

Sistemas Operacionais

Introdução

Sumário

- 1 Introdução
- 2 Objetivos
- 3 Categorização
- 4 GUI vs CLI

- 5 Gerenciamento de Recursos
- 6 Estrutura
- 7 Arquitetura
- 8 Bibliografia

Introdução

O que é um sistema operacional?

Sua função é fornecer aos programas do usuário um modelo do computador melhor, mais simples e mais limpo, assim como lidar com o gerenciamento de todos os recursos.

(TANENBAUM, 2010)

O sistema operacional é uma camada de software entre o hardware e os programas que executam tarefas para os usuários.

(OLIVEIRA; CARISSIMI; TOSCANI, 2001)

- › Em resumo, sistemas operacionais oferecem:
 - › **Conveniência**;
 - › **Eficiência**.
- › Tem aspectos de baixo nível (*drivers*);
- › Tem aspectos de alto nível (*interface* e utilitários).

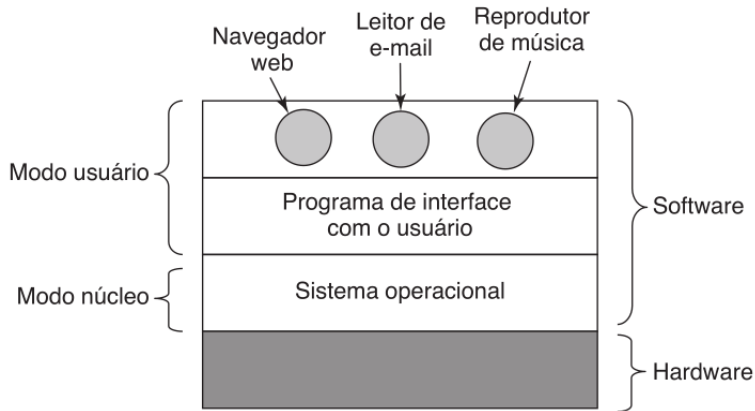


Figura 1: Sistema Operacional (TANENBAUM, 2010).

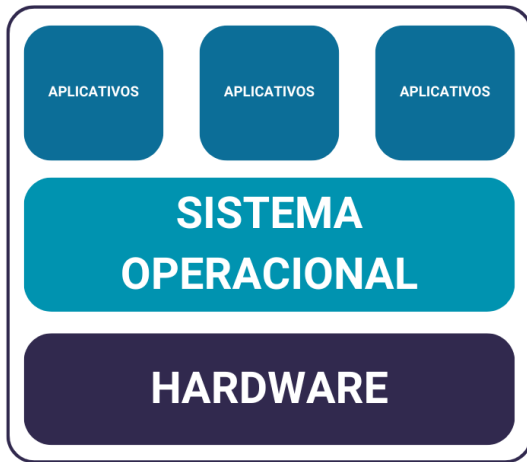


Figura 2: Estrutura de um computador.

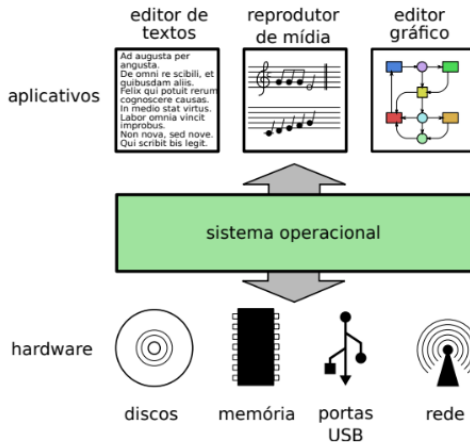


Figura 3: Estrutura de um computador (Maziero, 2019).

Objetivos

› Abstração de recursos

- ›› Prover interfaces de acesso aos dispositivos.
- ›› Tornar os aplicativos independentes do *hardware*.
- ›› Definir interfaces de acesso **homogêneas** para dispositivos com tecnologias distintas.

Exemplo: Abrir um arquivo.

- 1 Verificar se os parâmetros informados estão corretos (nome do arquivo, identificador do disco, buffer de leitura, etc.);
- 2 Verificar se o disco está disponível;
- 3 Ligar o motor do disco e aguardar atingir a velocidade de rotação correta;
- 4 Posicionar a cabeça de leitura sobre a trilha onde está a tabela de diretório;
- 5 Ler a tabela de diretório e localizar o arquivo ou subdiretório desejado;
- 6 Mover a cabeça de leitura para a posição do bloco inicial do arquivo;
- 7 Ler o bloco inicial do arquivo e depositá-lo em um buffer de memória.

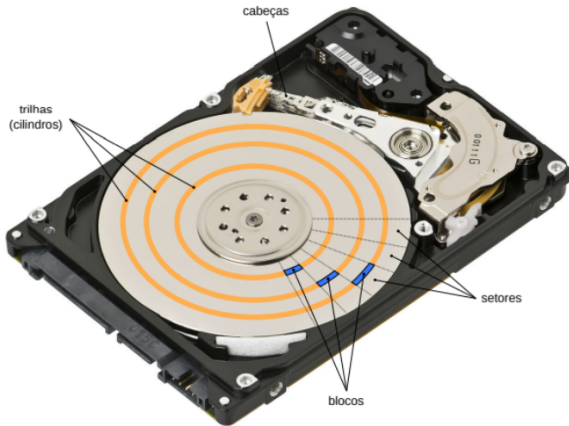


Figura 4: HD (Maziero, 2019).

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(void){
4     FILE *arquivo = fopen("nome_arquivo.txt", "w");
5     if(arquivo == NULL)
6         printf("Não foi possível abrir o arquivo!\n");
7     fclose(arquivo);
8     return 0;
9 }
```

Código 1: Exemplo fopen() em Linguagem C.

› Gerenciamento de recursos

- ›› Prover interfaces de acesso aos dispositivos.
- ›› Tornar os aplicativos independentes do *hardware*.
- ›› Definir interfaces de acesso **homogêneas** para dispositivos com tecnologias distintas.

Exemplo: Gerenciar Fila de Impressão.

- A impressora é um recurso cujo acesso deve ser efetuado de forma mutuamente exclusiva (apenas um aplicativo por vez).
- O sistema operacional resolve essa questão definindo uma fila de trabalhos a imprimir normalmente atendidos de forma sequencial (FIFO).

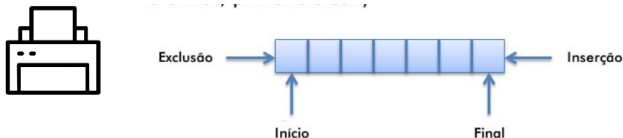


Figura 5: Fila de Impressão.

Nota:

Em resumo: Um sistema operacional objetiva abstrair o acesso e o gerenciamento dos recursos de *hardware*.

Oferece aos aplicativos um ambiente onde eles podem usar esses recursos através de interfaces fáceis de usar, sem precisar se preocupar com detalhes técnicos e minimizando conflitos no uso do *hardware*.

Categorização

› Sistemas Operacionais de lotes (*batch*):

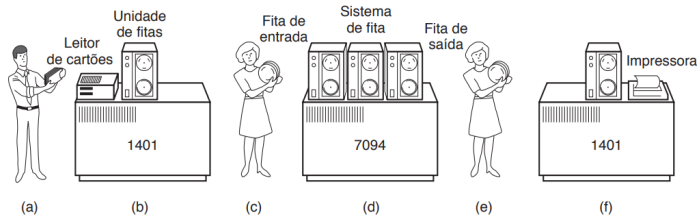
- ›› Trabalham “por lote”.
- ›› Não há interferência do usuário durante a execução.
- ›› Programas e dados eram colocados em uma fila para execução.
- ›› O processador processava os programas sem interação com os usuários.
- ›› Isso permitia um alto grau de utilização do sistema.

› Aplicação Atual:

- ›› Conceito ainda é usado em sistemas que processam tarefas sem interação direta com os usuários.

Tipos de Serviço II

Sistemas Operacional *Batch*



- (a) Programadores levavam cartões para o 1401.
(b) O 1401 lia o lote de tarefas em uma fita.
(c) O operador levava a fita de entrada para o 7094.
(d) O 7094 executava o processamento.
(e) O operador levava a fita de saída para o 1401.
(f) O 1401 imprimia as saídas.

Figura 6: Sistema de Lote antigo (TANENBAUM, 2010).

Tipos de Serviço III

Sistemas Operacional *Batch*



Figura 7: IBM 7094 OS/360.

- › **Sistemas Operacionais de tempo compartilhado (*time-sharing*) ou multitarefas:**
 - » Sistemas que permitem que múltiplos usuários compartilhem o tempo de um único processador (multiprogramação).
 - » Usuários podem interagir com o sistema em tempo real.
 - » O número de tarefas processadas por unidade de tempo é mais baixo.
 - » Recursos de hardware, como CPU e memória, são compartilhados entre os usuários e as tarefas.

- › **Sistemas Operacionais de tempo real (*real-time*):**
 - ›› Sistemas que garantem resposta a eventos dentro de um prazo específico.
 - ›› Sistemas dedicados.
 - ›› Devem atender requisitos de tempo.
 - ›› Alta disponibilidade e robustez.
- › **Exemplos de Aplicações:**
 - ›› Sistemas de controle de processos industriais.
 - ›› Sistemas de navegação e controle de aeronaves.
 - ›› Sistemas de controle de veículos autônomos.
 - ›› Sistemas de comunicação em tempo real.

› Atenção!

- » Enquanto todos os sistemas de tempo real podem ser considerados embarcados se forem projetados para um dispositivo específico, nem todos os sistemas embarcados são sistemas de tempo real.
- » O foco principal de um sistema embarcado é a especialização e integração, enquanto o foco de um sistema de tempo real é garantir que as operações atendam a prazos críticos e sejam previsíveis.

- › **Sistema de Mono usuário:**

- » Apenas um usuário pode utilizar todo o sistema de cada vez.

- › **Sistema Multi-usuário:**

- » Vários usuários podem acessar e utilizar o sistema de forma compartilhada ao mesmo tempo.

› Sistema Operacional de Rede:

- ›› Oferece acesso a recursos localizados em outros computadores da rede (ex: arquivos e impressoras).
- ›› Disponibiliza recursos locais para outros computadores de forma controlada.
- ›› A maioria dos sistemas operacionais atuais oferece esse tipo de funcionalidade.

› Sistema Operacional Distribuído:

- › Recursos de cada computador são disponíveis a todos na rede de forma transparente.
- › Usuários interagem com a aplicação sem saber onde ela está executando ou armazenando dados.

› Exemplos:

- › **Apache Hadoop:** Framework para processamento distribuído de grandes volumes de dados em ambientes de cluster.
- › **Sistemas de Arquivos Distribuídos:** Como Google File System (GFS) e Hadoop Distributed File System (HDFS), que guardam dados distribuídos e permitem acesso transparente.

› Servidor:

- » Gestão eficiente de grandes quantidades de recursos (disco, memória, processadores).
- » Imposição de prioridades e limites no uso dos recursos pelos usuários e aplicativos.
- » Podem ser distribuídos, de rede e multiusuários.
- » É comum não contar com interface gráfica.

› **Desktop:**

- ›› Atendimento ao usuário doméstico e corporativo para atividades cotidianas.
- ›› Geralmente tem interface gráfica.
- ›› Suporte à interatividade e operação em rede.

› Móvel:

- » Usado em dispositivos pessoais compactos como smartphones e tablets.
- » Geralmente sua gestão de energia é eficiente (bateria).
- » Conectividade com diversos tipos de redes (Wi-Fi, GSM, Bluetooth, NFC, etc.).
- » Interação com sensores variados (GPS, giroscópio, luminosidade, tela de toque, leitor de digitais, etc.).
- » **Exemplos:** Android, iOS.

› Embarcados:

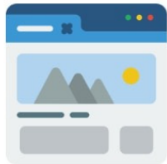
- ›› Sistemas operacionais embarcados são projetados para operar em hardware com recursos limitados de processamento, armazenamento e energia.
- ›› Otimização para hardware com capacidades reduzidas.
- ›› Foco em aplicações específicas e funções dedicadas.
- ›› Muitas vezes aparecem como bibliotecas ligadas ao programa durante a compilação.

› Exemplos de aplicação:

- ›› Automação e Controle: Sistemas de controle automotivo.
- ›› Equipamentos Eletrônicos: Leitores de DVD, TVs, fornos de microondas, centrais de alarme, impressora, entre outros.

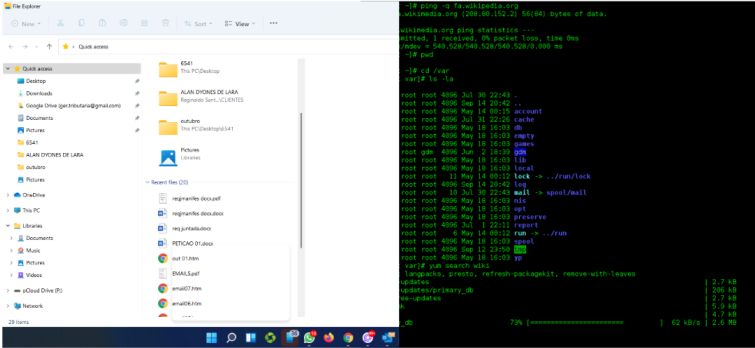
GUI vs CLI

- **GUI (Graphical User Interface):** Interface que permite ao usuário interagir com o sistema por meio de elementos gráficos como janelas, ícones e botões.
- **CLI (Command-line Interface):** Interface que permite ao usuário interagir com o sistema digitando comandos em texto por meio de um console ou terminal.



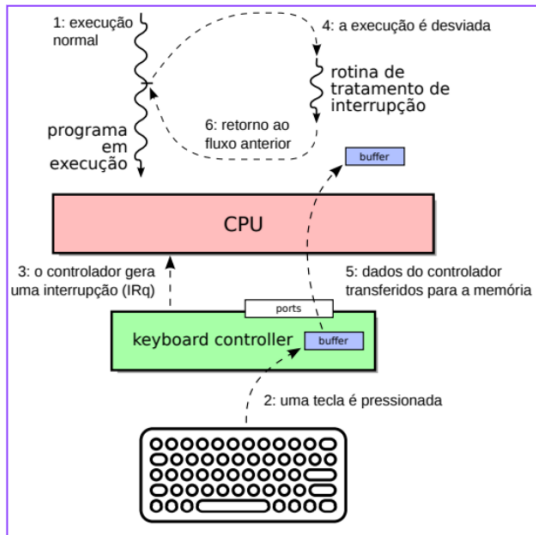
VS.







Gerenciamento de Recursos



- 1 O processador está executando um processo qualquer (em outras palavras, um fluxo de execução);
- 2 Um pacote vindo da rede é recebido pela placa Ethernet;
- 3 O controlador Ethernet envia uma solicitação de interrupção (IRQ) ao processador;

- 4 O processamento é desviado do programa em execução para a rotina de tratamento da interrupção;
- 5 A rotina de tratamento é executada para interagir com o controlador de rede (via barramentos de dados e de endereços) para transferir os dados do pacote de rede do controlador para a memória;
- 6 A rotina de tratamento da interrupção é finalizada e o processador retorna à execução do programa que havia sido interrompido.

```
~ vmstat 1
```

```
~ cat /proc/interrupts
```

- **Nível Núcleo:** Também chamado de nível supervisor ou kernel space, onde todas as funcionalidades do processador estão acessíveis e o processador opera durante a inicialização do computador.
- **Nível Usuário:** Também conhecido como userspace, permite apenas um subconjunto das instruções do processador e registradores, proibindo instruções "perigosas" e gerando erros se tentadas.

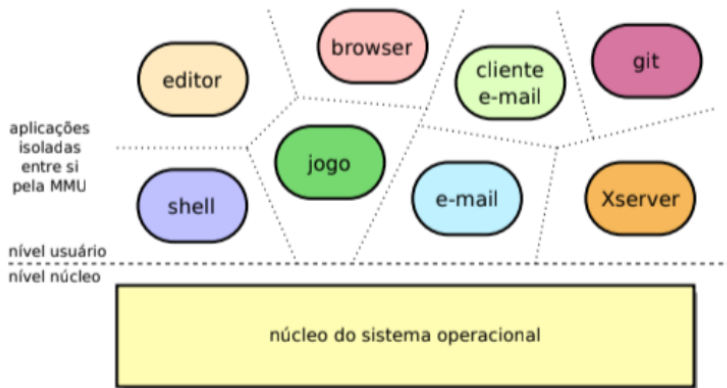
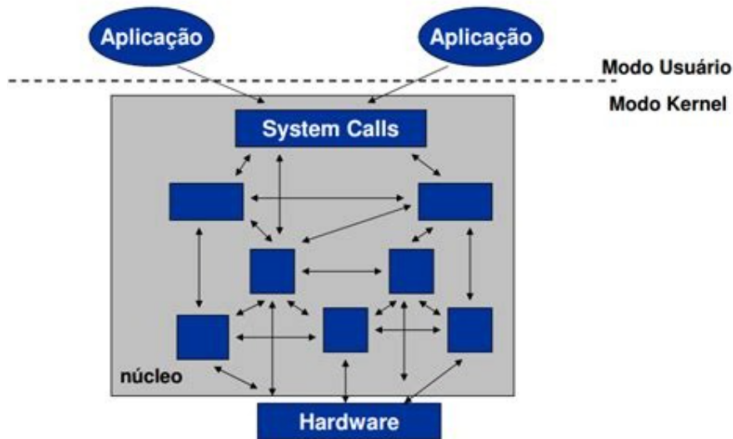
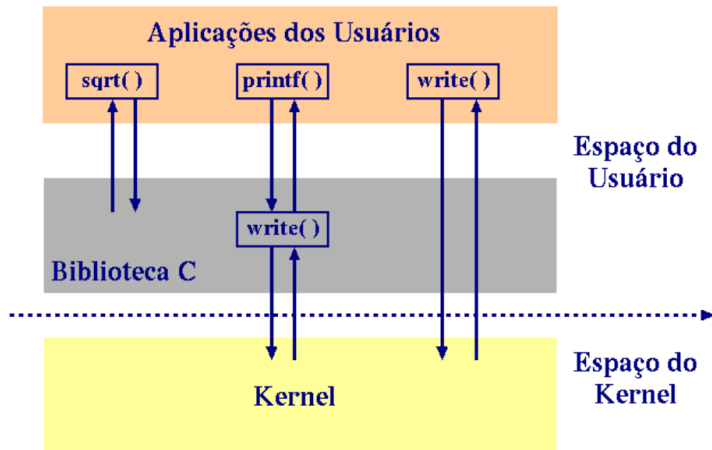


Figura 8: Níveis de Privilégio (Maziero, 2019)

- › As chamadas de sistema compõem a interface que o SO oferece às aplicações.
- › Executam no contexto do SO, usando o modo privilegiado do processador.
- › System calls (syscall) podem ser vistas como uma forma de API (Interface de Programação de Aplicações) no contexto dos sistemas operacionais.



CC-BY-SA 4.0



Estrutura

- **Núcleo:** é o coração do sistema operacional, portanto, gerencia os recursos de hardware usados pelas aplicações e implementa as principais abstrações para aplicativos e programas utilitários.
- **Código de inicialização (boot code):** envolve reconhecer, testar e configurar dispositivos de hardware, além de carregar e iniciar o núcleo do sistema operacional.
- **Drivers:** são módulos de código específicos para acessar dispositivos físicos, como discos rígidos SATA, portas USB e placas gráficas. Geralmente, são fornecidos pelos fabricantes em forma compilada.
- **Programas utilitários:** são ferramentas que complementam o núcleo do sistema, facilitam tarefas como formatação de discos, configuração de dispositivos, manipulação de arquivos, e fornecem interfaces gráficas e terminais.

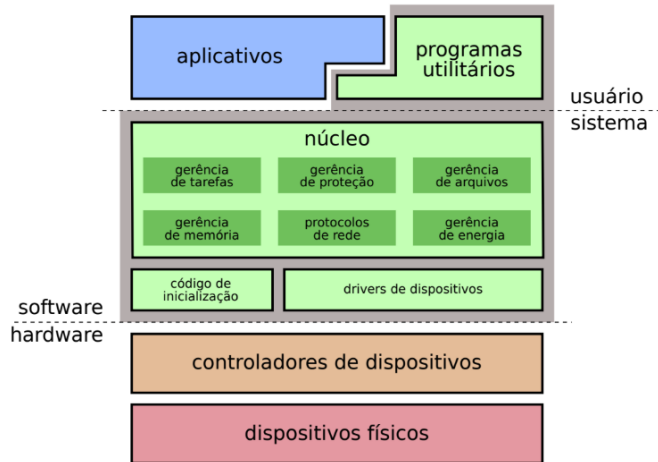
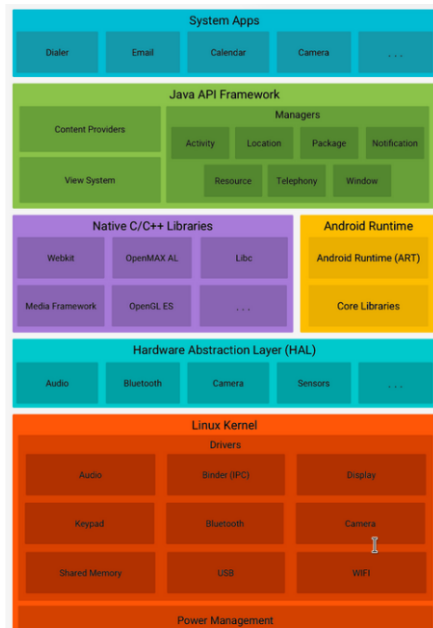
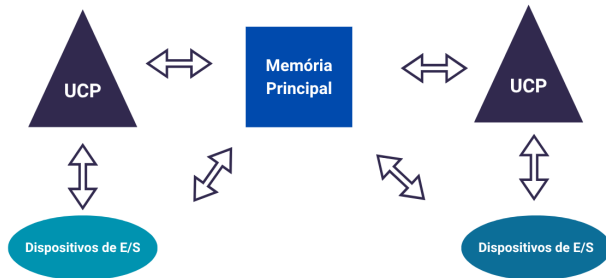


Figura 9: Estrutura básica (Maziero, 2019).



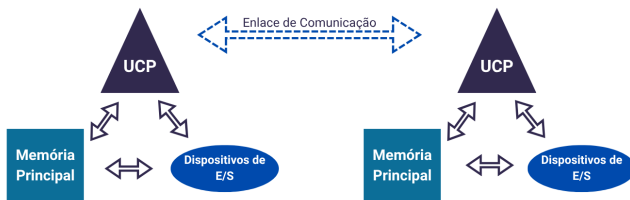
› Sistemas fortemente acoplados:

- ›› **Simétricos:** O SO gerencia.
- ›› **Assimétricos:** Uma UCP (primária) gerencia as demais (secundárias).



› Sistemas fracamente acoplados:

- › Vários computadores com controle independente sobre seus próprios recursos (UCP, MP e E/S) interagem entre si para executar as funções do sistema operacional ou de uma aplicação.



Arquitetura

- O sistema operacional é um "bloco maciço" de código que opera em modo núcleo, com acesso irrestrito ao hardware e memória.
- Componentes internos interagem diretamente, sem barreiras de acesso.
- Acesso direto entre componentes e dispositivos periféricos, resultando em sistemas mais rápidos e compactos.
- Erros em um componente podem afetar todo o sistema, levando o sistema ao colapso (travamento, reinicialização ou funcionamento errático).
- Alta interdependência entre componentes dificulta a manutenção e evolução, pois alterações em um componente podem impactar outros de forma inesperada.
- Sistemas antigos como UNIX e MS-DOS usavam arquitetura monolítica.
- O núcleo do Linux e o FreeBSD são monolíticos, com o Linux se tornando mais modular desde a versão 2.0.

Nota:

 Clique Aqui: Kernel Linux

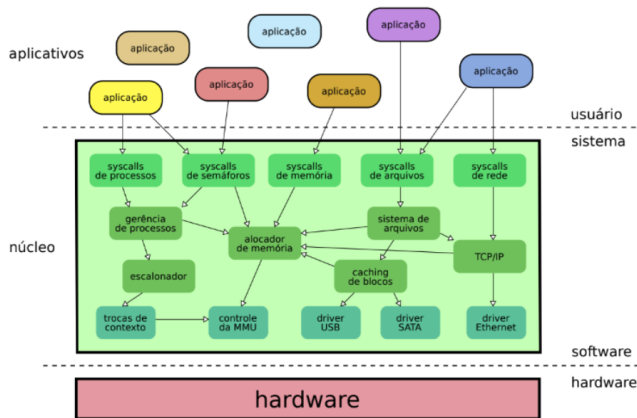


Figura 11: Sistemas Monolíticos (Maziero, 2019)

- **Camada Inferior:** Realiza a interface com o hardware.
- **Camadas Intermediárias:** Fornecem níveis de abstração e gerenciamento mais sofisticados.
- **Camada Superior:** Define a interface do núcleo para as aplicações (chamadas de sistema).
- **Privilégios:** A camada inferior tem acesso total ao hardware, enquanto a camada superior tem acesso mais restrito.
- **Modelo OSI:** Amplamente adotado no domínio das redes de computadores.
- **Desempenho:** O empilhamento de várias camadas pode aumentar o tempo necessário para um pedido chegar ao dispositivo ou recurso, prejudicando o desempenho.
- **Divisão de Funcionalidades:** Nem sempre é claro como dividir funcionalidades interdependentes em camadas.

› Sistemas em Camadas:

›› Exemplo:

- Camada HAL (Hardware Abstraction Layer): Implementada no Windows NT e seus sucessores para abstração de hardware (Windows 10).
- Minix 3, Windows 2000 e Android mostram uma organização parcial em camadas.

› Definição e Estrutura:

- » Retira do núcleo todo o código de alto nível relacionado a abstrações de recursos.
- » O núcleo contém apenas o código de baixo nível necessário para interagir com o hardware e criar abstrações básicas.
- » Código adicional é transferido para programas separados no espaço de usuário, chamados de serviços.
- » Essa abordagem é chamada de micronúcleo (ou [micro-kernel]).
- » Cada serviço pode ser desenvolvido de forma independente.
- » Serviços podem ser carregados e desativados conforme a necessidade.
- » Falhas em um serviço afetam somente esse serviço, devido ao confinamento de memória entre os serviços.
- » Implementa a noção de tarefa, espaços de memória protegidos para cada aplicação, comunicação entre tarefas e operações de acesso às portas de entrada/saída.
- » Uso do processador e da memória, sistema de arquivos, controle de acesso e drivers são implementados fora do núcleo.

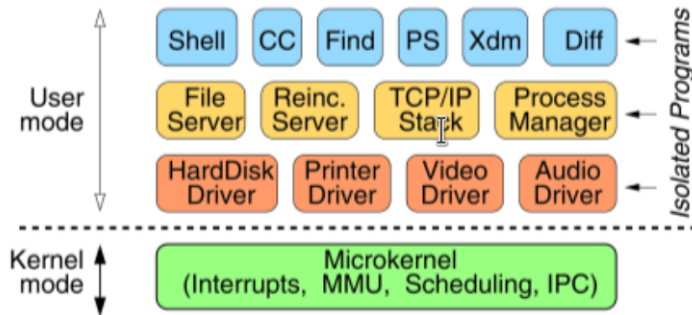


Figura 12: Minix 3 (Maziero, 2019)

- Os núcleos híbridos visam combinar a flexibilidade dos micronúcleos com o desempenho dos núcleos monolíticos, proporcionando uma solução equilibrada.
- **Sistemas de híbridos:**
 - » Micronúcleos oferecem boa modularidade e robustez.
 - » Camadas: não tiveram o sucesso esperado devido ao baixo desempenho.
 - » Abordagem Intermediária: Combina aspectos dos núcleos monolíticos e micronúcleos.
 - » Influência da Arquitetura em Camadas: Núcleos híbridos frequentemente incorporam características de camadas.
 - » Windows NT iniciou com um núcleo semelhante a um micronúcleo.
 - » Apple MacOS e iOS:
 - Núcleo XNU: Uma combinação dos núcleos Mach (micronúcleo) e FreeBSD (monolítico).

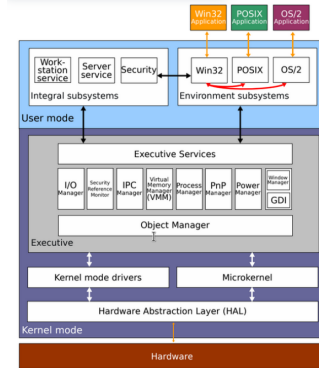


Figura 13: Windows 2000 (Maziero, 2019)

Capítulo 1 (OLIVEIRA; CARISSIMI; TOSCANI, 2001)



Bibliografia



OLIVEIRA, R. S.; CARISSIMI, A. S.; TOSCANI, S. S. **Sistemas Operacionais**. 2. ed. Porto Alegre: Sagra-Luzzatto, 2001.



STALLINGS, William. **Operating Systems: Internals and Design Principles**. 6. ed. Upper Saddle River, NJ: Prentice-Hall, 2009.



TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2010.



SILBERSCHATZ, Abraham; GALVIN, Peter; GAGNE, Greg. **Sistemas Operacionais: Conceitos e Aplicações**. Rio de Janeiro: LTC, 2001.

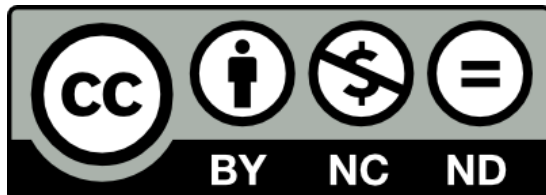


TANENBAUM, Andrew S.; WOODHULL, Albert S. **Sistemas Operacionais: Projeto e Implementação**. 2. ed. Porto Alegre: Bookman, 2000.



MAZIERO, Carlos Alberto. **Sistemas Operacionais: Conceitos e Mecanismos [recurso eletrônico]**. Curitiba: DINF - UFPR, 2019. ISBN 978-85-7335-340-2.

Estes slides estão protegidos por uma licença Creative Commons



Este modelo foi adaptado de Maxime Chupin.

Sistemas Operacionais

Introdução