Sistemas Operacionais: Introdução

Prof. Dr. Rafael Lopes Gomes
Universidade Estadual do Ceará (UECE)

Agenda

- Definição de SO
- Conceitos sobre SO
- Chamadas de Sistema
- Estrutura de SO

Computadores Modernos

- Um ou mais processadores
- Memória principal
- Disco
- Dispositivos I/O
  - Impressora
  - Teclado
  - Interface de Rede
  - Etc
- Outros

Motivação

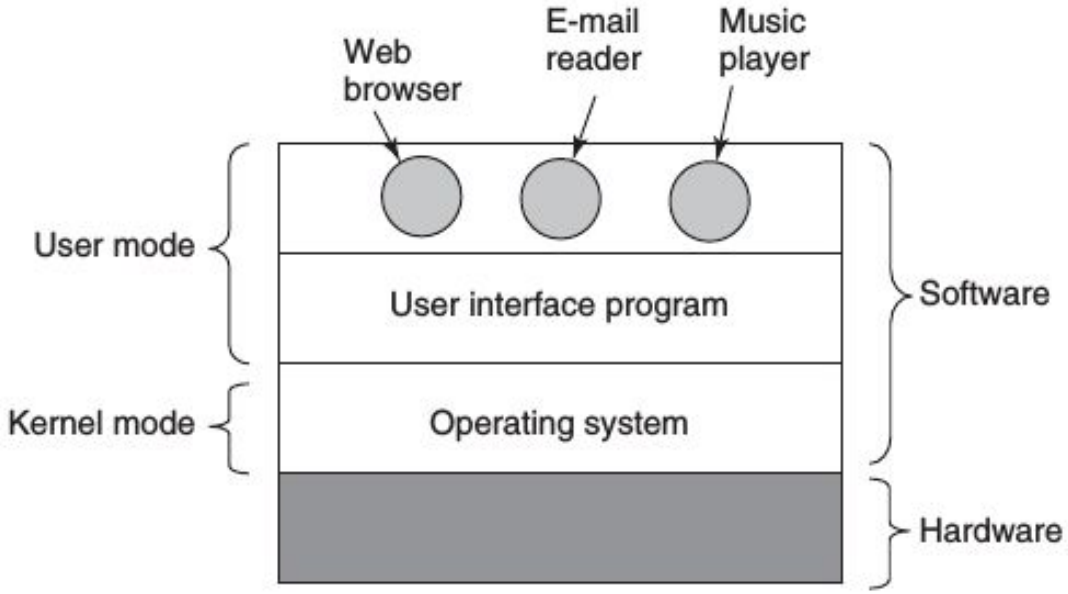
- Conhecer os detalhes de baixo nível dos computadores é incomum.
- Se todo o programador tivesse que compreender como todas as partes do computadores funcionam especificamente, nenhum código seria feito.
- Gerenciar todos os componentes de um computador e otimizar seu uso é um grande desafio.

Sistema Operacional

- Definição: Software que fornece aos programas do usuário um modelo do computador simplificado, bem como gerencia os recursos do computador.
- Exemplos:
  - Linux
  - Windows
  - FreeBSD
  - Etc

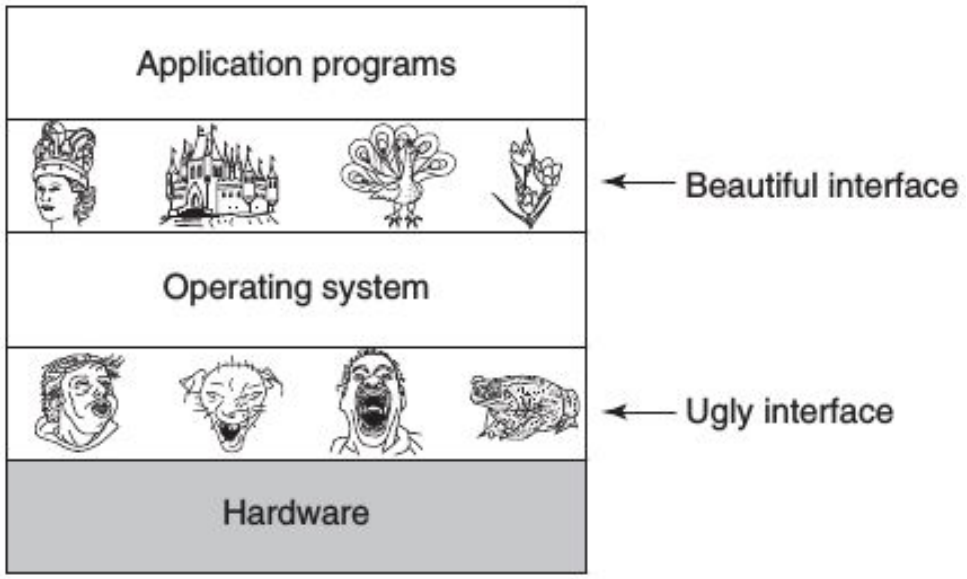
Modos de Operação

- A maioria dos computadores possui dois modos de operação:
  - Núcleo (Kernel): tem acesso completo a todo o HW e pode executar qualquer instrução existente.
  - Usuário (User): tem acesso a somente um subconjunto de instruções (excluindo as de controle e de E/S).



Funções de um SO

- De maneira geral os SOs realizam duas funções essenciais:
  - Fornecer aos programas (e programadores) um conjunto de recursos abstratos.
  - Gerenciar os recursos de HW (memória,CPU, E/S, etc).



Conceitos de SO

- Processos
- Espaço de Endereçamento
- Arquivos
- Entrada/Saída
- Proteção

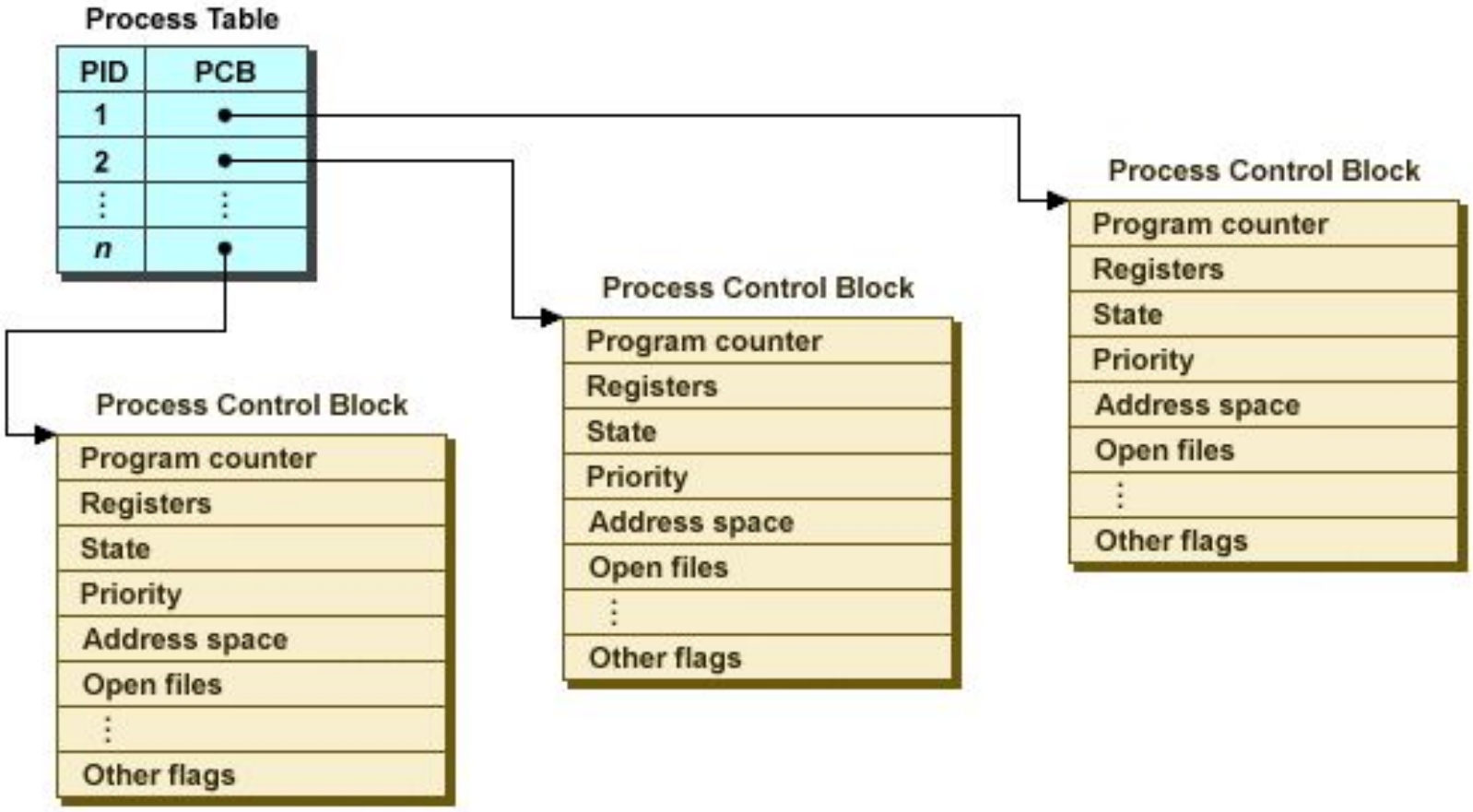
Processos

- Um **processo** é, de forma geral, um programa em execução.
- Cada processo é associado a um espaço de endereçamento (onde o processo pode ler e escrever).
- Além disso, o processo está associado a um conjunto de recursos:
  - Registradores (contador e ponteiro de pilha)
  - Lista de arquivos abertos
  - Alarmes pendentes
  - Processos relacionados
  - Outros

Processos

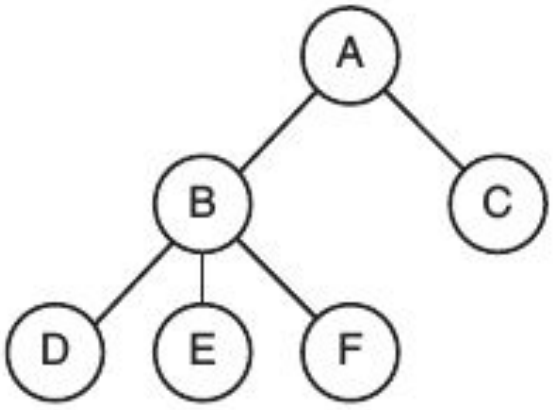
- Em geral, as informações básicas de cada processo ficam armazenadas na **Tabela de Processos** do SO.
- Cada entrada nesta tabela é chamada de **Bloco de Controle de Processo** (Process Control Block – PCB).
- A tabela é usada sempre que um processo é suspenso e precisa ser retomado posteriormente.
- Linux:
  - ps aux
  - top
  - htop

Processos


Process Table
PID
PCB
1
2
n
Process Control Block
Program counter
Registers
State
Priority
Address space
Open files
Other flags
Process Control Block
Program counter
Registers
State
Priority
Address space
Open files
Other flags
Process Control Block
Program counter
Registers
State
Priority
Address space
Open files
Other flags

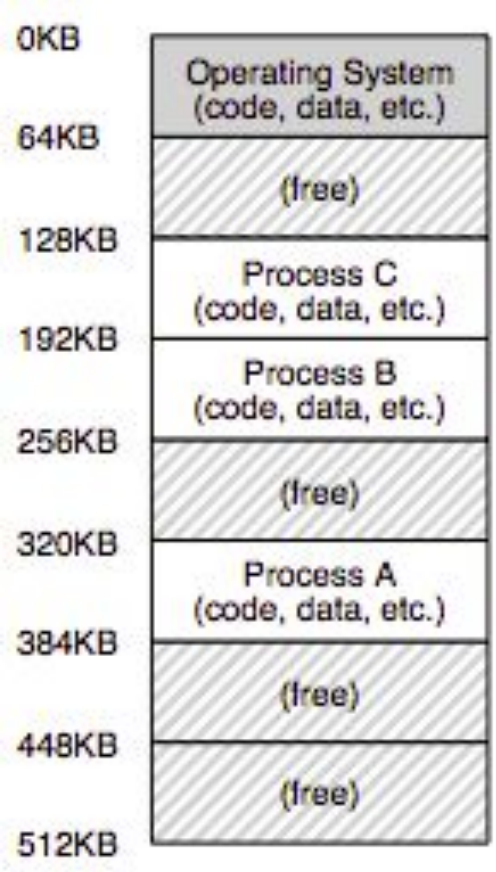
Processos

- Um processo pode criar um ou mais processos filhos, que por sua vez podem criar outros processos filhos, e assim por diante (via chamadas de sistema).
- Hierarquia de processos em formato de árvore.
- Em muitos casos os processos precisam comunicar-se entre si e sincronizar suas atividades.



Espaço de Endereçamento

- SOs modernos permitem que diversos programas estejam na memória simultaneamente.
- A fim de evitar que estes programas interfiram entre si é necessário algum mecanismo de proteção.
- O mecanismo de proteção fica no HW, mas é controlado pelo SO.
- Cada processo possui algum conjunto endereços que ele pode usar.
- Linux
  - ps aux | grep NOME
  - cat /proc/PID/status
  - pmap -x PID

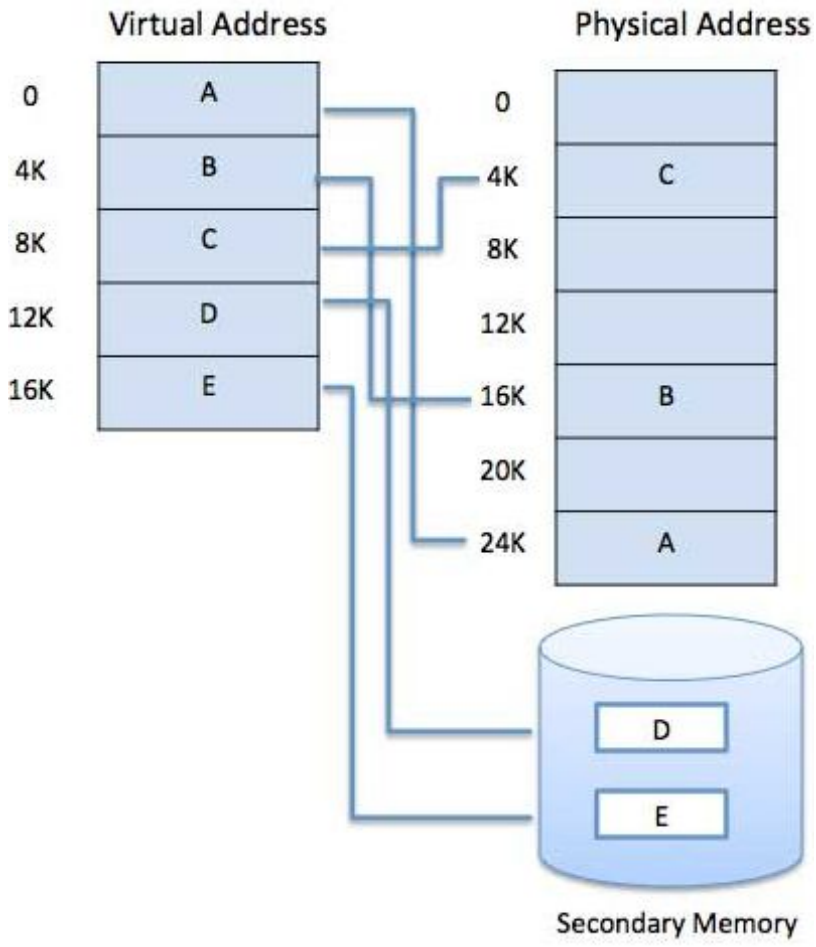


Espaço de Endereçamento

- Processos podem requisitar mais memória via chamadas de sistema.
- O que ocorre quando um processo precisa de mais memória que o disponível ?

Memória Virtual

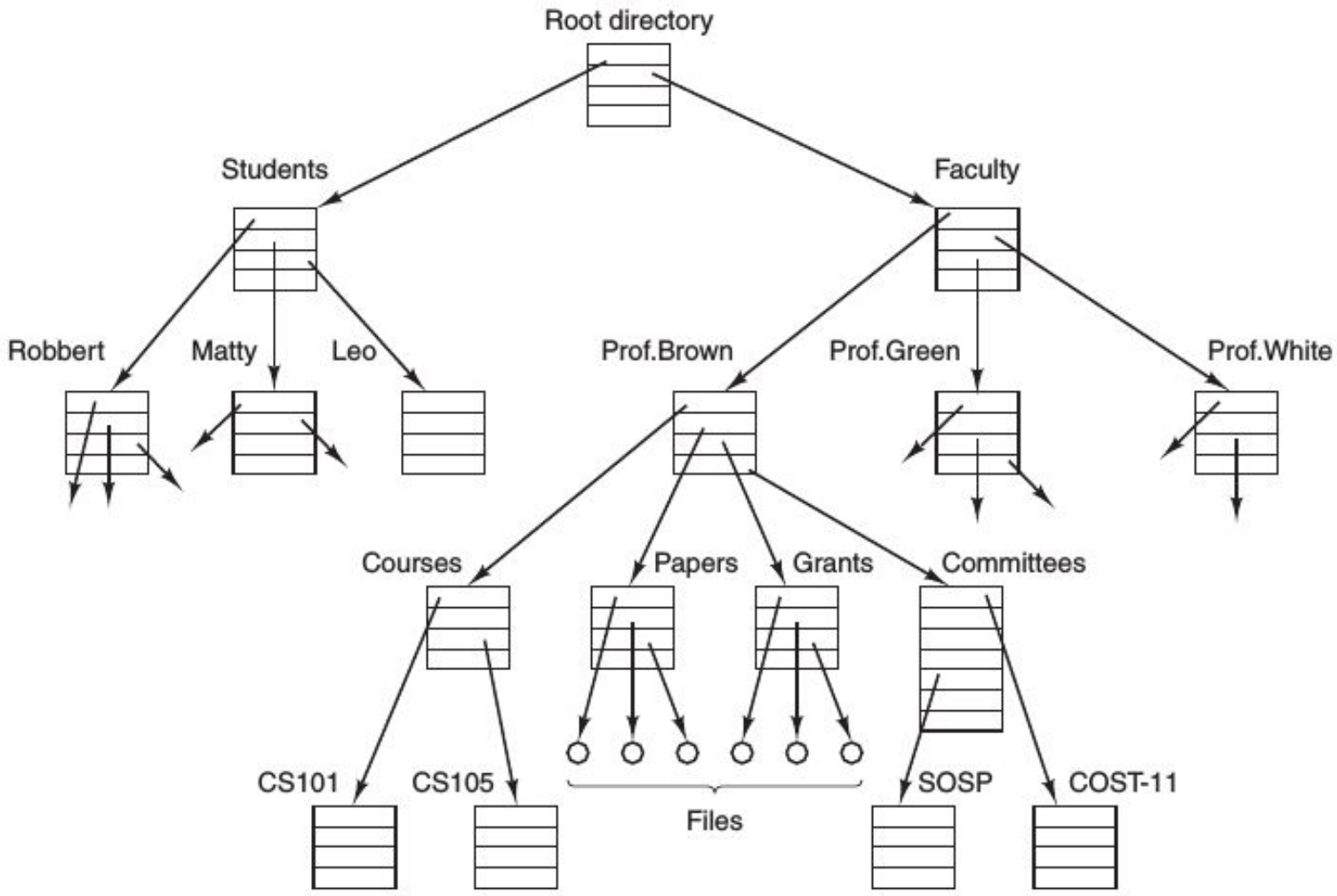
- O sistema operacional mantém parte do espaço de endereçamento na memória principal e parte no disco.
- São enviados trechos conforme a necessidade.
- O SO cria uma abstração de um espaço de endereçamento como o conjunto de endereços que um processo pode usar.



Arquivos

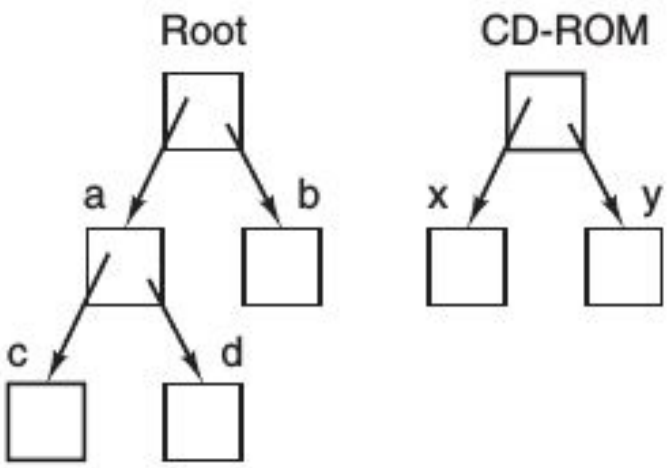
- Umas das funções do SO é esconder as peculiaridades dos discos e outros dispositivos de E/S.
- Arquivos: abstração de conteúdo dos dispositivos.
- Chamadas de sistemas são necessárias para criar, ler e escrever arquivos.
- Diretório: agrupamento de arquivos.
  - Nome de caminho
  - Diretório raiz
  - Diretório de trabalho

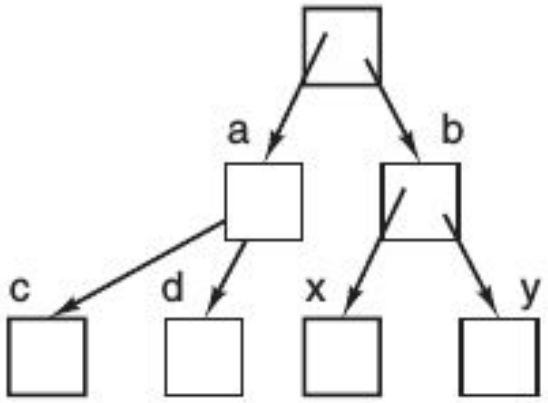
Arquivos


Root directory
Students
Faculty
Robbert
Matty
Leo
Prof.Brown
Prof.Green
Prof.White
Courses
Papers
Grants
Committees
CS101
CS105
SOSP
COST-11
Files

Arquivos

- Outros diretórios podem ser montados em diretórios existentes, estendendo a hierarquia atual.
- Arquivos e diretórios tem permissões, as quais são incluídas no Descritor de arquivos.



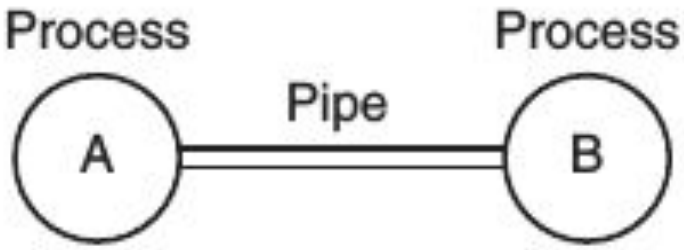


Arquivos

- Arquivos especiais: permitem que dispositivos de E/S sejam tratados como arquivos.
- Podem ser lidos e escritos com as mesmas chamadas de sistema de arquivos comuns.
  - De bloco: modelam dispositivos que consistem de uma coleção de blocos aleatoriamente endereçáveis, podem assim acessar diretamente blocos existentes independente da estrutura do sistema. Ex: discos.
  - De caracteres: modelam dispositivos que enviam fluxos de caracteres. Ex: impressora, modem, etc.

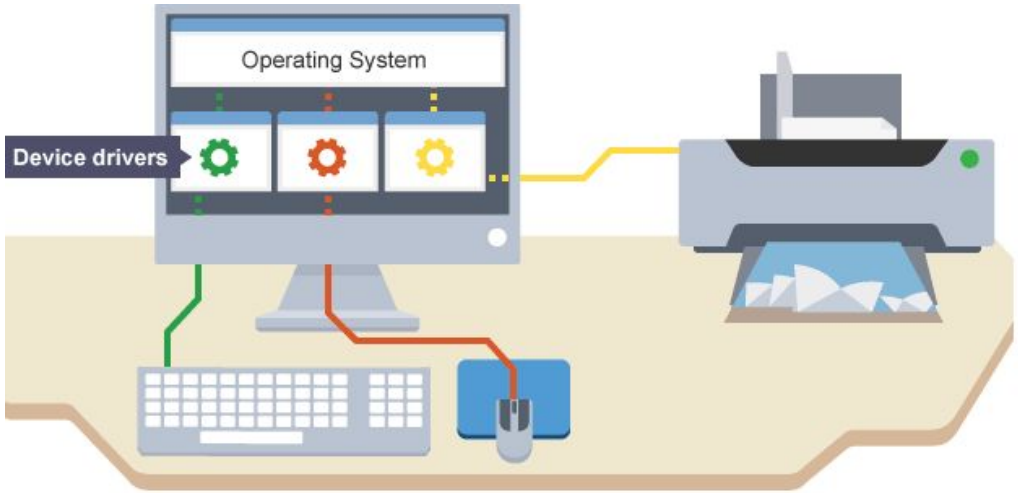
Arquivos

- Pipe é um pseudo arquivo que pode conectar dois processos.
- Precisa ser configurado previamente.
- Escreve os dados no pipe como se fosse um arquivo de saída.
- Lê os dados do pipe como se fosse um arquivo de entrada.



Entrada e Saída (E/S)

- Todo o SO tem um subsistema de E/S para gerenciar os dispositivos.
- Alguns dos software aplicados são independentes do dispositivo, i.e., genéricos. Outros são específicos a dispositivos de E/S particulares, os drivers.



Proteção

- Cabe ao SO gerenciar a segurança do sistema de maneira que os recursos sejam acessíveis somente por usuários autorizados.
- Exemplos:
  - Permissões
  - Detecção de intrusos
  - Ocultação

Chamadas de Sistema

- Chamadas de sistema são a interface entre os programas de usuário e o sistema operacional , sendo fundamentalmente abstrações .
- As chamadas de sistema variam entre os SOs, mas basicamente tem-se as seguintes funções:
  - Gerenciamento de processo
  - Gerenciamento de Arquivos
  - Gerenciamento do sistema de diretórios
  - Outras diversas

Chamadas de Sistema

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing, or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Chamadas de Sistema

Directory- and file-system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970

Chamadas de Sistema

- Chamadas de sistema são realizadas em uma série de passos.
- A seguir será mostrada a execução de uma chamada de sistema *read*

Contador = read(fd, buffer, nbytes)

- fd: especifica o arquivo
- Buffer: ponteiro para o espaço de memória
- Nbytes: quantidade de bytes a ser lido
- Contador: nº de bytes realmente lidos (zero = erro)

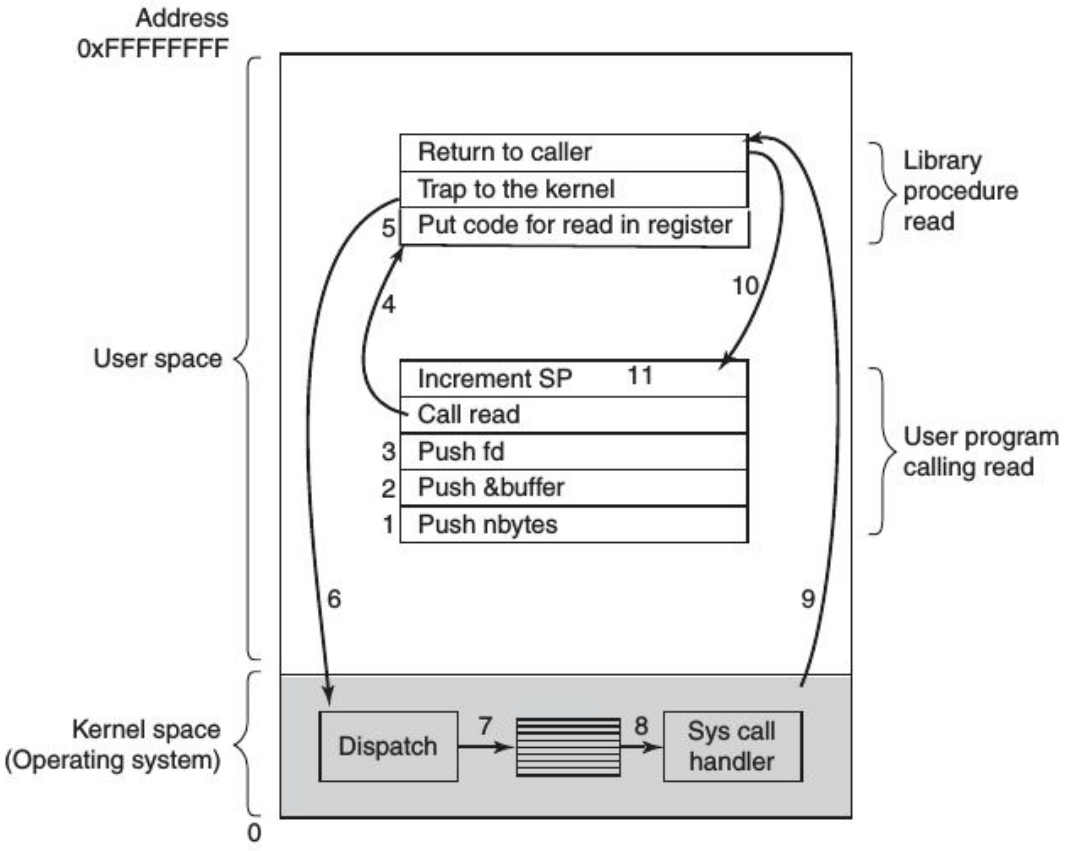
Chamadas de Sistema

1, 2 e 3 – Empilha os parâmetros;

4 – Chamada real para a rotina

5 – Coloca o número da chamada de sistema a espera (em registrador)

6 – Executa uma função TRAP para passar do modo usuário para o modo núcleo e começar a execução em um endereço dentro do núcleo.



Chamadas de Sistema

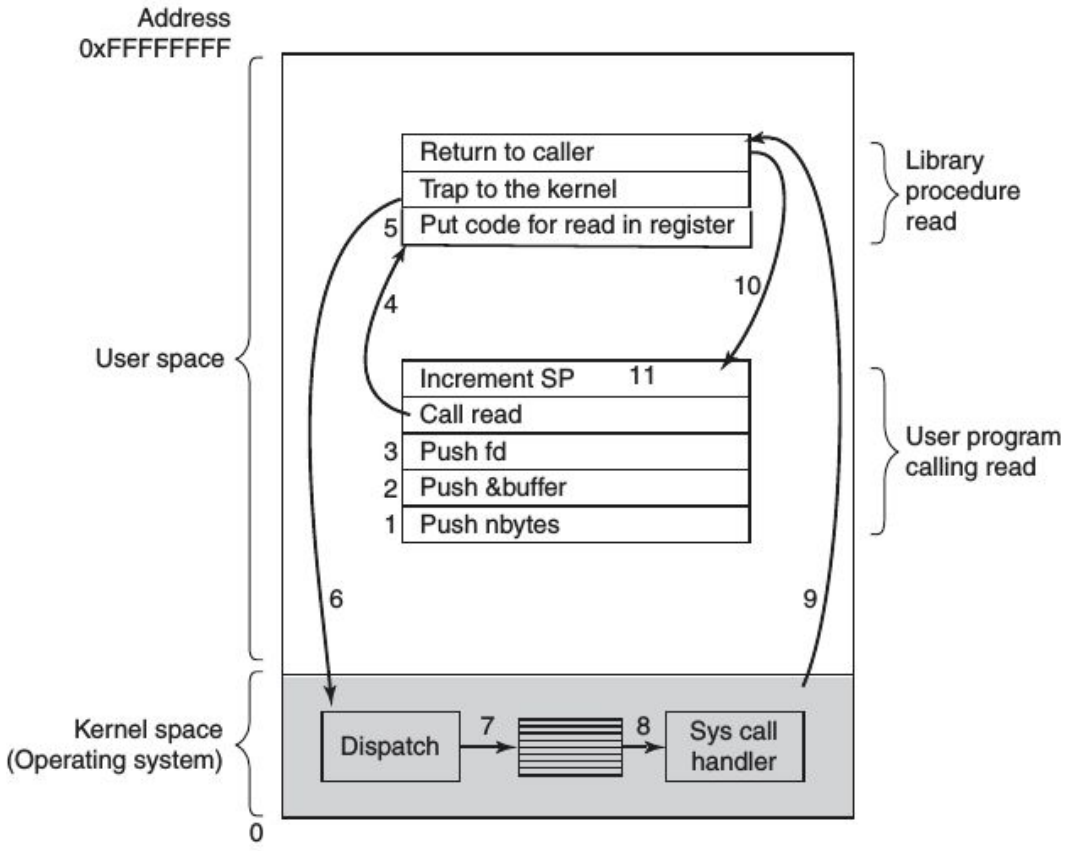
7 – Despacha para o tratador correto da chamada de sistema

8 – É executado o tratamento de chamada de sistema

9 – Tratamento finalizado, o controle é para a rotina da biblioteca em modo usuário (após o TRAP)

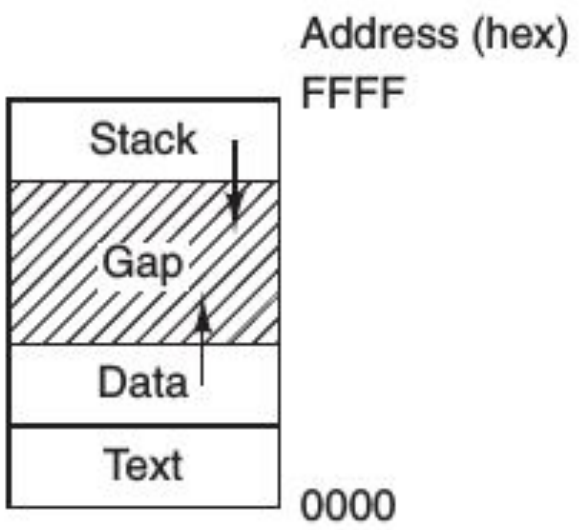
10 – Retorna ao programa que fez a chamada

11 - Termina a tarefa limpando a pilha.



Chamadas de Sistema

- Processos UNIX tem sua memória divida em três segmentos:
  - Texto: código
  - Dados: variáveis
  - Pilha
- A pilha cresce para cima e o de dados cresce para baixo (chamada de sistema brk).
- Linux:
  - strace df



Estrutura de SOs

- Sistemas Monolíticos
- Sistemas de Camadas
- Micronúcleos
- Sistemas Cliente-Servidor
- Máquinas Virtuais

Sistemas Monolíticos

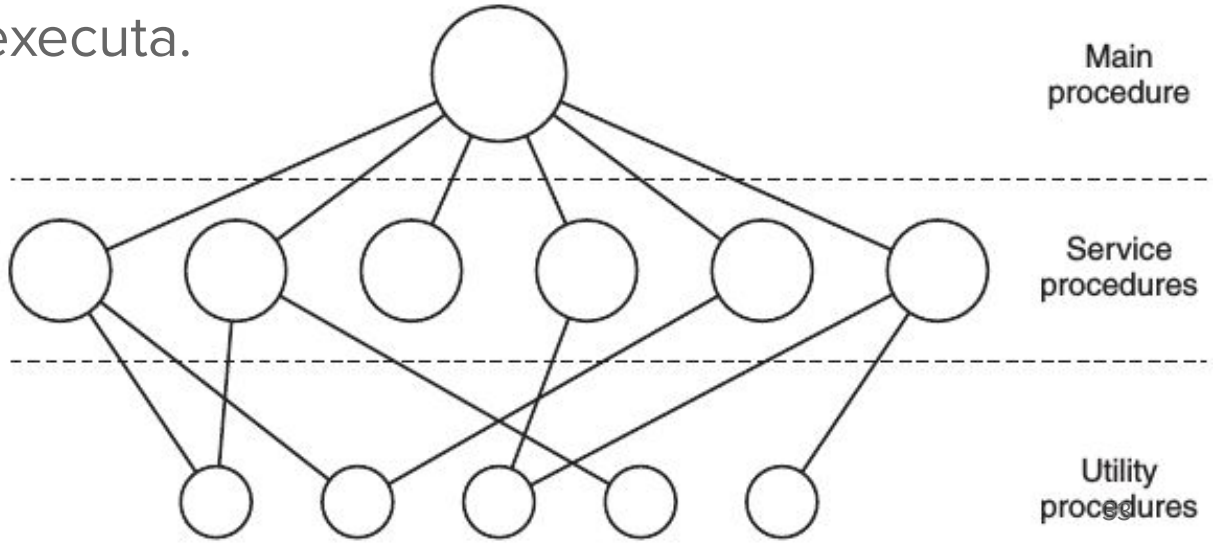
- O SO é executado como um único programa em modo núcleo.
- SO é uma coleção de rotinas, ligadas a um único binário executável.
- Um procedimento é capaz de chamar qualquer outro
  - Eficiência (+)
  - Complexidade (-)
  - Uma falha para o sistema (-)

Sistemas Monolíticos

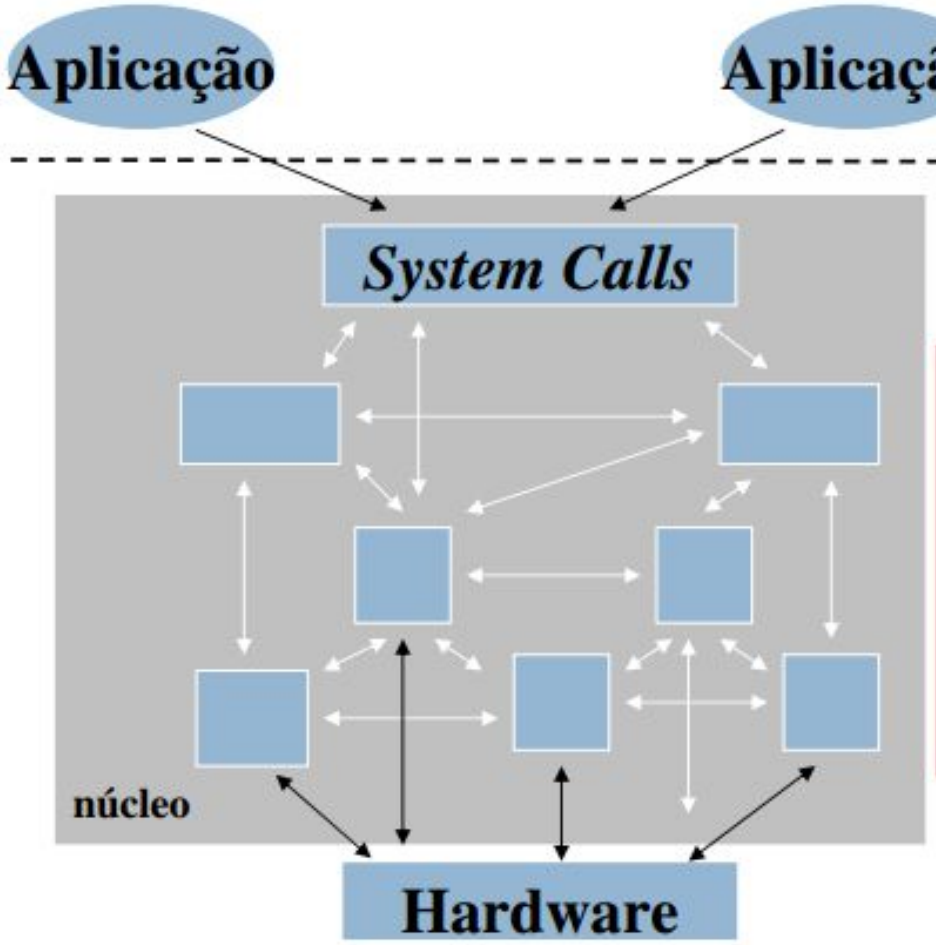
- Todas as rotinas são compiladas e agrupadas em um executável (ligador ou *linker*)
- Não há ocultação de informações
- Os serviços (chamadas de sistema) são requisitados colocando-se parâmetros em local bem definido (ex: pilha) e uma instrução *trap* é executada.

Sistemas Monolíticos

- Estrutura mínima:
  - Um programa principal que invoca a rotina de serviço requisitada;
  - Um conjunto de rotinas de serviço que executam as chamadas de sistema;
  - Um conjunto de rotinas utilitárias que ajudam as rotinas de serviço.
- Para cada chamada de sistema há uma rotina de serviço que se encarrega dela e a executa.



Sistemas Monolíticos



1. Um programa principal que invoca a rotina do serviço requerido.
2. Um conjunto de rotinas de serviço que executam as chamadas de sistema.
3. Um conjunto de rotinas de utilidade que auxiliam as rotinas de serviço.

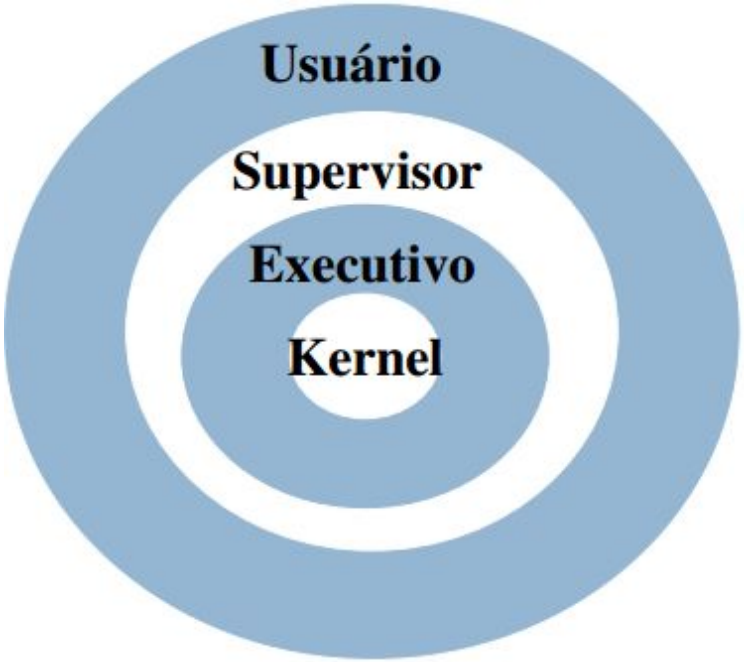
Sistemas de Camadas

- Forma uma hierarquia de camadas, cada uma construída (utiliza funções) sobre a camada abaixo.
- Cada camada oferece um conjunto de funções que pode ser usado por camadas adjacentes.
- Vantagens:
  - Isolamento: facilita alteração e depuração;
  - Hierarquia: proteção de funções.
- Desvantagens:
  - Desempenho.
- Exemplo: THE
  - Sistema de lote simples;
  - Define 6 camadas na estruturação.

Camada	Função
5	O operador
4	Programas de usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador–processo
1	Memória e gerenciamento de tambor
0	Alocação do processador e multiprogramação

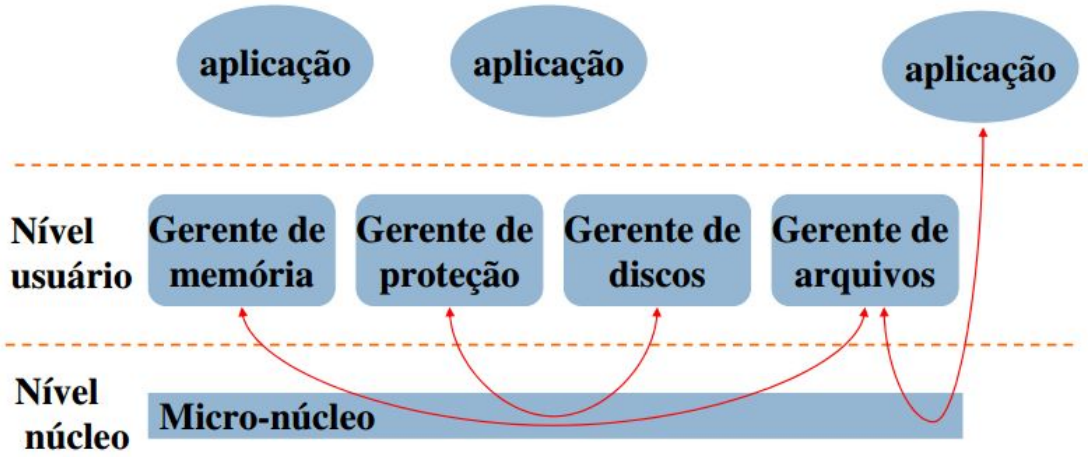
Sistemas de Camadas

- Variação de camadas: Anéis
  - Anéis mais internos são mais privilegiados que os externos;
  - Anéis externos executam chamadas de sistemas para usar os serviços de anéis mais internos;
  - Proteção;
  - Exemplo: Multics.



Microkernel

- Usualmente a maior parte das camadas é executada em modo núcleo;
- Qualquer erro na implementação das camadas pode comprometer o SO como um todo;
- Colocar o mínimo possível de funções em modo núcleo.
- Núcleo menor e simples: SO divido em vários processos.
  - Escalonamento;
  - Gerência de memória;
  - E drivers.

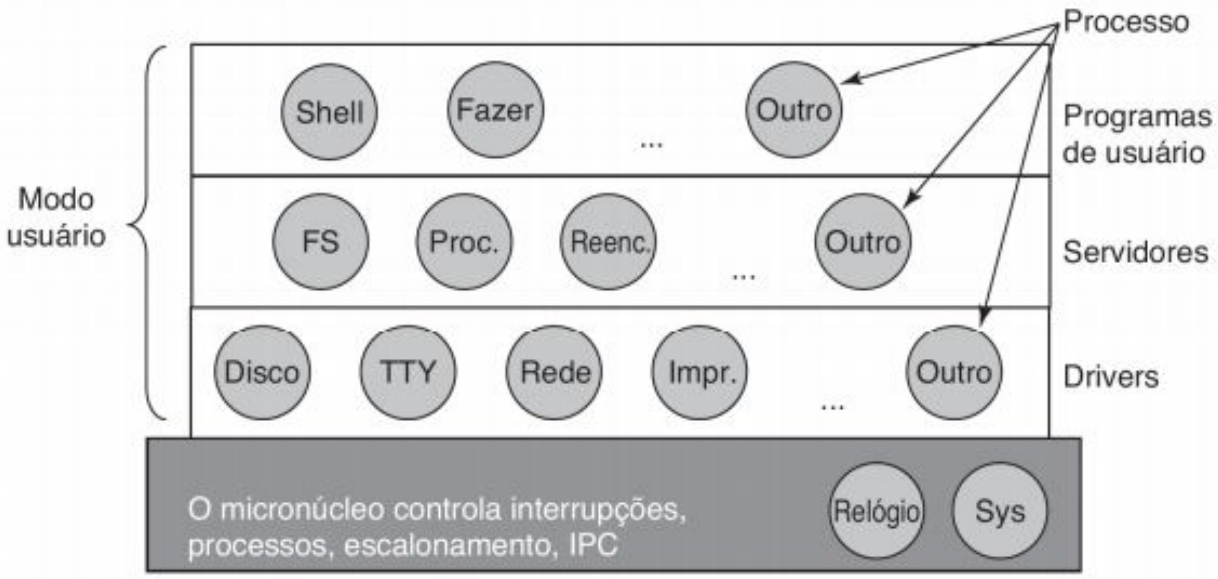


Microkernel

- Objetivo:
  - Alta confiabilidade (divisão do SO em módulos pequenos);
  - Apenas um módulo (micronúcleo) executa em modo núcleo ;
  - Demais partes são executadas como processos de usuário.
- Exemplo:
  - Execução de cada driver e de cada sistema de arquivo como processo separado.
  - Um erro pontual pode quebrar aquele componente, mas não o sistema todo.

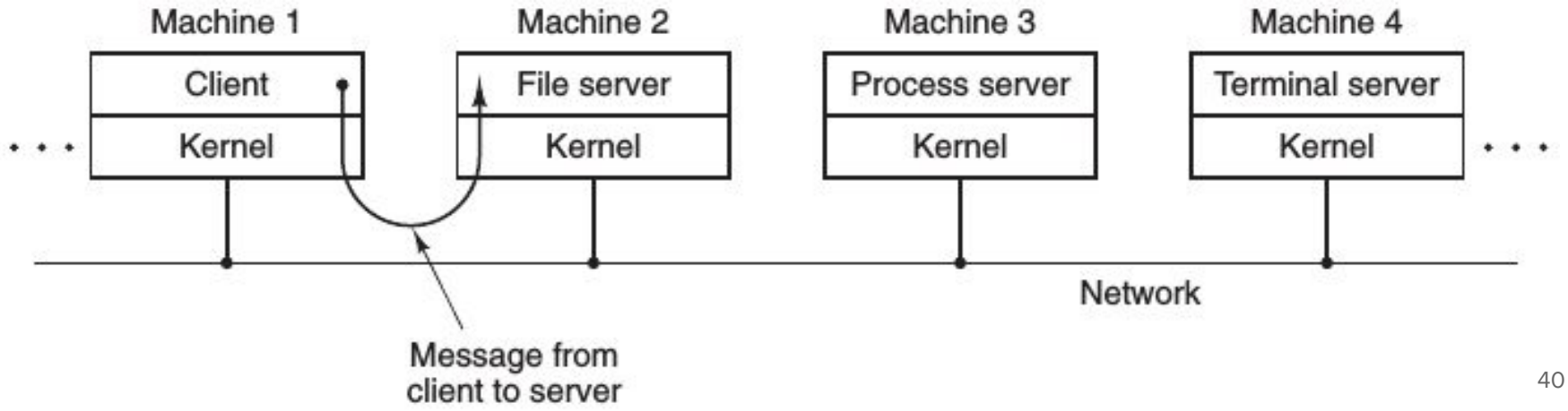
Microkernel

- Exemplo: MINIX
  - Fora do núcleo: 3 camadas de processo em modo usuário
  - Camada mais baixa contém os drivers
  - Não tem acesso direto: passa parâmetros e o núcleo faz as devidas verificações (autorização, validação, etc)

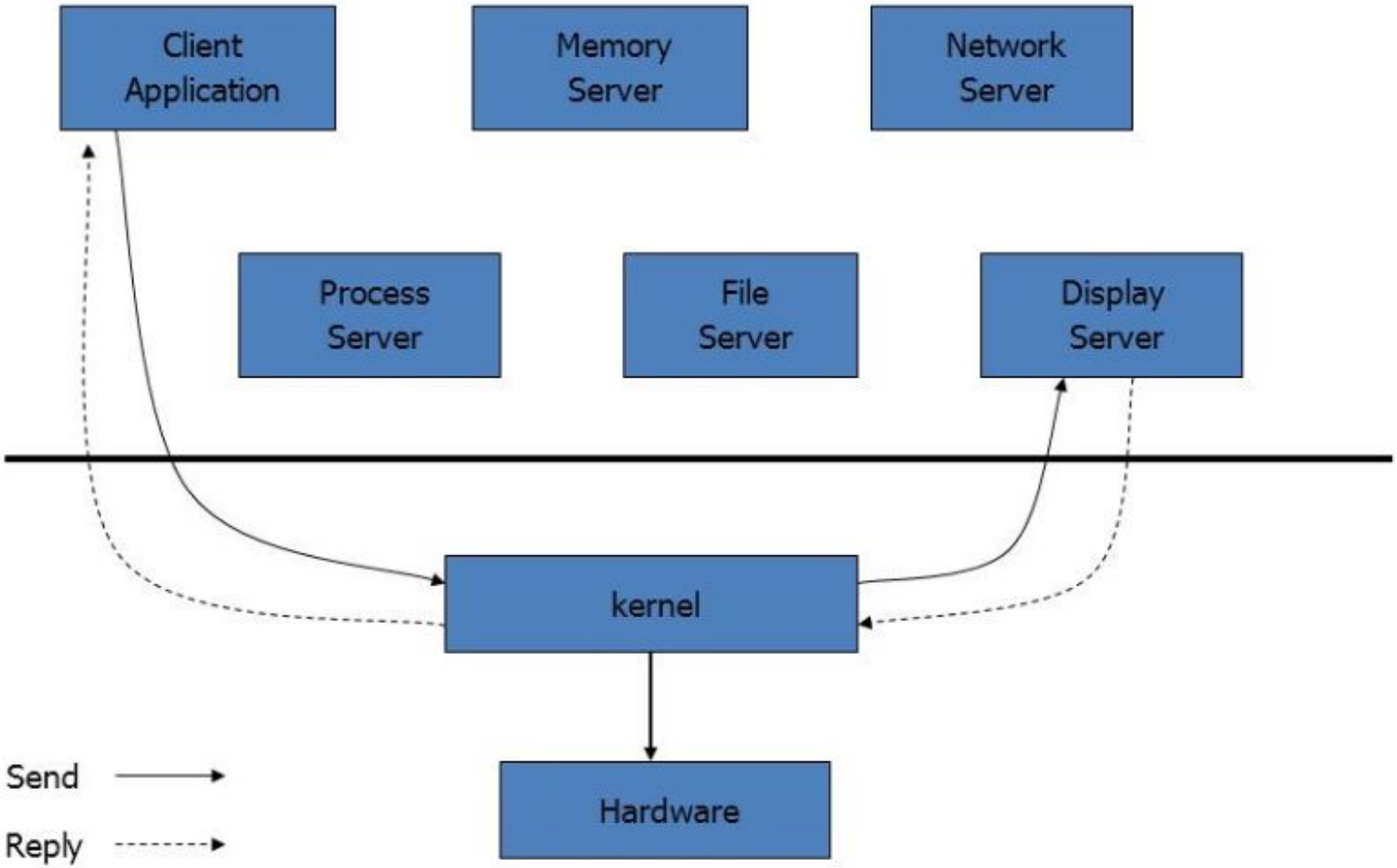


Modelo Cliente-Servidor

- Variação de microkernel
- Definição de duas classes de processo:
  - Cliente: requisita um serviço/procedimento;
  - Servidor: realizam o serviço/procedimento.
- O servidor pode ser tanto um processo usuário como o núcleo do sistema.

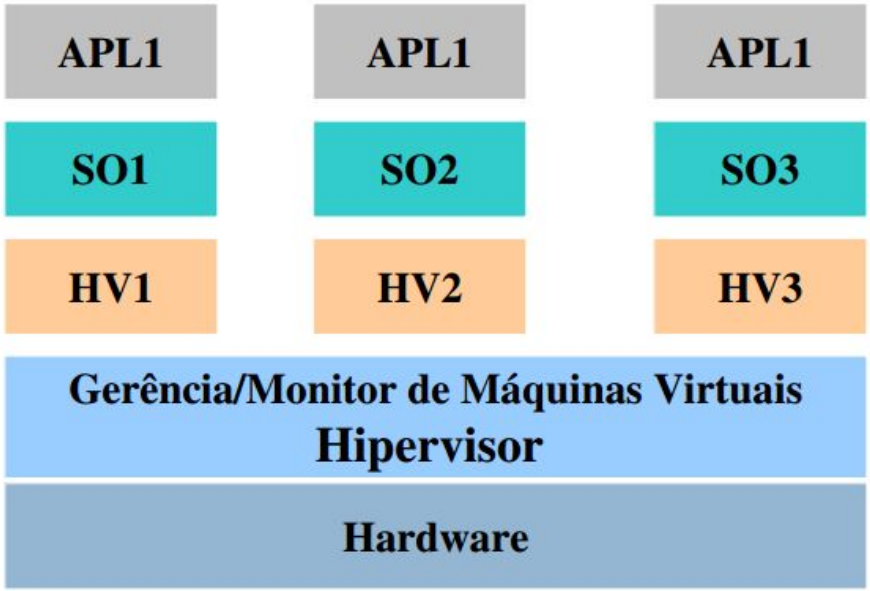


Modelo Cliente-Servidor



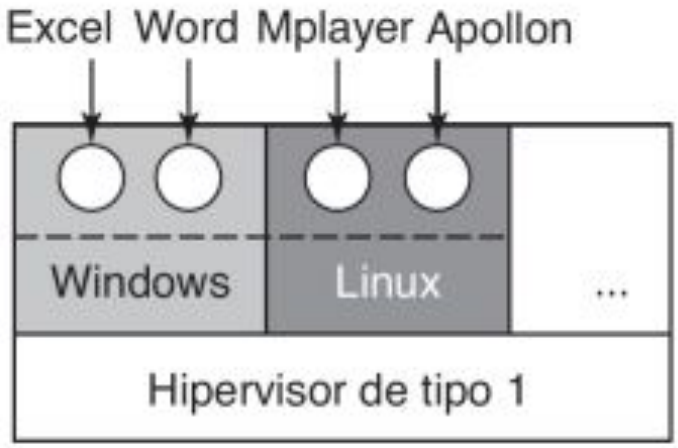
Máquina Virtual

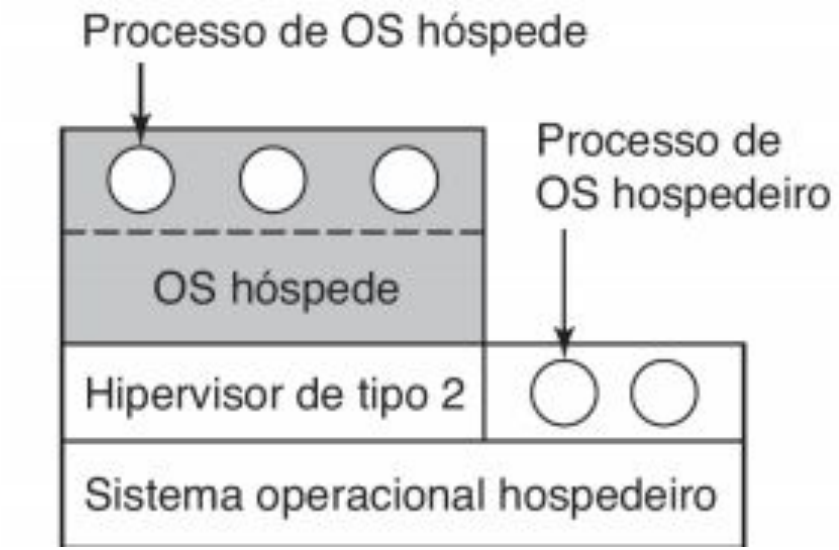
- São cópias lógicas do HW sobre o HW real
  - Modos de operação, E/S, interrupções, etc;
  - Não são máquinas estendidas.
- Monitor de Máquina Virtual ou Hypervisor:
  - Opera direto sobre o HW oferecendo multiprogramação
- Cada máquina virtual é idêntica ao HW original
- Cada uma pode executar o SO que quiser



Máquina Virtual

- Hipervisor
  - Tipo 1: executado diretamente sobre o HW;
  - Tipo 2: são executados como aplicativos na camada superior do SO.





Máquina Virtual

- Problema antigamente:
  - Quando um SO em uma VM (em modo usuário) executa uma instrução privilegiada
  - O HW precisa capturar e direcionar ao hipervisor
  - Queda de desempenho
- Melhoria de desempenho:
  - Acrescentar um módulo para fazer esse tratamento

Máquina Virtual

- Diferença entre tipo 1 e 2:
  - Tipo 2 usa um sistema operacional hospedeiro para criar processos, armazenar arquivos, etc;
  - Tipo 1 não tem suporte subjacente, ou seja, realiza todas as funções sozinho.

- Uma abordagem diferente é a paravirtualização
  - Modificação do SO
  - Gerenciamento de instruções.

FIM