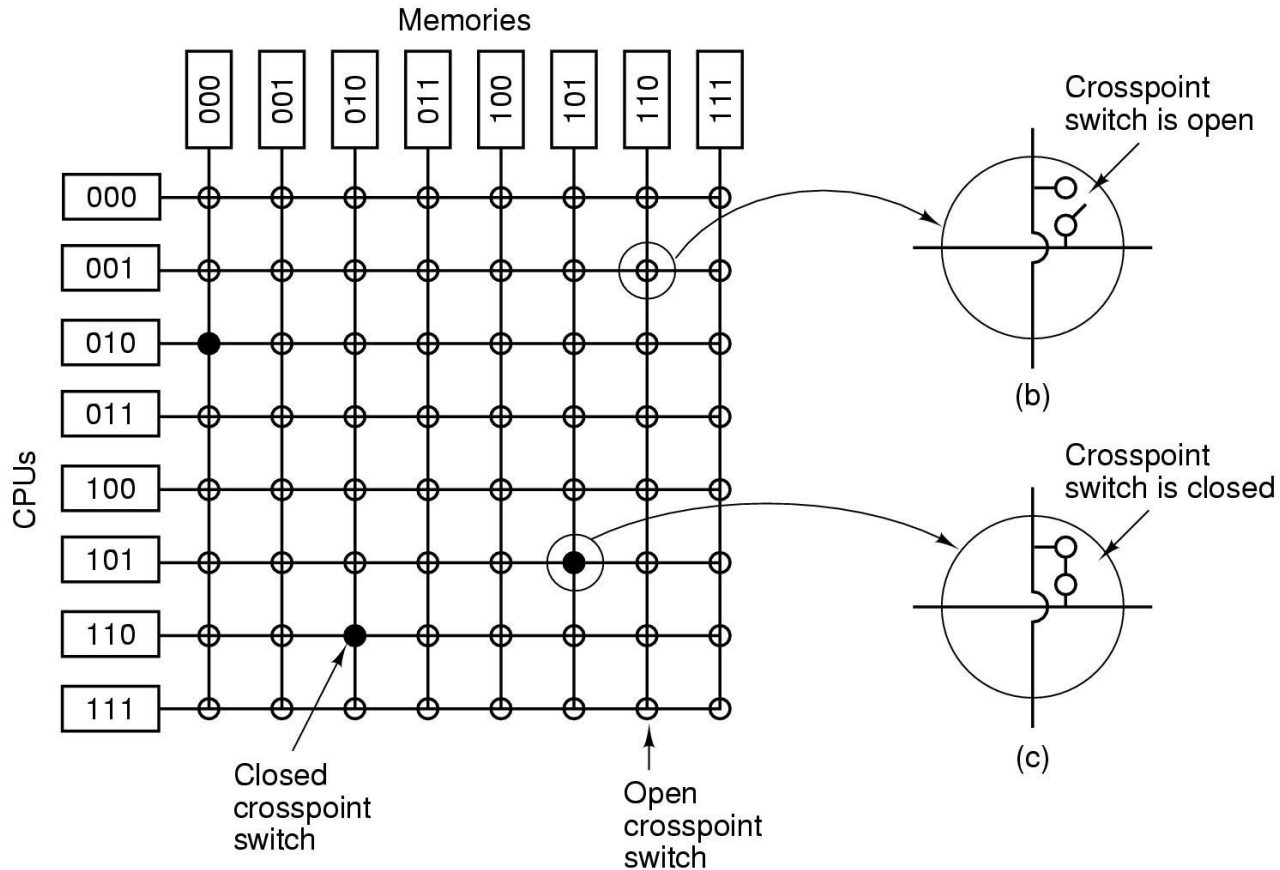
(c)

- Cada processador tem a sua cache e memória privada
 - Permite colocar toda a informação apenas de leitura na memória privada
 - Obriga no compilador a conhecer a configuração do sistema e colocar as parte apenas de leitura nas memórias privadas

Sistemas Multiprocessador

- Limitações das arquitecturas anteriores
 - A **contenção** no barramento limita o número de processadores a valores entre 16 e 32.
- Solução:
 - *Switching* para acesso à memória

Sistemas Multiprocessador

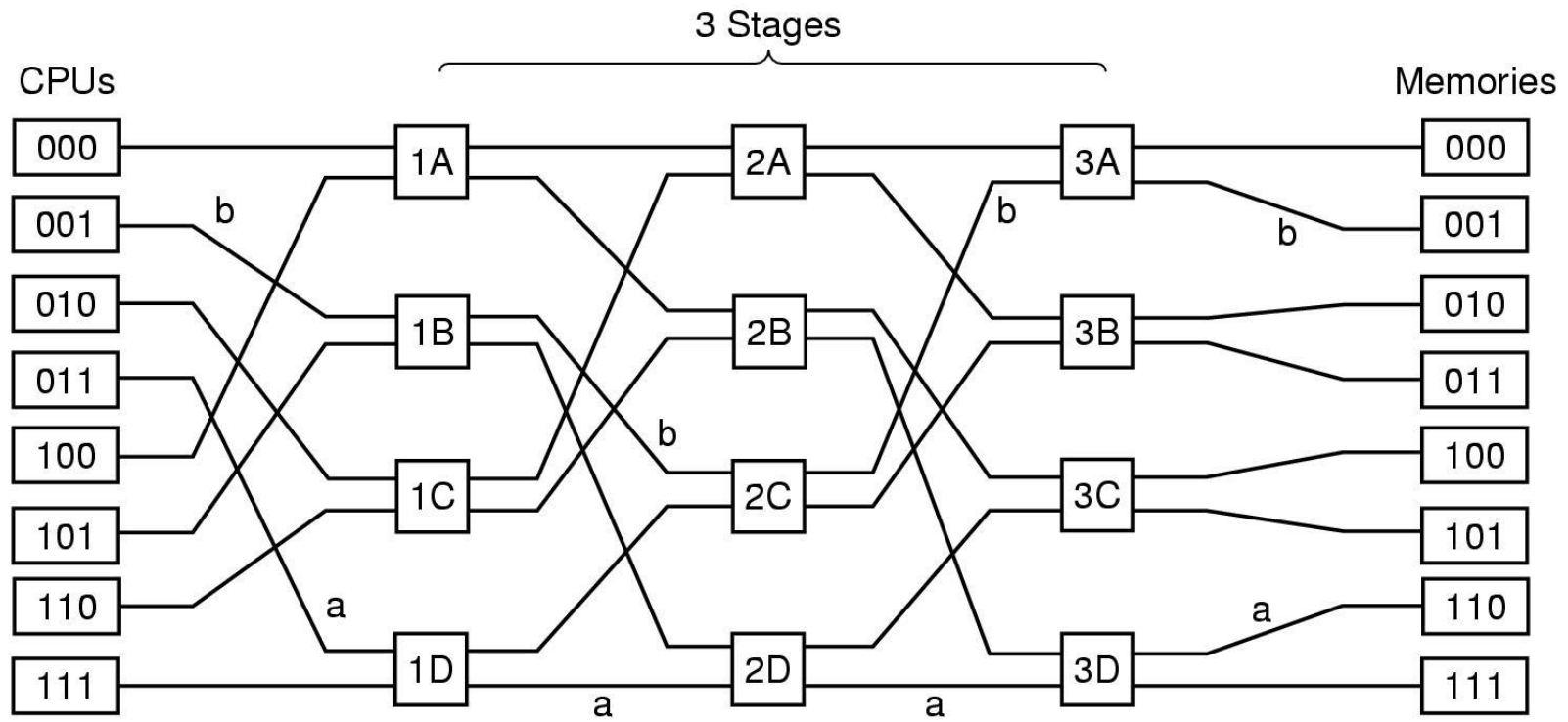


Sistemas Multiprocessador

- Crossbar Switching
 - A memória é dividida em blocos individuais
 - Cada bloco pode ser acedido independentemente por cada processador
 - Permite que outros processadores acessem em paralelo a outros blocos de memória
 - Estrutura complexas o número de conexões é n^2 (num sistema com n processadores)

Sistemas Multiprocessador

■ Multistage Switching

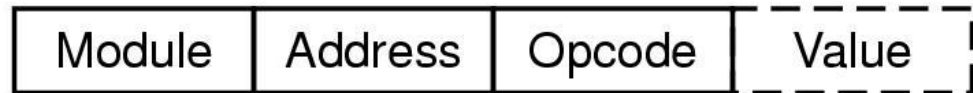


Sistemas Multiprocessador

- Multistage Switching



(a)



(b)

- **Module**: módulo de memória ao qual se quer aceder
- **Address**: endereço de memória, dentro desse módulo
- **OpCode**: operação a realizar {read, write}
- **Value**: conteúdo no caso da escrita

Sistemas Multiprocessador

- Para n CPU e n memórias o número de switches é igual a:
$$(n/2)\log_2 n \ll n^2$$
- Podem existir situações de bloqueio!
- Tenta-se atribuir cada módulo de memória a um processador de modo a reduzir os conflitos
- Solução cara e complexa

[Clusters]

- Sistemas em que os computadores (com um mais processadores) estão ligados através de uma rede local de alta performance (p.e. Gigabit ethernet ou InfiniBand)
 - Vantagens:
 - Maior fiabilidade
 - Operação em modo standby: um dos nós está pronto a substituir o outro caso ele falhe
 - Operação simétrica: dois ou mais nós executam as mesmas funções e podem substituir-se um ao outro
 - Escalabilidade

Sistema Operativo

- Um Sistema Operativo pode ser visto como um programa de grande complexidade, responsável pela gestão eficiente de **todos os recursos de um computador**
- Composto por um conjunto de camadas funcionais (módulos)
- Cada módulo constitui um nível de abstracção que implementa uma máquina virtual com uma interface bem definida

[Sistema Operativo]

- Objectivos

- Interagir com o hardware
- Disponibilizar um ambiente fiável para execução de aplicações

Módulos de um Sistema Operativo

- Gestão de Processos
- Gestão da Memória Principal
- Gestão de Ficheiros
- Gestão de I/O
- Gestão da Memória Secundária
- Gestão de Rede
- Protecção do Sistema
- Interpretador de Comandos

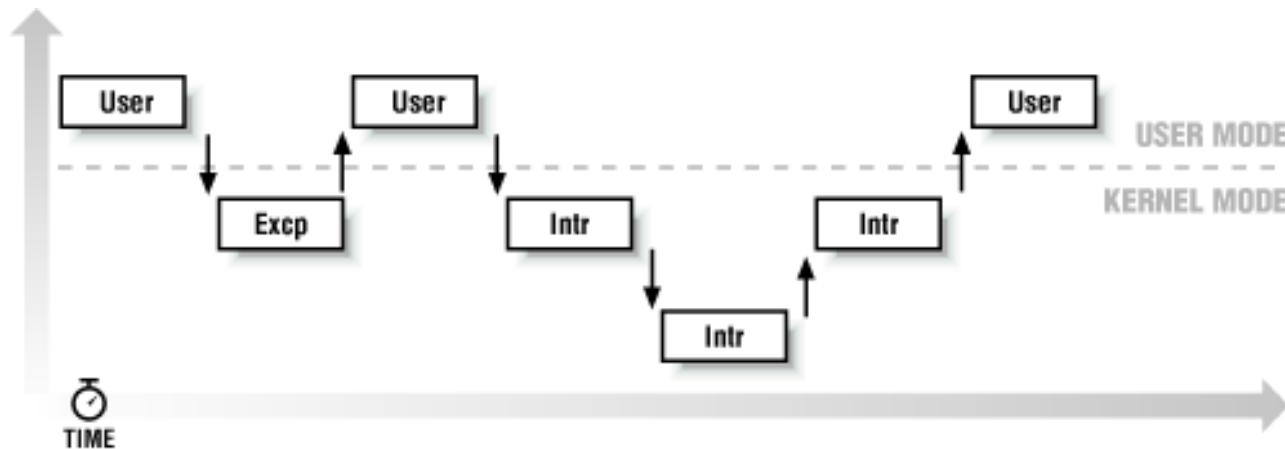


- Como é que o utilizador interage com o SO?

Chamadas ao Sistema

- Fornecem uma interface entre o utilizador e os módulos de SO
- Podem ser acedidas através de instruções em *assembly* (MS-DOS) ou através de linguagens de alto nível (UNIX e Windows)
- Passagem de parâmetros
 - Através dos registos do CPU
 - Armazenamento em memória
 - Através do *stack* das funções
- Correm em *Kernel Mode*

[Operação em dual-mode]

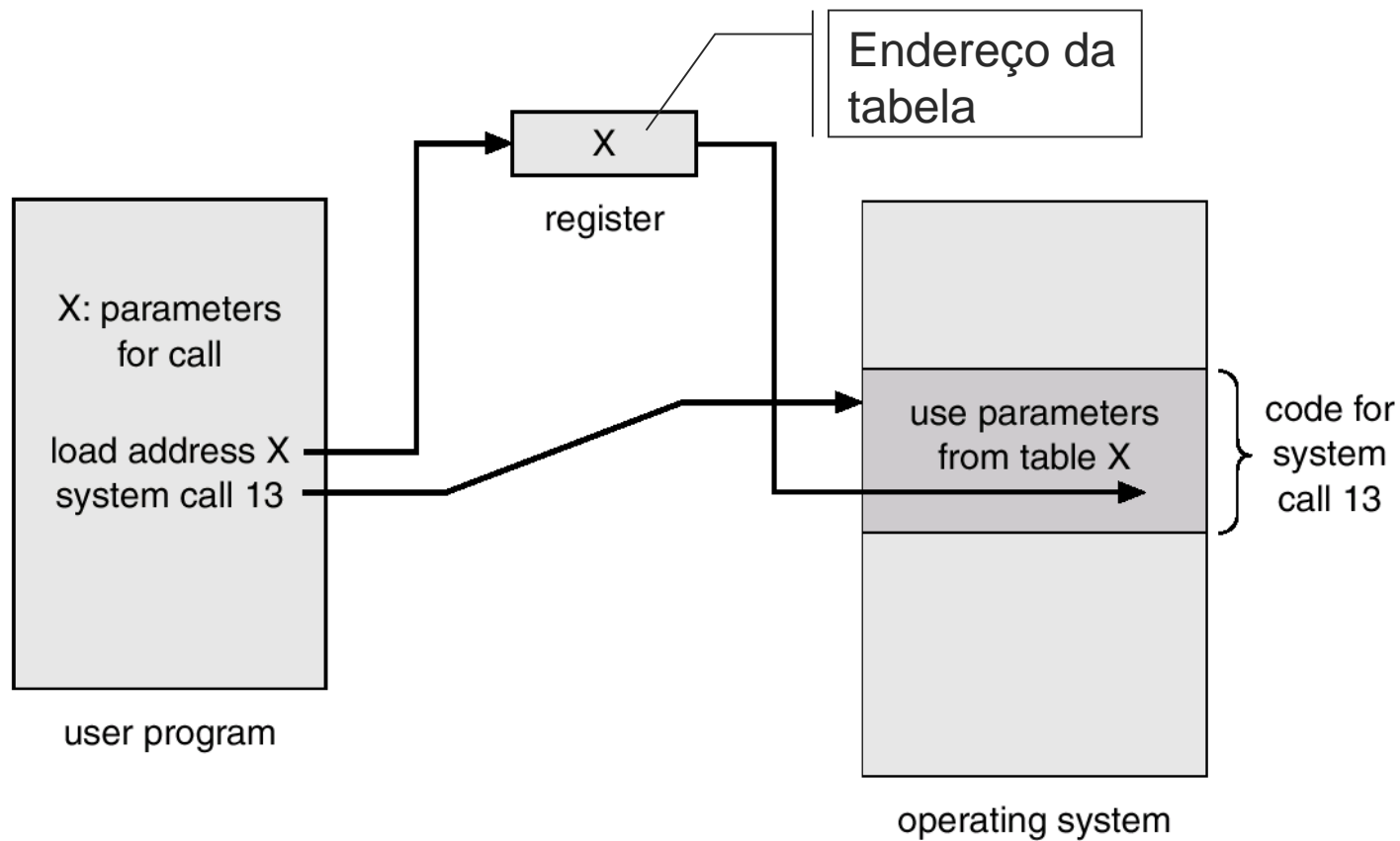


Do livro “Understanding the Linux Kernel”

[Chamadas ao Sistema]

- Em linguagens de alto-nível (C, C++, Visual Basic) as chamadas ao sistemas encontram-se nas funções existentes nas livrarias
- Raramente utilizadas (directamente) por um programador comum

Chamadas ao Sistema



Chamadas ao Sistema - Linux

- Através da instrução `int 0x80`
 - Que gera uma interrupção por software
 - O registo `eax` vai conter o número da interrupção/chamada ao sistema
 - Os restantes registos os parâmetros da chamada ao sistema
 - Podem também conter apontadores para outras zonas de memória
 - Os parâmetros podem também ser passados através do *stack* do programa
- As chamadas ao sistema retornam sempre pelo menos um número inteiro, que depois vai servir para actualizar a variável `errno`

Chamadas ao Sistema - Linux

■ Alguns exemplos

%eax	Nome	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	int	-	-	-	-
2	sys_fork	Struct pt_regs	-	-	-	-
3	sys_read	unsigned int	Char *	Size_t	-	-
4	sys_write	unsigned int	Char *	Size_t	-	-

[Chamadas ao Sistema]

- Tipos de chamadas
 - Controlo de processos
 - Gestão de ficheiros
 - Gestão de dispositivos
 - Gestão de informação
 - Comunicações

[Processador, Programa e Processo]

- O **Processador** é o órgão material onde é executada toda a actividade do sistema
- Um **Programa** é um conjunto de instruções armazenadas num ficheiro
- Um **Processo** é uma instância de um programa em execução. No entanto, um programa pode ser constituído por vários processos

Gestão de Processos

■ Processos:

- Cada processo pode ser considerado como um programa em execução
- Abstracção do SO que contém dados referentes ao código a executar, às variáveis, à pilha (*stack*), às áreas de memória, aos parâmetros, etc.

■ Exemplos:

- Comandos ou programas em execução
- *Shell*
- Processos do sistema: *syslog*, *rpciod*, etc.

[Processos]

- Um Processo define:
 - Um conjunto de operações
 - Operações elementares, normalmente um subconjunto das instruções do processador
 - Operações de interacção com outros processos
 - Um espaço de endereçamento
 - Um processo executa-se dentro de um espaço de endereçamento bem delimitado, evitando que possa aceder e forma indevida ao espaço de endereçamento de outros processos ou do próprio Sistema Operativo

Gestão de Processos

- SO fornece serviços para:
 - Criação e eliminação de processos
 - Escalonamento de processos (multiprogramação)
 - Tratamento das interrupções
 - Mecanismos para sincronização de processos
 - Mecanismos para a comunicação de processos

Chamadas de Controlo de Processos

- *end*
 - Finalização ordenada de um processo
- *abort*
 - Finalização de um processo devido a um erro
- *load*
 - Permite carregar um programa em memória
- *execute*
 - Executa de forma controlada um outro programa, o processo **Pai** pode executar o outro processo de forma concorrente ou pode substituir um processo pelo outro

Chamadas de Controlo de Processos

- *create process*
 - Permite a criação de um novo processo
- *terminate process*
 - Força a finalização de um processo **filho** do processo evocado
- *get process attributes, set process attributes*
 - Permite operar sobre os atributos do processo criado: alterar prioridade, máximo tempo de execução, etc.
- *wait for time (sleep)*
 - Suspende a execução do processo durante um determinado tempo

Chamadas de Controlo de Processos

- *wait event*

- Bloqueia um programa até que um determinado evento aconteça, por ex:
 - Um outro processo filho termine
 - Receba um sinal

- *signal event*

- Permite sinalizar a ocorrência de um evento ao SO

- *allocate, free memory*

- Alocar e libertar memória

[

]

[Gestão da Memória Principal]

- Controla a utilização da memória física
- Cada posição de memória, *byte* ou *word* é endereçada individualmente
- A memória é utilizada para armazenar:
 - Código referente a processos
 - Dados e a *stack* referente a cada processo
 - Comunicação com os dispositivos de I/O

Gestão da Memória Principal

- Os algoritmos de alocação de memória devem ter como objectivo reduzir a fragmentação da memória
 - *First Fit*
 - Aloca o primeiro pedaço de memória livre que tenha espaço suficiente
 - *Best Fit*
 - Aloca o pedaço de memória livre mais pequeno mas com espaço suficiente para conter os dados
 - *Worst Fit*
 - Aloca o maior pedaço de memória livre

Gestão da Memória Principal

- Funções do Sistema de Gestão da Memória Principal:
 - Registo actualizado das zonas de memória sob utilização e por que processo
 - Decisão sobre os processos a carregar em memória face ao espaço ainda disponível em memória
 - Reservar e libertar dinâmicamente espaço de memória

Serviços relacionados

- *malloc*

- Aloca um bloco de memória para uma aplicação

- *free*

- Liberta um bloco de memória

Gestão de Ficheiros

- O Sistema Operativo fornece uma visão uniforme do sistema de ficheiros, independentemente da tecnologia usada
- Ficheiro:
 - Colecção de informação relacionada entre si
 - Programas
 - Dados
 - Organizados por directórios

Gestão de Ficheiros

- Funções do Sistema de Gestão de Ficheiros:
 - Criar/Apagar ficheiros e directórios
 - Operações de leitura e escrita em ficheiros
 - Mapeamento dos ficheiros no disco
 - Escalonamento do acesso ao disco
 - Protecção de acesso aos ficheiros

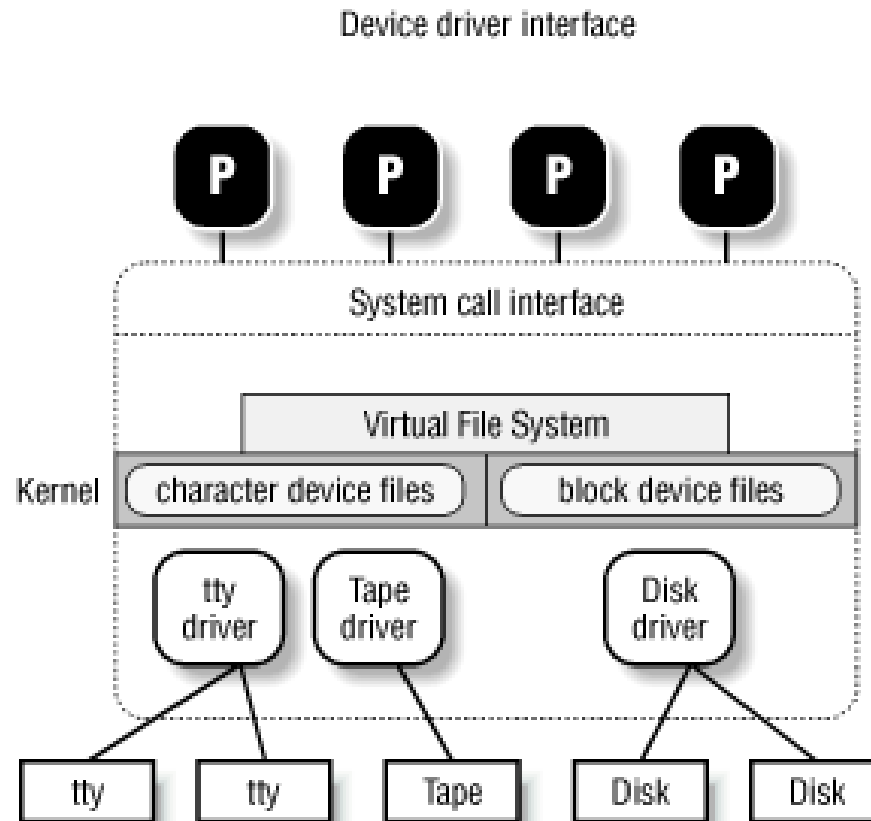
Chamadas para Manipulação de Ficheiros

- *create/delete file*
- *open, close*
- *read, write, reposition*
- *get/set file attributes*
 - Nome, tipo, permissões...

Gestão de I/O

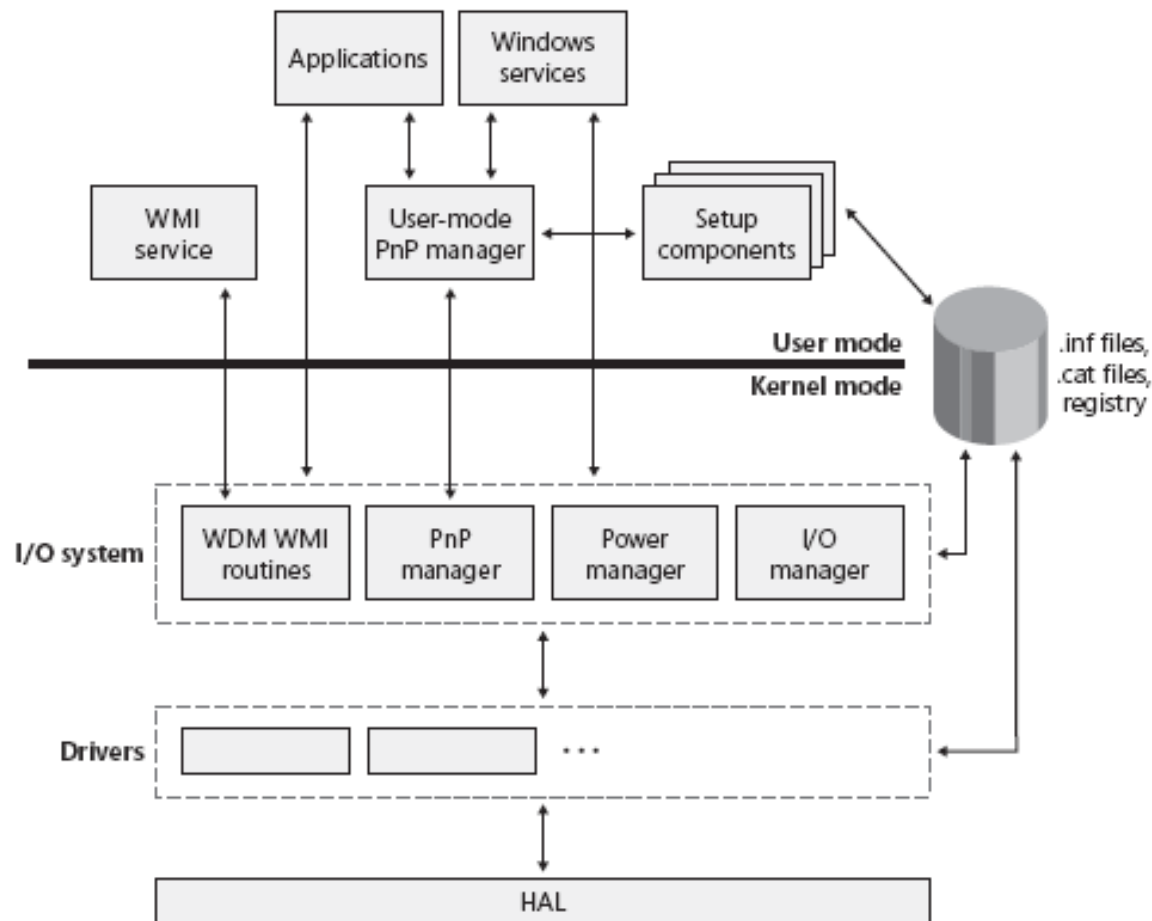
- A implementação das operações de I/O é complexa, uma vez que interactuam com o hardware dos dispositivos.
- Uma das principais funções do SO é esconder as especificidades do hardware ao utilizador
 - Implementado através de *Device Drivers*
- Componentes de I/O
 - Sistema de buferização, *caching* e *spooling*
 - Interface genérica para *device drivers*
 - *Device Drivers* específicos

Sistema de I/O do Linux



Do livro “Understanding the Linux Kernel”

Sistema de I/O do Windows



Gestão da Rede

- O Sistema Operativo disponibiliza serviços para:
 - A comunicação com outras máquinas, constituindo um sistema distribuído
- Um **Sistema Distribuído** é definido como um conjunto de computadores que comunicam através de uma rede partilhando os seus recursos e funcionalidades, com objectivos comuns
 - Partilha de ficheiros
 - Aplicações de bases de dados
 - Servidores web

[Gestão da Rede]

- Funções do Sistema de Gestão da Rede:
 - Oferece uma interface e protocolos de comunicação normalizados que permitem a comunicação entre diferentes máquinas
 - Gerir a configuração e os parâmetros de rede
 - Exemplos:
 - TCP/IP
 - SMB
 - FTP
 - NFS
 - etc.

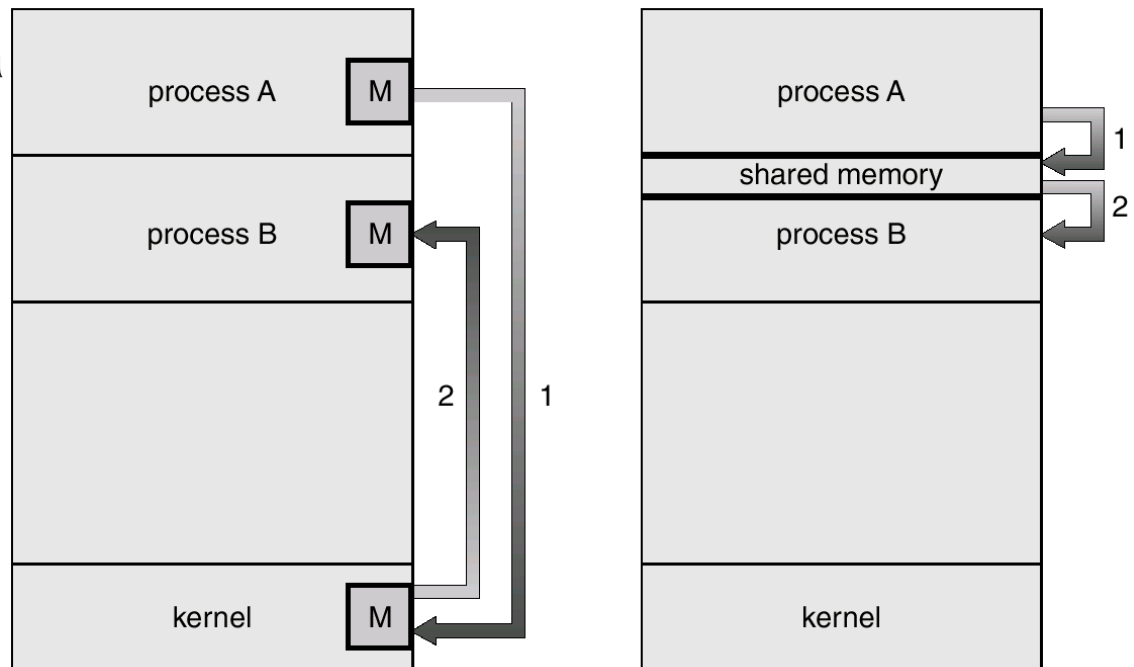
[Chamadas para Comunicações]

- *create*
 - Estabelece um canal de comunicações entre 2 processos
- *delete communication connection*
- *send, receive messages*
- *transfer status information*
- *attach, detach remote devices*

Chamadas para Comunicações

- Modelos de comunicação:

- Passagem de mensagens
- Memória partilhada



[Protecção do Sistema]

- O Sistema Operativo permite controlar o acesso pelos programas e processos aos recursos do sistema, autorizando ou não o acesso, assim como, o tipo de permissões atribuídas

[Protecção do Sistema]

■ Exemplos:

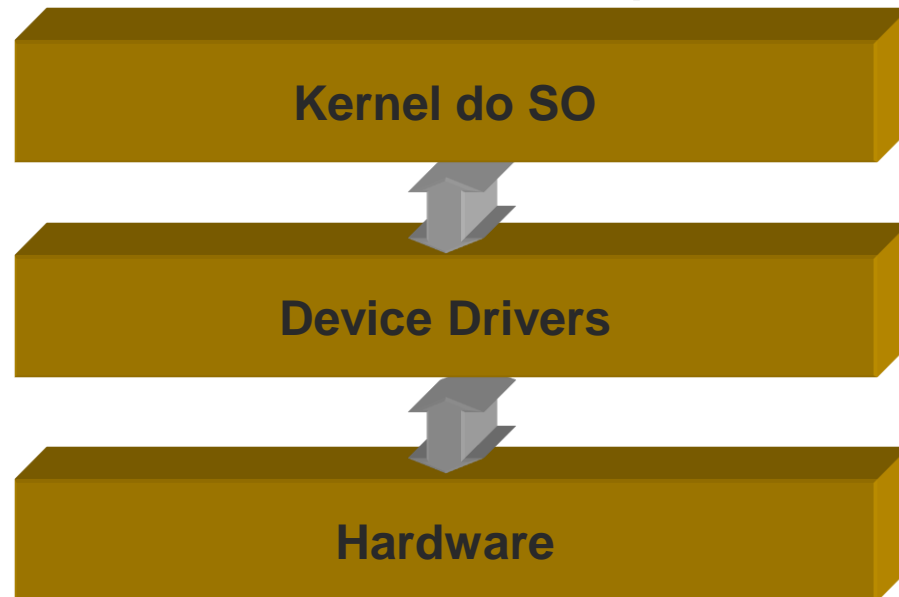
- Sistema de ficheiros
- Acesso a dispositivos de I/O
- Restrição do acesso a áreas de memória de outros processos
- Detecção de erros (evitando a propagação do erro aos restantes processo em execução)

[Interpretador de Comandos (*Shell*)]

- Fornece uma Interface entre o utilizador e o Sistema Operativo, permitindo que o utilizador possa correr comandos do SO
- Exemplos:
 - No MS-DOS a *shell* está incorporada no núcleo do SO
 - Em *Linux* a *shell* é um programa à parte que interage com o SO através de chamadas ao sistema
 - A *shell* também pode ser vista como uma interface gráfica:
 - *Windows/explorer*
 - *Linux/Gnome/KDE/Xwindows*

Estrutura por Camadas

- O SO é dividido em várias camadas, cada uma delas fornece serviços apenas à camada imediatamente por cima

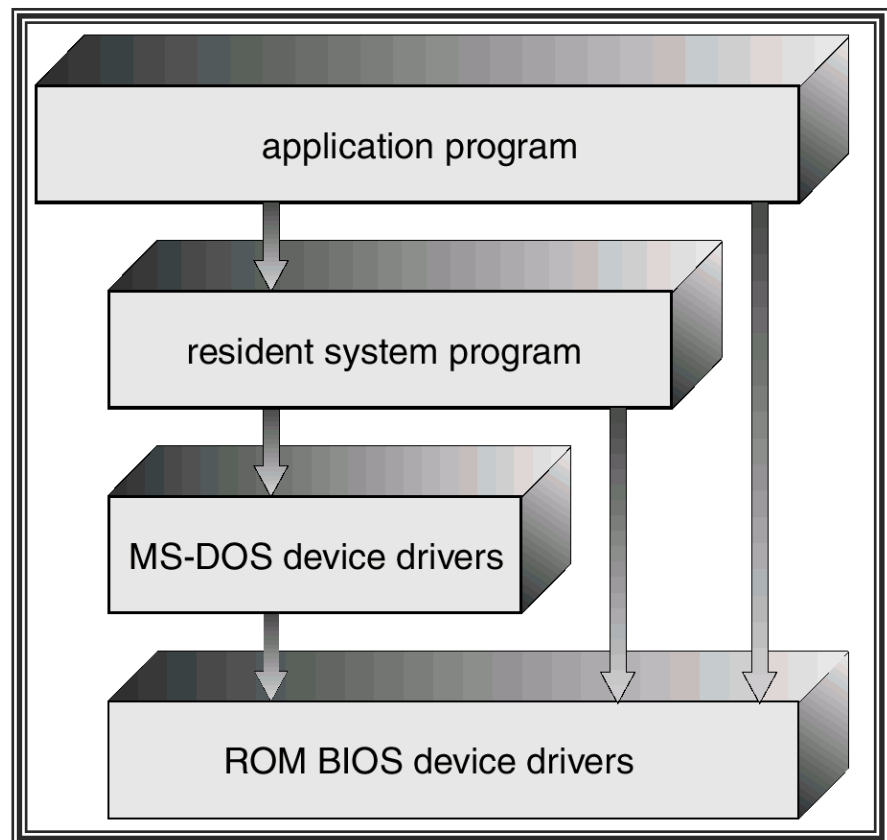


Estrutura de um SO

MS-DOS

Características:

- Baixa modularidade
- Por **camadas**
- Necessita de muito poucos recursos



[Tipos de S.O.]

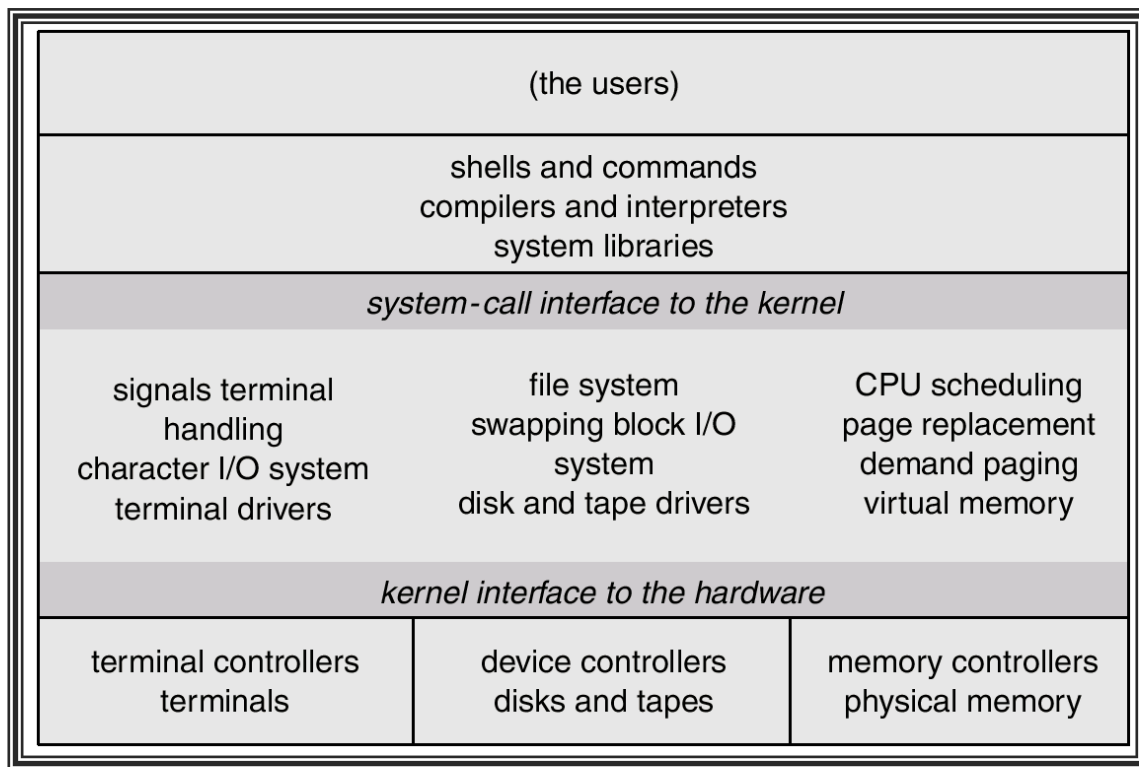
- Computadores Desktop e servidores
 - Linux, Windows, Mac OS X
- Sistemas multimédia
 - Media centers, decodificadores TV cabo
- Sistemas embebidos
 - Automóveis, aviões, electrodomésticos
- Sistemas de tempo-real
 - Sistemas em que o sucesso de uma computação depende, não só da sua correcção, mas também do tempo que fica disponível.
- Sistemas móveis
 - Telemóveis, PDAs, outros terminais móveis

[Estrutura de um SO]

Características:

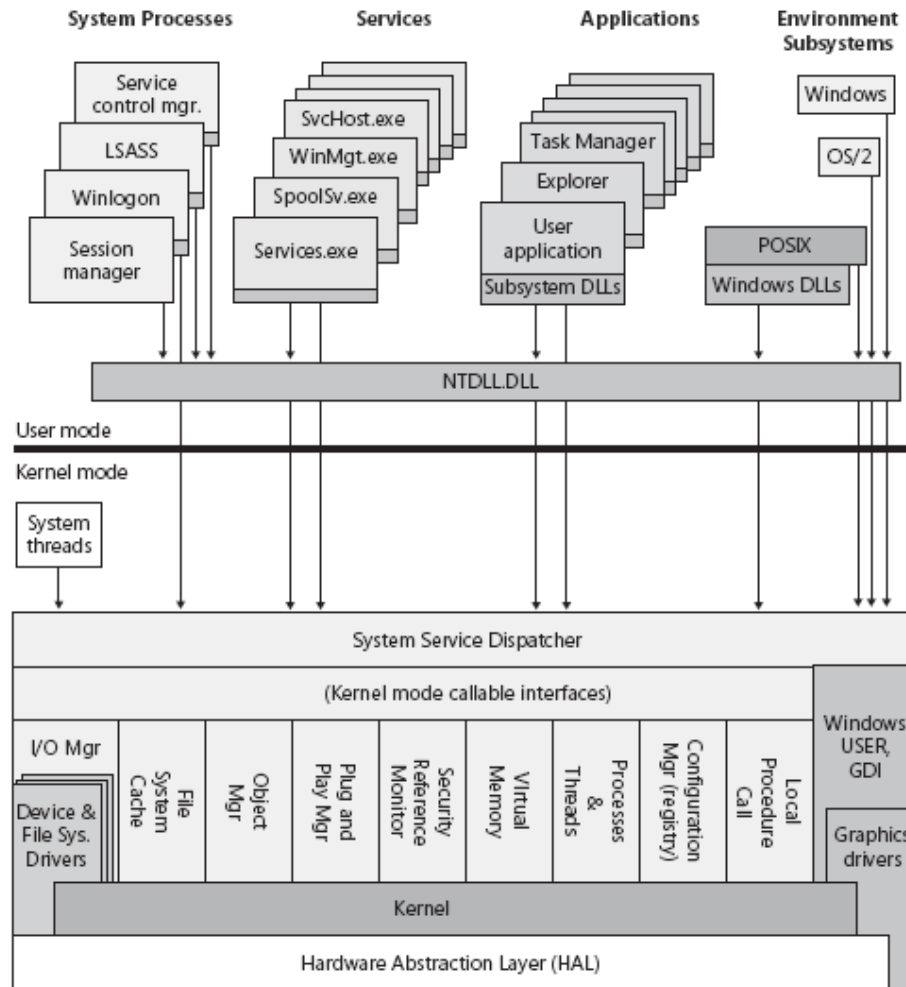
- **Modular**
- **Por camadas**
- **Necessita de elevados recursos**

UNIX



Estrutura de um SO

Windows



Hardware interfaces (buses, I/O devices, interrupts, interval timers, DMA, memory cache control, etc.)

[Estrutura de um SO]

- Windows

- O windows suporta sistemas multiprocessador utilizando uma arquitectura simétrica
- Suporta:
 - Hyperthreading
 - Permite ter vários processadores a partilhar a mesma cache
 - Non Uniform Multiprocessors (NUMA)



Sistemas Operativos I

Arquitectura de um Sistema Operativo

Luis Lino Ferreira

Fevereiro de 2009

Bibliografia

- Estes slides foram baseados nas fontes seguintes:
- "Operating System Concepts: 7th Edition", Avi Silberschatz and Peter Galvin, John, Wiley & Sons, 2001, <http://codex.cs.yale.edu/avi/os-book/os7/>
- "Operating Systems", William Stallings, Prentice Hall, 2005, <http://www.williamstallings.com/OS/OS5e.html>
- "Windows Internals, 4th Edition", Mark Russinovich, Davis Soloman, Microsoft Press
- "Modern Operating Systems, 2nd Edition", Andrew Tanenbaum, <http://www.cs.vu.nl/~ast/books/mos2/>

[Gestão de Dispositivos]

- Objectivo:
 - Oferecer ao utilizador uma interface standard e de fácil utilização, incluindo
 - Gestão de memória: buffering, caching e spooling
 - Interface estandardizada de acesso aos drivers

Chamadas para Gestão de Dispositivos

- *request*
 - Solicita ao SO o acesso exclusivo a um dispositivo
- *release device*
 - Sinaliza ao SO que já não necessita de um determinado recurso
- *read, write, reposition*
 - Em Linux estas operações são muito semelhantes à leitura e escrita em ficheiros
- *get/set device attributes*
- *attach/detach device*