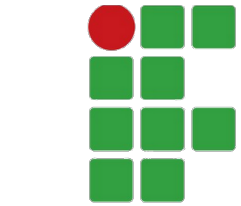
Sistemas Operacionais

Estrutura Sistemas Operacionais

Felipe Augusto Lima Reis

felipe.reis@ifmg.edu.br



INSTITUTO FEDERAL
Minas Gerais

Sumário

Introdução

Introdução

- Sistemas Operacionais podem ser analisados sob as seguintes óticas:
  - Serviços que o sistema fornece;
  - Interface que disponibiliza aos usuários e programadores;
  - Componentes e suas interconexões;
- Os serviços, as interfaces para usuários e as componentes são escolhidas durante o projeto de um SO, com base no objetivo dos mesmos [Silberschatz et al., 2012];
- Nesta seção serão estudados os recursos mais comuns e algumas formas de organização dos núcleos dos SOs.

Serviços dos Sistemas Operacionais

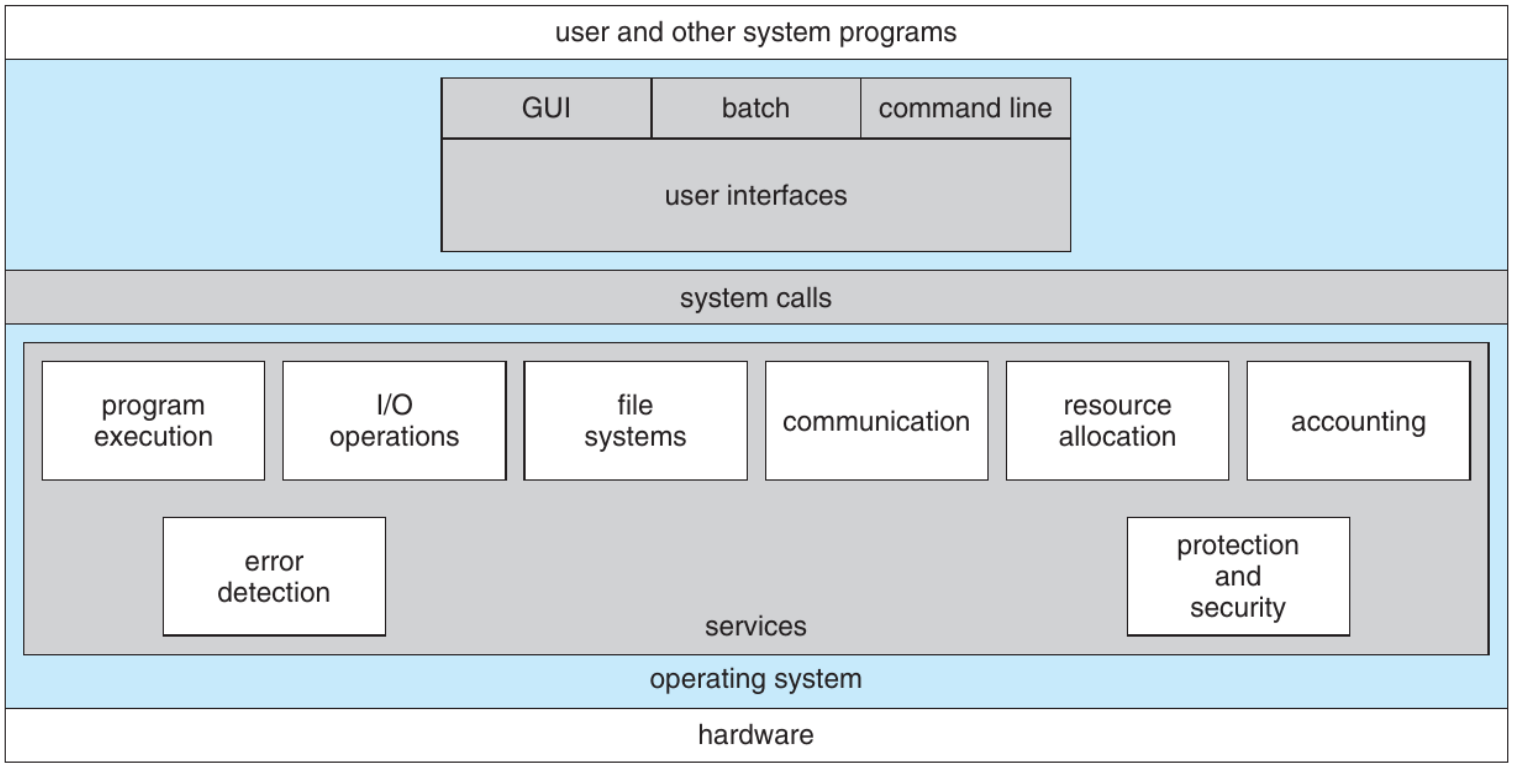
Serviços

- Sistemas operacionais fornecem serviços e ambientes para execução para programas[1];
  - Cada sistema operacional fornece um conjunto diferente de serviços, de acordo com seus objetivos;
  - Esses serviços podem ser convenientes aos programadores, facilitando o desenvolvimento de aplicações [Silberschatz et al., 2012];
  - Ao facilitar o desenvolvimento de aplicações, esses sistemas podem ter melhores programas, fazendo com que mais usuários utilizem esse SO.

[1] Consequentemente, os serviços também são oferecidos, mesmo que indiretamente, aos usuários.

Serviços

- Alguns serviços comumente fornecidos por SOs estão disponíveis na figura abaixo.



Visão geral de serviços dos Sistemas Operacionais.
Fonte: [Silberschatz et al., 2012]

Interfaces de Usuário

- Interfaces de usuário (UI), em geral, podem ser divididas em:
  - Interface de Linha de Comando (CLI): *shell* (terminal) e *command-prompt*;
  - Interface de Batch: possibilita a execução de arquivos em lote;
  - Interfaces Gráficas (GUI): interfaces que utilizam recursos gráficos para interação com o usuário.
- Sistemas operacionais podem fornecer uma ou mais interfaces de sistemas, de acordo com o propósito do sistema
  - Sistemas desktop: CLI e GUI;
  - Sistemas para dispositivos móveis: GUI;
  - Sistemas para servidores: Batch, GUI e CLI.

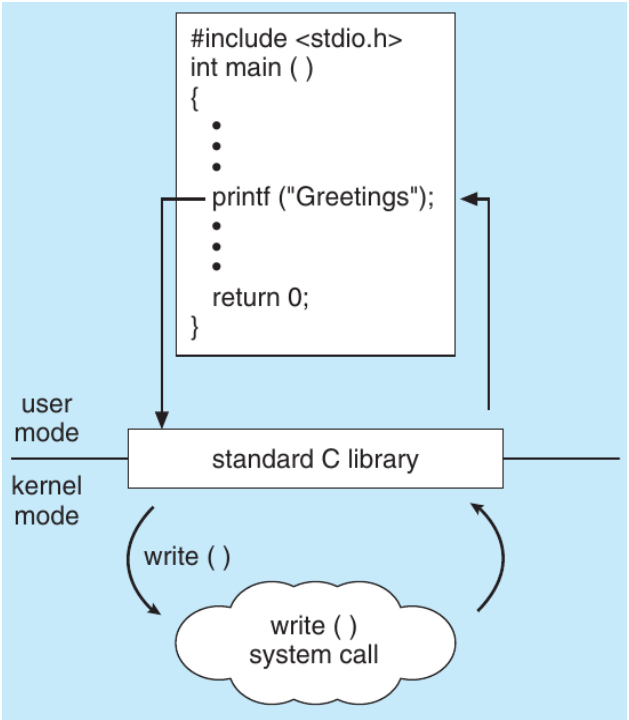
Chamadas de Sistema

- Chamadas do sistema fornecem uma interface para os serviços disponibilizados por um sistema operacional;
- SOs proveem abstrações para programas de usuários
  - O gerenciamento de recursos do SO é transparente para usuários e feito automaticamente;
  - Programas "conversam" apenas com uma interface do SO (API[2]), que provê acesso aos recursos desejados [Tanenbaum and Bos, 2014];

[2] API corresponde ao acrônimo de *Application Programming Interface*.

Chamadas de Sistema

- Para execução de um comando, a biblioteca de uma linguagem (ex. C, C++) pode interceptar a instrução e executar a chamada correspondente do SO [Silberschatz et al., 2012].



Chamadas de sistema em uma biblioteca C padrão.
Fonte: [Silberschatz et al., 2012]

Execução de Programas

- O SO deve ser capaz de carregar um programa na memória e executá-lo;
- Após a execução, o programa deve retornar a informação de finalização do processo ao SO
  - Caso exista algum problema na execução, o software pode indicar um código de erro [Silberschatz et al., 2012].

Nota: Em C++, por convenção, o sistema deve retornar 0 na função principal (*int main*) caso não exista erro. Ao retornar outros números, denominados níveis de erro (*error level*), o sistema indica que houve uma falha de execução. Esse erro pode ser recuperado e tratado por uma rotina que iniciou o programa (ex. *batch* de execução).

Operações de I/O

- Um programa em execução pode exigir acesso ou utilização de um dispositivo de Entrada / Saída;
- Por controle e proteção, o acesso direto não é autorizado aos softwares que desejam manipular esses dispositivos [Silberschatz et al., 2012].

Sistemas de Arquivos

- Sistemas operacionais gerenciam a leitura, escrita e o acesso a arquivos e diretórios;
- Programas podem manipular os arquivos utilizando chamadas de sistema (às vezes encapsuladas em rotinas definidas pelos compiladores).

Comunicação

- Processos podem comunicar com outros processos no mesmo computador ou com processos em outros dispositivos, em uma rede de computadores;
- A comunicação entre processo pode ser feita por meio de memória compartilhada[3] ou por meio de mensagens entre os processos.

[3] Dois ou mais processos possuem permissão para acessar um mesmo endereço de memória.

Alocação de Recursos

- Em sistemas multitarefas e/ou multiusuários, os recursos devem ser alocados para cada um deles;
- Para melhor funcionamento do SO e divisão de recursos entre usuários, o sistema deve compartilhar os recursos entre os usuários, além de dividir o uso da CPU entre os processos / usuários.

Contabilidade

- Sistemas operacionais podem analisar o uso de recursos computacionais para contabilidade (cobrança)
  - Servidores em nuvens podem cobrar por transmissão de dados pela rede, gravação / acesso a arquivos, uso de CPU, etc.;
  - A contabilização dos recursos é transformada em valores monetários, que são cobrados do usuário;
  - A contabilização também pode ser usada para estimar a vida útil dos dispositivos, evitando perda de informação (ex. SSD).

Contabilidade

- Sistemas operacionais podem também gerar estatísticas de uso dos dispositivos
  - Informações podem ser armazenadas, permitindo que o SO utilize algoritmos diferentes para melhoria de desempenho;
    - Ex. 1: a partir de estatísticas é possível alterar o mecanismo de gerenciamento da fila de processos, reduzindo a fila;
    - Ex. 2: a partir de estatísticas é possível alterar o tamanho do bloco de memória, permitindo melhor aproveitamento do recurso ou favorecendo um melhor desempenho do sistema;
  - Estatísticas sobre o uso do sistema podem ser usadas pelos administradores do sistema, para planejamento da aquisição de hardwares ou alteração da configuração de um servidor.

Detecção de Erros

- Erros devem ser detectados e corrigidos (ou administrados) pelos sistemas operacionais de forma constante;
- Falhas de processamento, falhas no acesso à memória, erros de leitura / gravação de discos de disco, erros nos dispositivos de I/O e nos programas de usuários devem ser identificados e administrados pelo sistema operacional
  - Esses erros não devem comprometer a integridade e o funcionamento adequado do sistema operacional.

Proteção e Segurança

- Proteção:
  - Todos os recursos de sistema devem ser controlados;
  - Quando vários processos separados são executados concorrentemente, o sistema deve evitar a interferência de um processo em outro;
  - Também devem evitar que um determinado processo interfira no SO [Silberschatz et al., 2012];

Proteção e Segurança

- Segurança:
  - O sistema deve controlar o uso de recursos não autorizados pelos usuários;
  - O sistema pode permitir apenas usuários autenticados ou restringir muito o acesso a recursos de usuários convidados;
  - Sistemas ainda podem adotar diretivas de segurança implementadas pela organização e definidas via regras da rede/domínio.

Arquiteturas de Sistemas Operacionais

Arquiteturas de Sistemas Operacionais

- Sistemas operacionais são construídos utilizando diferentes arquiteturas
  - Os primeiros sistemas utilizavam estruturas monolíticas, com poucas divisões entre componentes (ou módulos) do sistema;
  - No entanto, essa abordagem apresenta problemas com o crescimento dos sistemas e com o aumento da complexidade;
  - Outras arquiteturas surgiram para solução de desses problemas e trouxeram inovações, que foram incorporadas aos sistemas;
  - Na prática, muitos SOs não podem ser enquadrados em uma só categoria - porém o estudo delas é importante para avaliação de potenciais pontos fortes e fracos dos sistemas.

Sistemas Monolíticos

Sistemas Monolíticos

- Sistemas monolíticos correspondem a um tipo de arquitetura onde o sistema completo roda como se fosse um único programa em modo kernel
  - Sistemas monolíticos não possuem estruturas bem definidas;
  - O sistema consiste apenas a uma coleção de *procedures*, compiladas em um único programa binário executável;
  - Esse tipo de sistema é, historicamente, o mais comum;
  - Muitos deles, começaram pequenos, e foram sendo melhorados até se tornarem sistemas comerciais, com um escopo muito maior que o original [Tanenbaum and Bos, 2014] [Silberschatz et al., 2012].

Sistemas Monolíticos

- Nos sistemas monolíticos, os procedimentos podem chamar quaisquer outros, sem restrição
  - Esse tipo de estrutura reduz o tamanho do sistema, o que pode ser considerado uma vantagem;
  - No entanto, uma falha grave na execução de um procedimento causa um erro generalizado do sistema;
- Sistemas monolíticos tem dificuldade em adotar mecanismos de proteção e segurança, uma vez que a construção do sistema permite qualquer tipo de ação. [Tanenbaum and Bos, 2014]

Sistemas Monolíticos

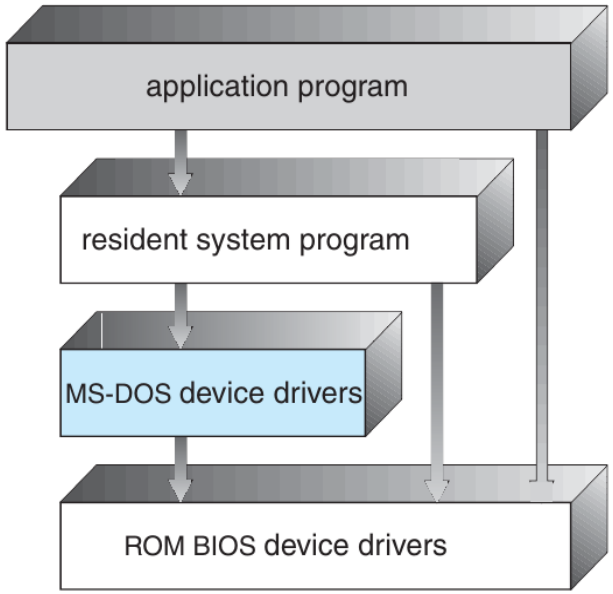
- Alguns sistemas monolíticos possuem uma organização primitiva, com a seguinte configuração:
  1. Um programa principal que invoca o procedimento de serviço solicitado;
  2. Um conjunto de procedimentos de serviço, que realizam as chamadas do sistema;
  3. Um conjunto de procedimentos utilitários, que ajudam os procedimentos de serviço [Tanenbaum and Bos, 2014].

Sistemas Monolíticos

- MS-DOS é um exemplo de sistema monolítico
  - As interfaces e os níveis de funcionalidade não são bem separados;
  - Os programas podem acessar as rotinas de I/O básicas para gravar diretamente no monitor e nas unidades de disco;
  - As permissões do MS-DOS deixam o sistema vulnerável a programas mal escritos, que podem causar o travamento de todo o sistema em caso de falhas;
  - É importante ressaltar que os hardwares da época não permitiam execução em modo dual nem proteção de hardware
    - Com isso, o MS-DOS era compatível com a tecnologia da época [Silberschatz et al., 2012].

Sistemas Monolíticos

- A figura abaixo contém a estrutura básica de um sistema MS-DOS.



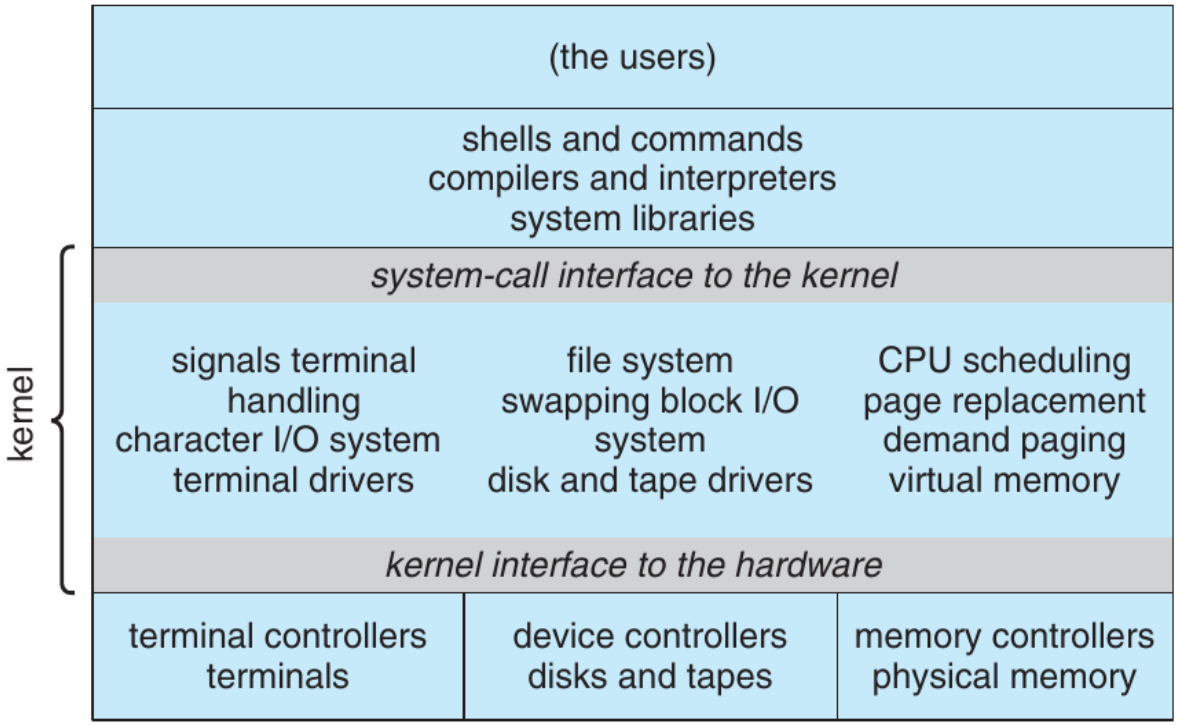
Sistemas Monolíticos - MS-DOS.
Fonte: [Silberschatz et al., 2012]

Sistemas Monolíticos

- O UNIX tradicional é outro exemplo de sistema monolítico
  - Originalmente, ele consistia de duas partes: kernel e programas de usuários
    - Ao longo do tempo, entretanto, o sistema evoluiu de modo a separar interfaces e drivers
  - No sistema original, qualquer estrutura acima da interface de chamadas de sistema podia ser considerada kernel;
  - Todos os recursos eram providos pelo kernel, mantidos em uma única estrutura difícil de implementar, manter e evoluir [Silberschatz et al., 2012].

Sistemas Monolíticos

- A figura abaixo contém a estrutura básica de um sistema UNIX tradicional.



Sistemas Monolíticos - UNIX.
Fonte: [Silberschatz et al., 2012]

Sistemas em Camadas

Sistemas em Camadas

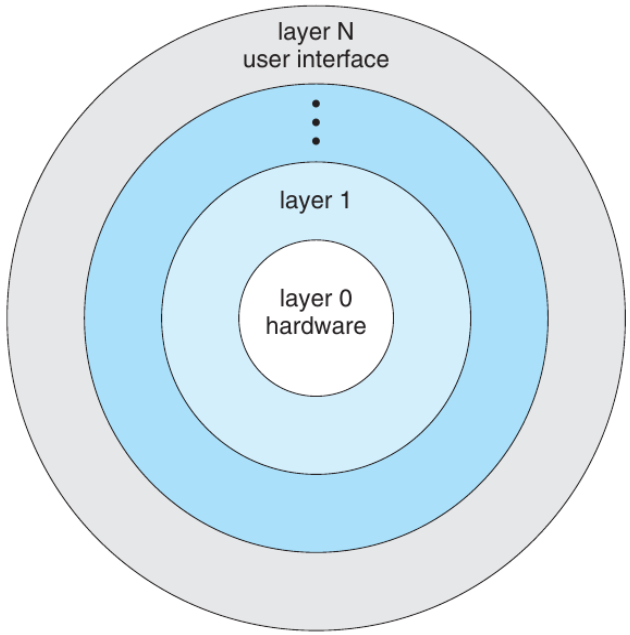
- Sistemas em Camadas correspondem a SOs que organizam sua estrutura em uma hierarquia de camadas (ou níveis)
  - A primeira camada pode ser considerada o hardware, enquanto a última camada corresponde à interface de usuário;
  - Cada uma das camadas intermediárias é responsável por um grande grupo de tarefas;
- Sistemas em camadas são mais simples de serem construídos e depurados (*debug*) [Tanenbaum and Bos, 2014] [Silberschatz et al., 2012].

Sistemas em Camadas

- Em um sistema em camadas, uma determinada camada *M* contém rotinas que podem ser invocadas por camadas de nível superior;
  - Entretanto, não pode solicitar operações às camadas superiores nem receber solicitações das camadas de níveis mais baixos.
- Sistemas operacionais em camadas requerem uma definição adequada das responsabilidades de cada uma das camadas
  - Com isso, o sistema precisa ser bem planejado e implementado para que funcione adequadamente [Tanenbaum and Bos, 2014] [Silberschatz et al., 2012].

Sistemas em Camadas

- A figura representa um sistema que utiliza uma arquitetura dividida em camadas.



Abordagem em Camadas.
Fonte: [Silberschatz et al., 2012]

Sistemas em Camadas

- Um primeiro sistema em camadas foi o sistema THE, criado por Dijkstra em 1968 [Tanenbaum and Bos, 2014].
- Esse sistema continha 6 camadas, detalhadas abaixo:

Camada	Funcionalidade
5	Usuário
4	Programas de usuários
3	Gerência de I/O
2	Comunicação entre processos e usuários
1	Gerência de memória
0	Alocação de processador e multiprogramação

Camadas do sistema THE.
Fonte: [Tanenbaum and Bos, 2014]

Sistemas em Camadas

- O sistema MULTICS foi outro sistema operacional que utilizava uma abordagem em camadas;
- [Tanenbaum and Bos, 2014] descreve o sistema como uma sequência de anéis concêntricos, onde os anéis internos possuem mais privilégios que os externos;
- Para um anel externo invocar um procedimento em um anel interno, era necessário que fosse feita uma solicitação equivalente a uma chamada de sistema [Tanenbaum and Bos, 2014].

Microkernels

Microkernels

- Microkernels correspondem a uma arquitetura de Sistemas Operacionais onde o kernel têm apenas as funcionalidades estritamente necessárias
  - O microkernel busca remover todas as camadas que não são obrigatórias, deixando o kernel o mais enxuto possível;
  - Tal abordagem reduz a possibilidade de falhas, uma vez que o código é mais conciso;
  - A arquitetura, então, possui menor probabilidade de erros graves que comprometam o funcionamento de todo o sistema;
  - A arquitetura também provê segurança e confiabilidade, uma vez que a maioria dos serviços é executado em modo usuário [Tanenbaum and Bos, 2014] [Silberschatz et al., 2012].

Microkernels

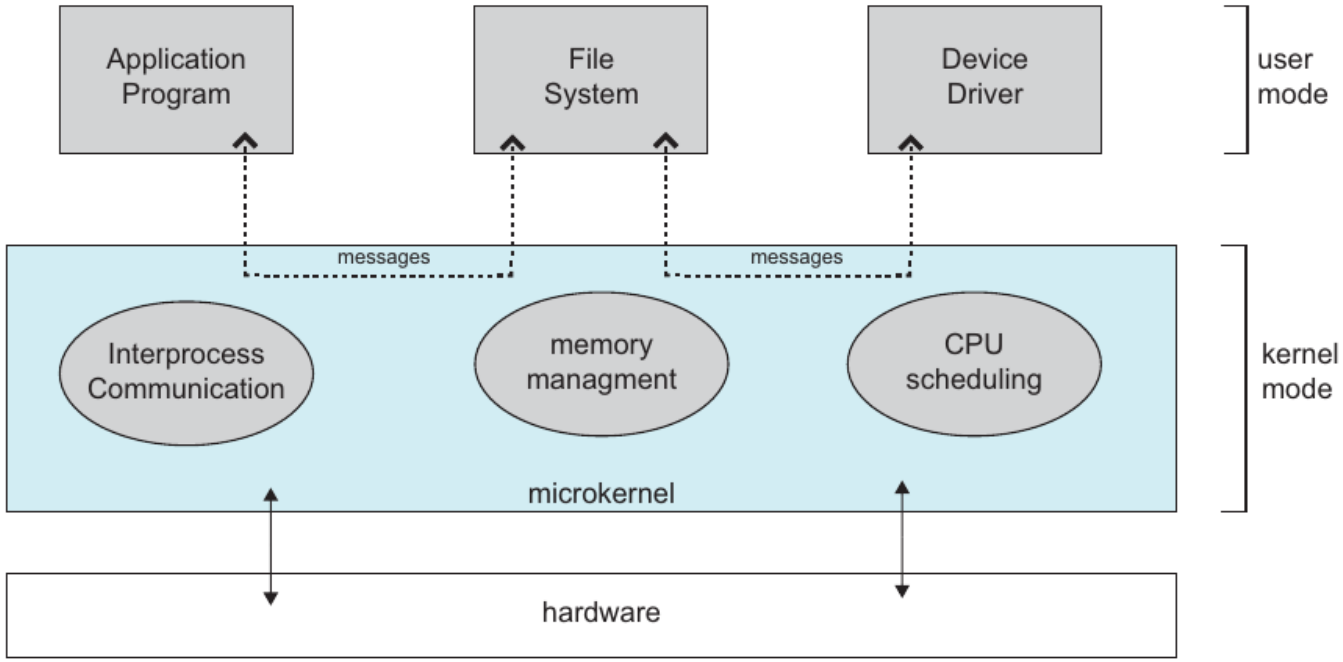
- Diversos estudos nos anos 1980, 1990 e 2000 indicaram que existem inúmeros bugs nos *kernels* de sistemas operacionais [Tanenbaum and Bos, 2014]
  - Nem todos esses *bugs* são comuns nem fatais, porém eles podem ocorrer em situações diversas;
  - Ao reduzir o tamanho do código, espera-se que a quantidade de *bugs* seja menor, deixando o kernel mais estável;
- A dificuldade desse mecanismo de microkernel é definir quais funcionalidades são estritamente necessárias e quais serão implementadas no espaço do usuário[Silberschatz et al., 2012].

Microkernels

- Os microkernels proveem o mínimo de gerenciamento de memória e processos [Silberschatz et al., 2012]
  - Essas tarefas rodam em modo kernel;
  - As atividades restantes são alocadas em processos ordinários, de usuários;
- O microkernel também fornece mecanismos de comunicação entre os programas do cliente e os serviços
  - Os serviços são executados no espaço do usuário;
  - A comunicação é feita por meio de troca de mensagens;
  - O programa do cliente e o serviço nunca interagem diretamente [Silberschatz et al., 2012].

Microkernels

- A figura abaixo representa um sistema que utiliza uma arquitetura de microkernels.



Microkernels.
Fonte: [Tanenbaum and Bos, 2014]

Microkernels

- Nos anos 1980, pesquisadores de universidade Carnegie Mellon desenvolveram um sistema operacional denominado Mach
  - Esse sistema utilizava a abordagem em microkernels;
- O kernel do sistema Mac OS X (denominado Darwin) é parcialmente baseado no microkernel do Mach
  - Apesar de utilizado no Mac OS X, a abordagem em microkernels não é muito comum em SOs para desktop;
  - No entanto, essa arquitetura é bastante comum em sistemas de tempo real, indústria e aviação [Tanenbaum and Bos, 2014].

Microkernels

- Os sistemas operacionais Symbiam e MINIX são outros exemplos de abordagem em microkernels;
- O sistema MINIX 3 é um sistema de código aberto em conformidade com padrão POSIX
  - O sistema busca levar a modularidade do sistema ao limite;
  - É subdividido em um conjunto de processos independentes em modo usuário;
  - O microkernel tem cerca de 12.000 linhas de código em C e 1.400 linhas em Assembly [Tanenbaum and Bos, 2014].

O sistema MINIX originalmente foi a base do sistema operacional Linux.

Microkernels

- Apesar das vantagens da arquitetura em microkernel, o desempenho dos sistemas podem sofrer devido à necessidade de fornecimento de uma maior quantidade de recursos.
  - O Windows NT, originalmente, optou por uma organização em microkernel;
  - Seu desempenho, entretanto, era baixo, quando comparado ao Windows 95;
  - No Windows NT 4.0, muitas das funcionalidades foram movidas para o espaço de kernel, aumentando o desempenho;
  - O Windows XP, derivado do Windows NT, possuia uma arquitetura mais próxima ao sistema monolítico do que da arquitetura em microkernel [Silberschatz et al., 2012].

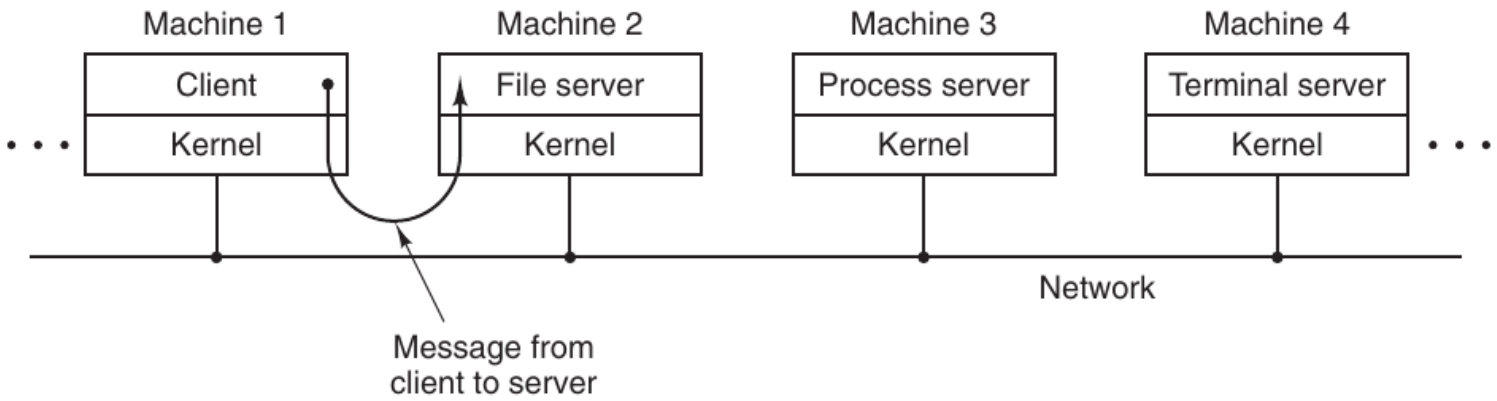
Sistemas Cliente-Servidor

Sistemas Cliente-Servidor

- Sistemas Cliente-Servidor correspondem a uma variação da arquitetura de microkernel, onde os processos são divididos em duas classes distintas: servidores e clientes.
  - Processos servidores são aqueles que fornecem algum serviço;
  - Processos clientes são aqueles que usam esses serviços.
- A comunicação entre clientes e servidores é feita por meio de troca de mensagens
  - Para obter um serviço, um cliente cria uma mensagem e a envia ao serviço apropriado;
  - O serviço processa a requisição e retorna a resposta [Tanenbaum and Bos, 2014]

Sistemas Cliente-Servidor

- Modelos clientes-servidores podem ser generalizados para trabalharem em computadores diferentes
  - Essa arquitetura pode ser utilizada para trabalhar em redes locais ou via internet [Tanenbaum and Bos, 2014].



Estrutura Cliente-Servidor em uma rede de computadores.
Fonte: [Tanenbaum and Bos, 2014]

Sistemas Modulares

Sistemas Modulares

- Sistemas modulares são uma arquitetura de Sistemas Operacionais que utilizam módulos carregáveis de kernel[4]
  - O kernel possui um conjunto de componentes principais e links de serviços adicionais, utilizando módulos;
  - Os módulos podem ser carregados durante a inicialização do sistema ou, caso necessário, durante a execução;
- Essa arquitetura pode ser encontrada nos sistemas Windows, Linux, Solaris e Mac OS X [Silberschatz et al., 2012].

[4] Tradução do inglês de *Loadable Kernel Module* (LKM).

Sistemas Modulares

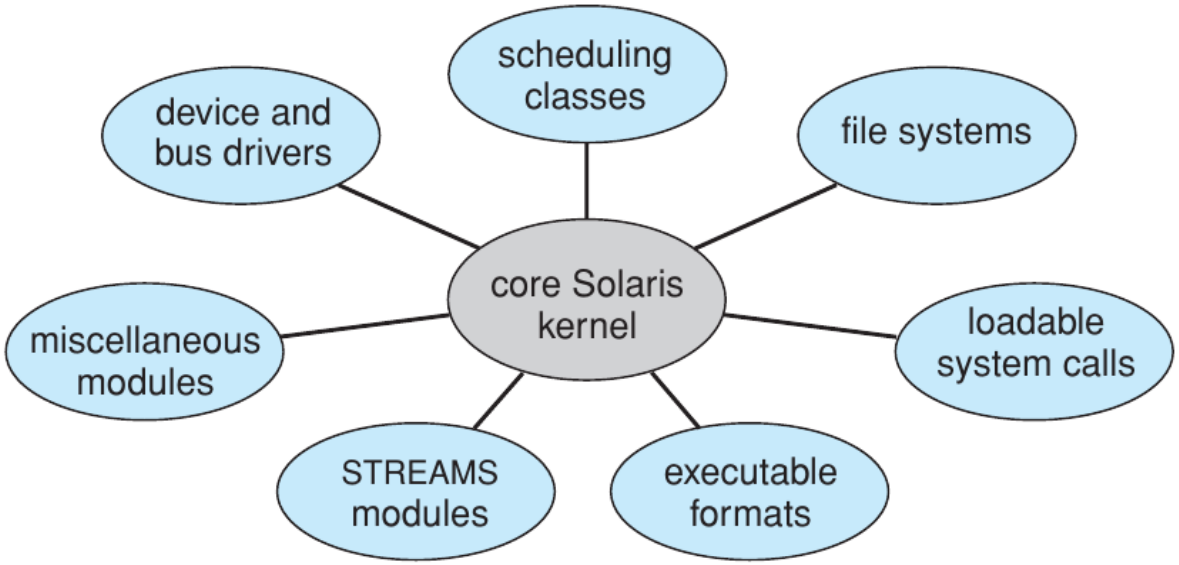
- Nos sistemas que utilizam arquitetura modular, o kernel deve fornecer apenas serviços essenciais
  - Outros serviços são implementados (ou carregados) dinamicamente, enquanto o kernel está em execução;
  - Adicionar recursos dinamicamente permite a atualização e adição de novos funcionalidades ao kernel;
  - Esse tipo de estrutura não exige a recompilação do kernel em caso de modificações [Silberschatz et al., 2012].

Sistemas Modulares

- A arquitetura modular é semelhante ao sistema em camadas no sentido de que cada seção do kernel possui interfaces definidas e protegidas
  - No entanto, é mais flexível, uma vez que um módulo pode invocar outro qualquer [Silberschatz et al., 2012];
- A arquitetura modular é semelhante à abordagem de microkernel uma vez que o módulo principal tem apenas funções essenciais e é capaz de carregar outros módulos.
  - No entanto, é mais eficiente, pois os módulos não precisam utilizar troca de mensagens para se comunicar [Silberschatz et al., 2012].

Sistemas Modulares

- O Solaris é um exemplo de sistema modular, conforme representação na figura abaixo.
  - Nesse sistema, os módulos são organizados ao redor do kernel.



Sistemas Modulares - Solaris.
Fonte: [Silberschatz et al., 2012]

Referências I

Silberschatz, A., Galvin, P. B., and Gagne, G. (2012).
Operating System Concepts.
Wiley Publishing, 9th edition.

Tanenbaum, A. S. and Bos, H. (2014).
Modern Operating Systems.
Prentice Hall Press, USA, 4th edition.