Sistemas Operacionais

Prof. Dr. Márcio Andrey Teixeira

Estrutura de sistemas operacionais

Estruturas de sistemas operacionais

A estrutura e o funcionamento de um SO são tópicos de difícil compreensão. Um SO não é executado como uma aplicação seqüencial, com início, meio e fim. As rotinas do SO são executadas sem uma ordem predefinida.

Existem diferentes estruturas de sistemas operacionais, por exemplo:

- sistemas monolíticos;

- sistemas em camadas;

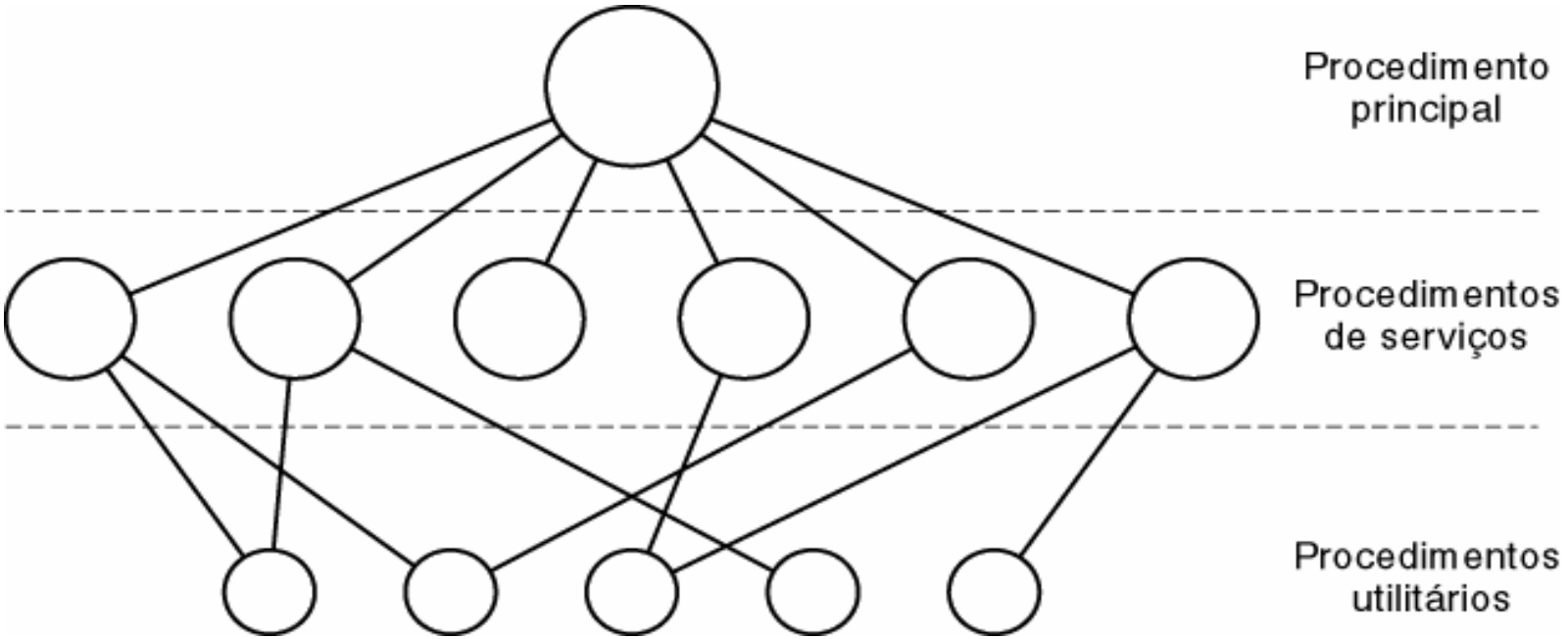
- sistemas clientes servidores;

A seguir estudaremos cada item acima citado.

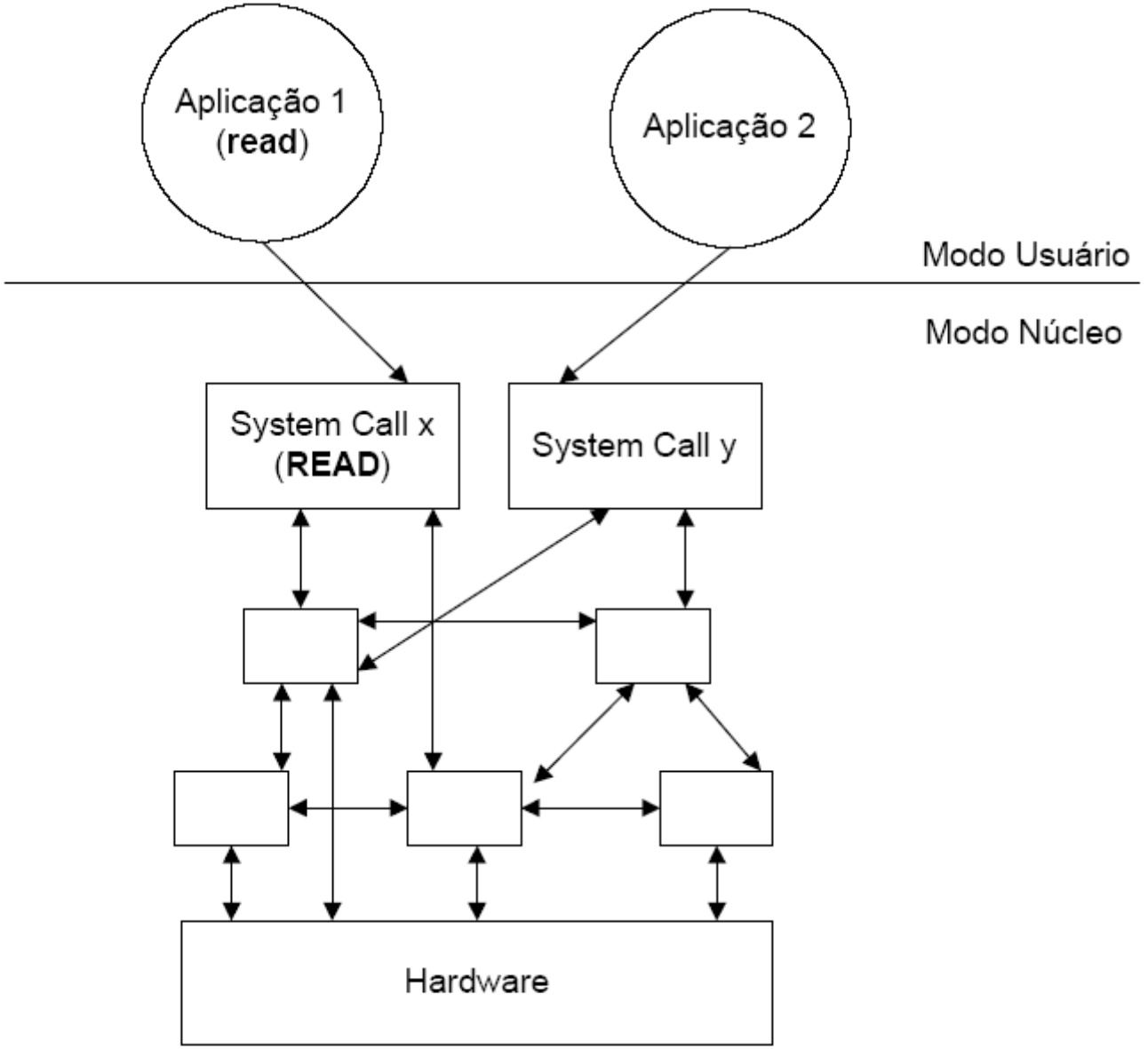
Sistemas monolíticos

Neste tipo de estrutura, o SO é escrito como uma coleção de rotinas, onde cada rotina pode chamar qualquer outra rotina, sempre que for necessário.

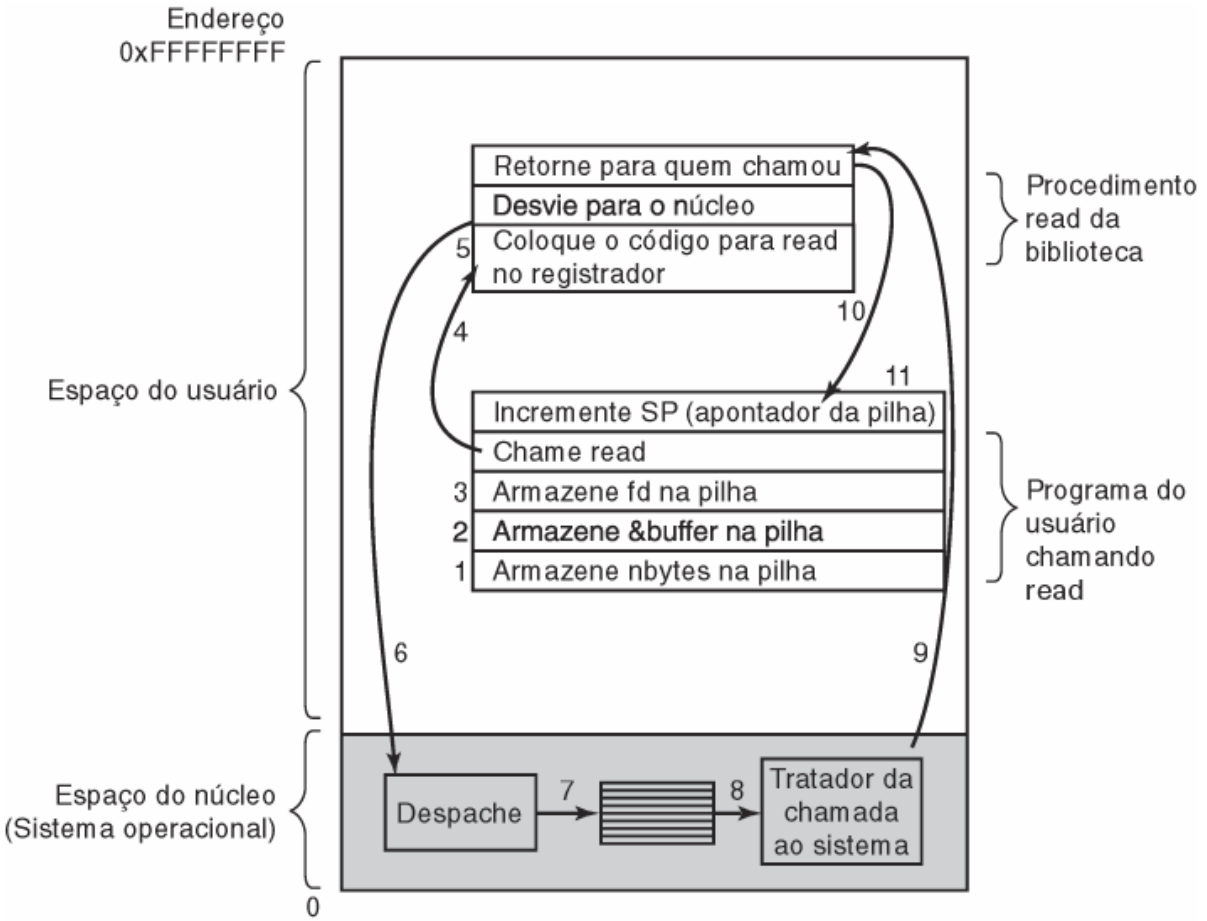
Portanto, o sistema é estruturado de forma que as rotinas podem interagir livremente umas com as outras. Quando esta técnica é usada, cada rotina no sistema possui uma interface bem definida em termos de parâmetros e resultados.



A figura a seguir mostra um exemplo de um sistema monolítico:



A figura a seguir mostra um exemplo de uma chamada de sistema “read”.

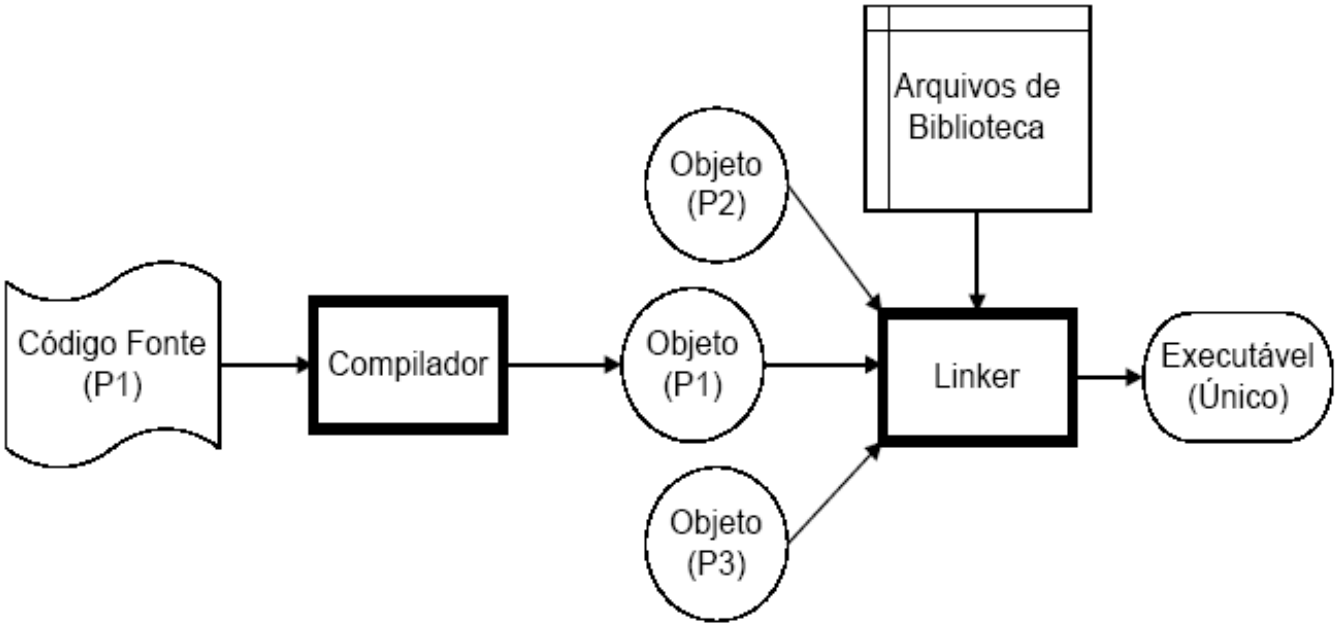


Os serviços do SO (gerenciamento de processos, gerenciamento de memória, gerenciamento do sistema de arquivos, ...) são implementados por meio de *System Calls* em diversos módulos, executando em Modo Núcleo.

Como todos os módulos (rotinas) executam no mesmo espaço de endereçamento, um *bug* em um dos módulos pode derrubar o sistema inteiro. Evidentemente que esta é uma situação indesejável.

Tal código executa em Modo Núcleo. Não existe ocultação de informação, o que também é indesejável, pois cada rotina é visível a qualquer outra.

A Figura a seguir mostra o processo de criação de um código executável de um SO Monolítico.



Embora possa parecer que não há quase estruturação em um SO Monolítico, existe um pouco de estruturação quando os serviços do SO são solicitados por meio das *System Calls*.

Como vantagem dos SOs Monolíticos pode-se afirmar que, se a implementação do sistema está completa e confiável, a forte integração interna dos componentes permite que detalhes de baixo nível do hardware sejam efetivamente explorados, fazendo com que um bom SO Monolítico seja altamente eficiente. Entre os SOs Monolíticos estão as versões tradicionais do **UNIX**, incluindo o **Linux**, e também o **MS-DOS**.

Sistemas em Camadas

A idéia por trás deste tipo de SO é fazer a organização por meio de hierarquia de camadas.

O SO é dividido em camadas sobrepostas, onde cada módulo oferece um conjunto de funções que podem ser utilizadas por outros módulos.

Módulos de uma camada podem fazer referência apenas a módulos das camadas inferiores.

O primeiro SO construído de acordo com esta concepção foi o **THE**, que foi desenvolvido na Technische Hogeschool Eindhoven na Holanda por E. W. Dijkstra (1968) e seus estudantes.

O computador que executava o THE possuía Memória Principal com capacidade de 32K palavras de 27 bits cada. A estrutura do **THE** pode ser vista na Figura a seguir:

Sistemas em Camadas

A idéia por trás deste tipo de SO é fazer a organização por meio de hierarquia de camadas.

Camada	Função
5	O operador
4	Programas do usuário
3	Gerenciamento de entrada/saída
2	Comunicação operador-processo
1	Gerenciamento da memória e do tambor magnético
0	Alocação de processador e multiprogramação

Sistemas em Camadas

A camada 0 era responsável pela alocação do processador entre os processos, chaveamento entre processos quando ocorria interrupções ou quando os temporizadores expiravam. Resumindo, a camada 0 fornecia a multiprogramação básica da CPU.

Acima da camada 0, o sistema consistia de processos seqüenciais que podiam ser programados sem se preocupar se havia múltiplos processos executando na CPU.

A camada 1 realizava o gerenciamento de memória. Ela alocava espaço para os processos na Memória Principal do sistema e também em um Tambor (dispositivo de armazenamento magnético usado nos computadores antigamente) de 512K palavras, usado para armazenar partes de processos (páginas) para as quais não havia espaço na Memória Principal.

Acima da camada 1, os processos não tinham que se preocupar se eles estavam na Memória Principal ou no Tambor; a camada 1 do SO era quem tratava deste tipo de situação, trazendo as partes do software para a Memória Principal sempre quando necessário.

Sistemas em Camadas

A camada 2 manipulava a comunicação entre cada processo e o operador do *console*. Um *console* consistia de um dispositivo de entrada (teclado) e um de saída (monitor ou impressora).

A camada 3 era responsável pelo gerenciamento dos dispositivos de E/S. Acima da camada 3, cada processo podia lidar com dispositivos de E/S abstratos, com propriedades mais agradáveis, e não com os dispositivos reais em si.

Na camada 4 havia os programas do usuário, e na camada 5 havia o processo do operador do sistema.

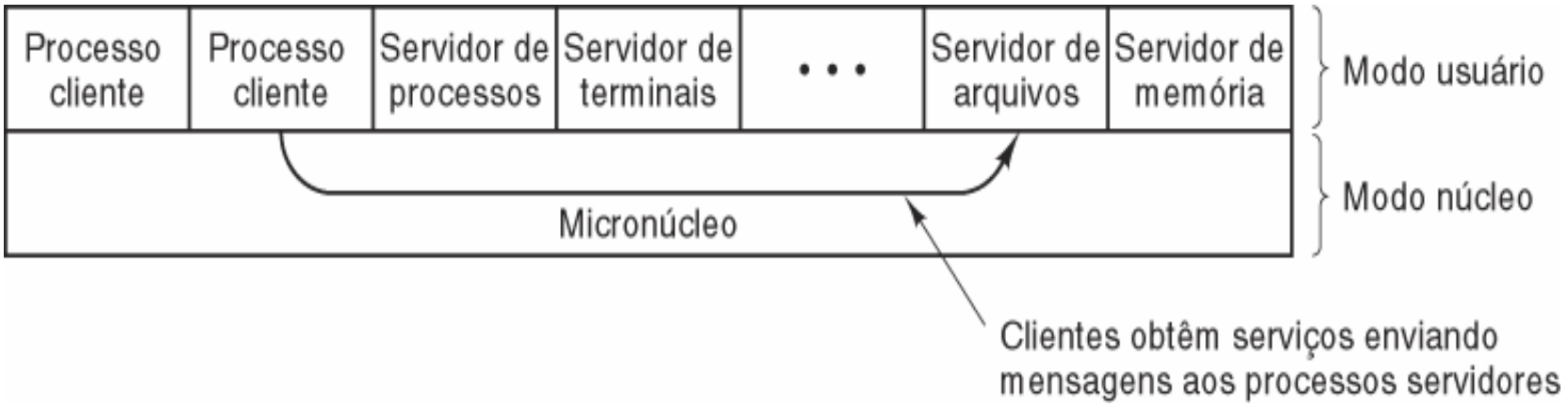
O esquema de camadas do THE era, de fato, apenas um auxílio de desenho (*design*), pois todas as partes do sistema eram intimamente unidas em um único código executável.

Sistemas Cliente - Servidor

Uma tendência dos sistemas operacionais modernos é transferir códigos para as camadas mais superiores e remover o máximo possível de código em modo núcleo, deixando um micronúcleo mínimo (também chamado de **microkernel**)

Normalmente, se implementa o máximo do sistema operacional como um processo do usuário. Para requisitar um serviço, como ler um bloco de arquivo, um processo de usuário (**agora conhecido como processo cliente**), envia uma requisição para um **processo servidor**, que estão executa o trabalho e envia a resposta.

A figura a seguir mostra a arquitetura cliente-servidor:



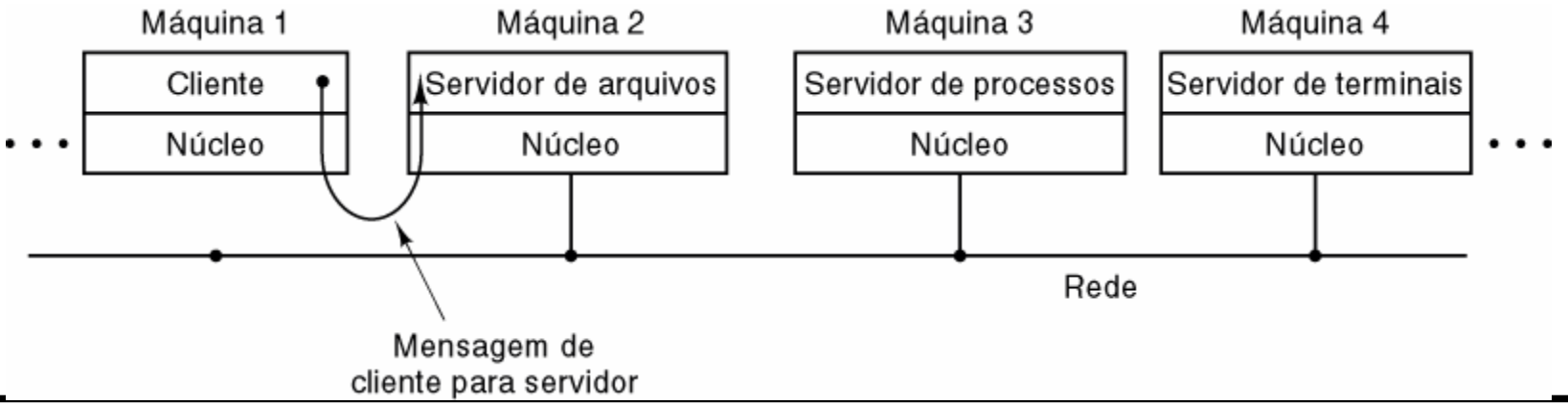
Sistemas Cliente - Servidor

Como mostra a figura anterior, tudo o que o núcleo faz é tratar da comunicação entre clientes e servidores., dividindo o sistema operacional em várias partes, por exemplo:

- servidor de arquivos;
- servidor de processos;
- servidor de terminais;
- servidor de memória;

Todos esses processos executam em modo usuário, com isso, se algum problema ocorrer, o sistema não será danificado;

O sistema cliente-servidor possui uma adaptabilidade ao uso em sistema distribuídos. Se um cliente se comunica com processo servidor enviando-lhe uma mensagem, o cliente não precisa saber se esta mensagem será tratada local ou remoto.



Sistemas Cliente – Servidor X Sistemas monolíticos

Em uma primeira análise, uma estrutura de SO Cliente-Servidor parece ser bem melhor do que um SO Monolítico. Porém, em termos práticos, a implementação de uma estrutura Cliente-Servidor é bastante complicada devido a certas funções do SO exigirem acesso direto ao hardware, como operações de E/S.

Um núcleo Monolítico, por outro lado, possui uma complexidade menor, pois todo código de controle do sistema reside em um espaço de endereçamento com as mesmas características (Modo Núcleo).

Um aspecto interessante sobre qual a melhor estrutura de SO foi a discussão entre ***Linus Torvalds***, o criador do SO Linux, e ***Andrew Tanenbaum***, um dos principais pesquisadores na área de SOs e criador do SO Minix.

Em 1992, quando o Linux estava no seu início, ***Tanenbaum*** decidiu escrever uma mensagem para o ***Newsgroup comp.os.minix***, acusando justamente o Linux de ser um SO obsoleto. O ponto principal do argumento de ***Tanenbaum*** era justamente a estrutura Monolítica, considerada por ele ultrapassado.

Sistemas Cliente – Servidor X Sistemas monolíticos

http://choices.cs.uiuc.edu/cache/Linus_vs_Tanenbaum.html

O Windows 2000

A história do Windows NT iniciou com o desejo da Microsoft de criar um sistema operacional que explorasse as inovações tecnológicas apresentadas pelos processadores no final da década de 80, início da década de 90.

O objetivo era desenvolver um sistema operacional multitarefa para ser utilizado tanto em ambientes monousuário como multiusuário. O nome *Windows* é originário de um sistema de janelas (*Windows 3.x for Workgroup*) projetado para competir com a interface usuário dos computadores Macintosh (Apple).

Esse ambiente de janelas emprestou a sua “aparência” para a primeira versão do Windows NT. A sigla NT vem de *New Technology*, e foi criada para caracterizar a nova filosofia que orientou a sua concepção.

A primeira versão do Windows NT (versão 3.1) foi lançada em 1993 e constituiu o primeiro sistema operacional de 32 bits da Microsoft. Esse sistema operacional caracterizava-se por fornecer uma compatibilidade com o sistema operacional MS-DOS, com aplicações desenvolvidas para o “velho” sistema de janelas (*Windows 3.x for Workgroup*).

Após sucessivas versões do Windows NT 3.x, nasce o Windows NT 4.0. Em relação à arquitetura interna de sistema operacional, o NT 4.0 mantém essencialmente a mesma de seu antecessor (Windows NT 3.x).

As principais modificações em relação ao Windows NT 3.x estão na interface gráfica, que agora se assemelha à do Windows 98, e na migração de vários serviços, também relacionados com a parte gráfica, do subsistema Win32 para o núcleo do Windows NT 4.0.

Em 1999, a Microsoft lançou uma nova versão do Windows NT, a 5.0, que comercialmente recebeu o nome de Windows 2000. A estrutura básica do sistema operacional é a mesma do NT 4.0. A principal diferença está na inclusão de serviços orientados a ambientes distribuídos e de rede. Na realidade, dependendo das funcionalidades adicionadas ao Windows 2000, existem 4 diferentes versões desse sistema operacional:

- Windows 2000 Professional, o qual substitui o NT workstation, isto é, as máquinas empregadas como ponto de trabalho (máquina cliente).

- Windows 2000 Server, equivalente ao NT Server. Essa configuração apresenta alguns serviços orientados ao compartilhamento de recursos e destina-se, como o próprio nome induz, a máquinas servidoras em uma rede NT.

- Windows 2000 Advanced Server, que fornece uma série de facilidades para ambientes de rede e distribuídos, incluindo o conceito de *clustering* e suporte ao balanceamento de carga.

- Windows 2000 Datacenter Server, que agrega todas as funcionalidades disponíveis no Windows 2000 e suporta o endereçamento de até 64 GB.

O desenvolvimento do Windows 2000 foi orientado por cinco objetivos principais que sempre nortearam o projeto de todos os produtos da família Windows NT: confiabilidade e robustez; extensibilidade e facilidade de manutenção; portabilidade; desempenho; e, por último, conformidade com o padrão POSIX

O objetivo de ***confiabilidade e robustez*** traduz-se no fato de que um sistema deve ter a capacidade de se proteger do mau funcionamento e de problemas oriundos do próprio sistema operacional, assim como de fontes externas (ataques).

O segundo objetivo de projeto, ***extensibilidade e facilidade de manutenção***, diz respeito à perenidade do sistema. Era necessário que o Windows 2000 pudesse evoluir, adaptando-se facilmente a novas necessidades, tanto de hardware como de software.

Arquitetura do Windows 2000: visão geral

Um sistema operacional é um *software* extremamente complexo. Assim, vários modelos de arquiteturas foram propostos para melhor organizar os detalhes de sua implementação. Esses modelos vão desde sistemas baseados em *kernel* monolítico até sistemas totalmente moduláveis, baseados em micronúcleo (*microkernel*).

A arquitetura do Windows 2000 é fortemente inspirada no princípio de micronúcleo. Assim, cada funcionalidade do sistema é oferecida e gerenciada por um único componente do sistema operacional.

Os demais componentes do sistema operacional e todas as aplicações acessam os serviços providos por um determinado componente através de uma interface bem definida.

Teoricamente, cada módulo (componente) pode ser removido, atualizado, ou substituído sem necessitar de alterações nas demais partes do sistema. O Windows 2000 não é puramente orientado à filosofia micronúcleo porque módulos fora do micronúcleo executam operações em modo protegido (modo kernel).

A justificativa para essa decisão de projeto está no desempenho. Em uma abordagem orientada a micronúcleo “pura”, uma aplicação que necessite executar uma operação privilegiada deve solicitar esse serviço ao micronúcleo. Esse procedimento envolve uma série de trocas de contexto.

No Windows 2000, para evitar essa troca de contexto, certos subsistemas (módulos ou componentes) passam de modo usuário para modo protegido e implementam diretamente a função desejada, evitando assim a passagem pelo núcleo e as trocas de contexto que isso implica.

O Windows 2000 segue também uma organização em camadas. Nessa abordagem, o sistema operacional é dividido em módulos que são dispostos uns sobre os outros em camadas. Cada camada oferece um conjunto de serviços à camada superior e só pode utilizar serviços fornecidos pela camada imediatamente inferior.

Outro conceito explorado pelo Windows 2000 é o modelo orientado a objetos. Nesse modelo, recursos do sistema, arquivos, memória e dispositivos físicos, são implementados por objetos e manipulados através de métodos (serviços) associados a esses objetos.

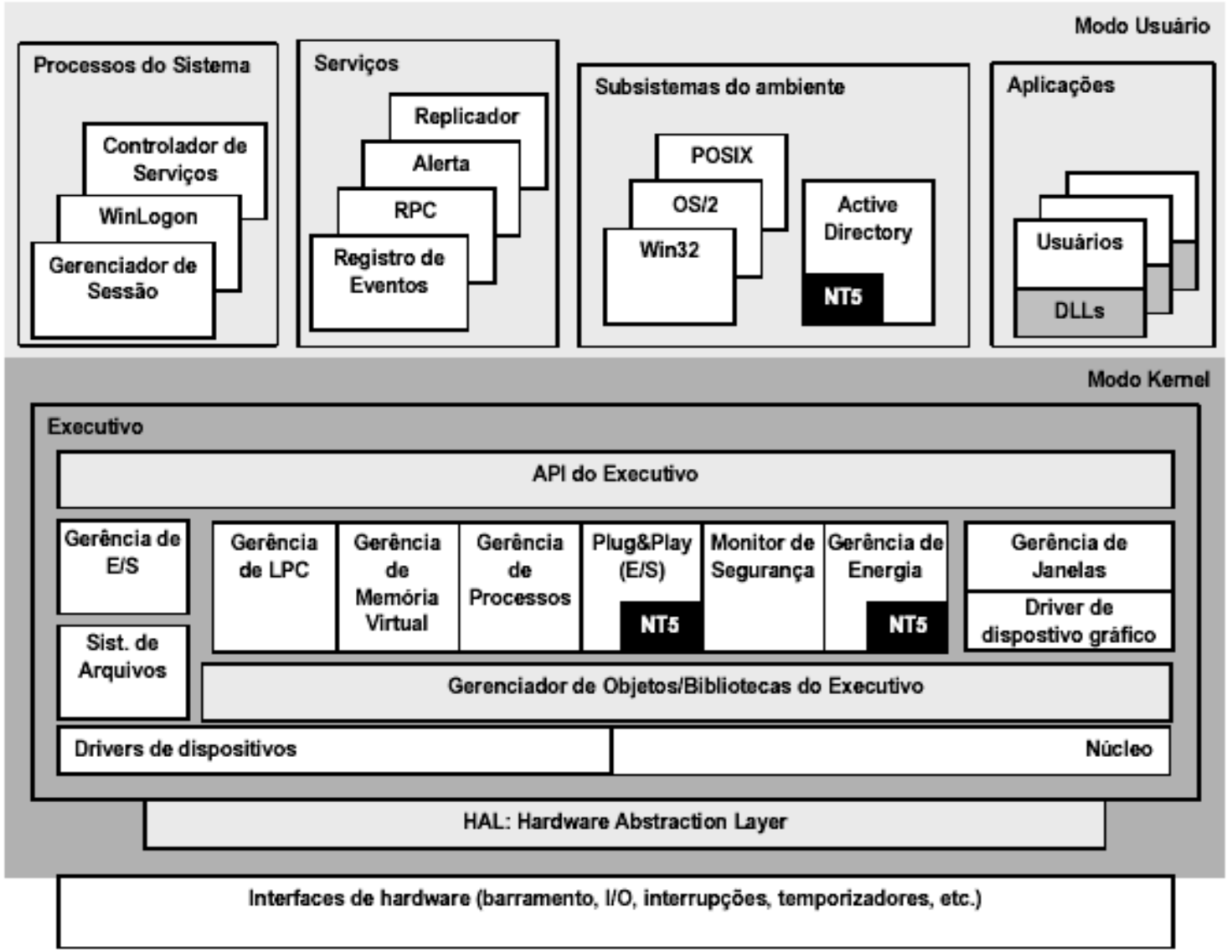
A estrutura do Windows 2000 pode ser dividida em duas partes: modo usuário (onde estão localizados os subsistemas protegidos) e modo kernel (o executivo). Os subsistemas protegidos são assim denominados porque residem em processos separados cuja memória é protegida do acesso de outros processos.

Os subsistemas interagem entre si através de um mecanismo de troca de ***mensagens (Local Procedure Call - LPC)***. No modo kernel, rodam os componentes do sistema operacional que necessitam de desempenho e por isso interagem com o hardware e um com o outro sem estarem sujeitos a trocas de contexto e de modo.

Todos os componentes estão protegidos das aplicações porque estas não possuem acesso à parte protegida do sistema operacional. Ainda, cada componente está protegido um do outro devido à adoção da orientação a objetos. Todo acesso a um objeto é feito através de um método.

O modo kernel é estruturado em três grandes módulos funcionais: ***hardware abstraction layer, drivers de dispositivos e o executivo. A camada denominada de hardware abstraction layer*** (HAL) é um módulo carregável do núcleo. A figura a seguir mostra a arquitetura do windows 2000.

Estrutura do Windows NT



O windows NT apresenta uma arquitetura microkernel;

O Unix - Linux

A origem do Unix tem ligação com o sistema operacional Multics, projetado na década de 1960. Esse projeto era realizado pelo Massachusets Institute of Technology (MIT), pela General Eletric (GE) e pelos laboratórios Bell (Bell Labs) e American Telephone na Telegraph (AT&T).

A intenção era de que o Multics tivesse características de tempo compartilhado (vários usuários compartilhando os recursos de um único computador), sendo assim, o sistema mais arrojado da época.

Ken Thompsom era um pesquisador do Multics e trabalhava na Bell Labs. No entanto, a empresa se retirou do projeto tempos depois, mas ele continuou seus estudos no sistema. Desde então, sua idéia não era continuar no Multics original e sim criar algo menor, mas que conservasse as idéias básicas do sistema. A partir daí, começa a saga do sistema Unix. Brian Kernighan, também pesquisador da Bell Labs, foi quem deu esse nome.

A primeira versão do Unix foi escrito na linguagem assembler. Em 1973, outro pesquisador da Bell Labs, Dennis Ritchie, rescreveu todo o sistema Unix numa linguagem de alto nível, chamada C, desenvolvida por ele mesmo. Por causa disso, o sistema passou a ter grande aceitação por usuários externos à Bell Labs.

Em 1980, a Universidade da Califórnia em Berkely foi financiada pelo Departamento de Defesa para desenvolver uma versão de UNIX voltada para os novos sistemas de computação distribuída. Esse esforço resultou no sistema 4.1 BSD.

No início dos anos 80, a Microsoft, oferecia uma versão comercial do UNIX, denominada XENIX, baseada em micro computadores compatíveis com o IBM PC, de 16 bits (8086, 80286). Esse sistema evoluiu para o SCO-Unix.

Em 1982 a AT&T passou a oferecer ao mercado o UNIX System III, que evoluiu para o System V. Ao ser proibida de atuar no mercado de computação em conseqüência das legistação americana, a AT&T licenciou o System V para outras empresas e instituições de pesquisa.

A Sun Microsystems desenvolveu a sua versão do UNIX, denominada SunOs, com base no 4.2 BSD, de Berkeley. O sistema da Sun introduziu o conceito de ***Network File System***, NFS, que possibilita o compartilhamento de arquivos por máquinas ligadas em rede local.

No final dos anos 80, havia duas grandes "famílias" de sistemas UNIX, incompatíveis entre si: 4.3 BSD e System V Release 3. Cada fabricante incorporava suas próprias melhorias.

Varias tentativas foram feitas no sentido de se padronizar as interfaces do UNIX, principalmente no sentido de permitir o porte das aplicações para as várias versões do sistema.

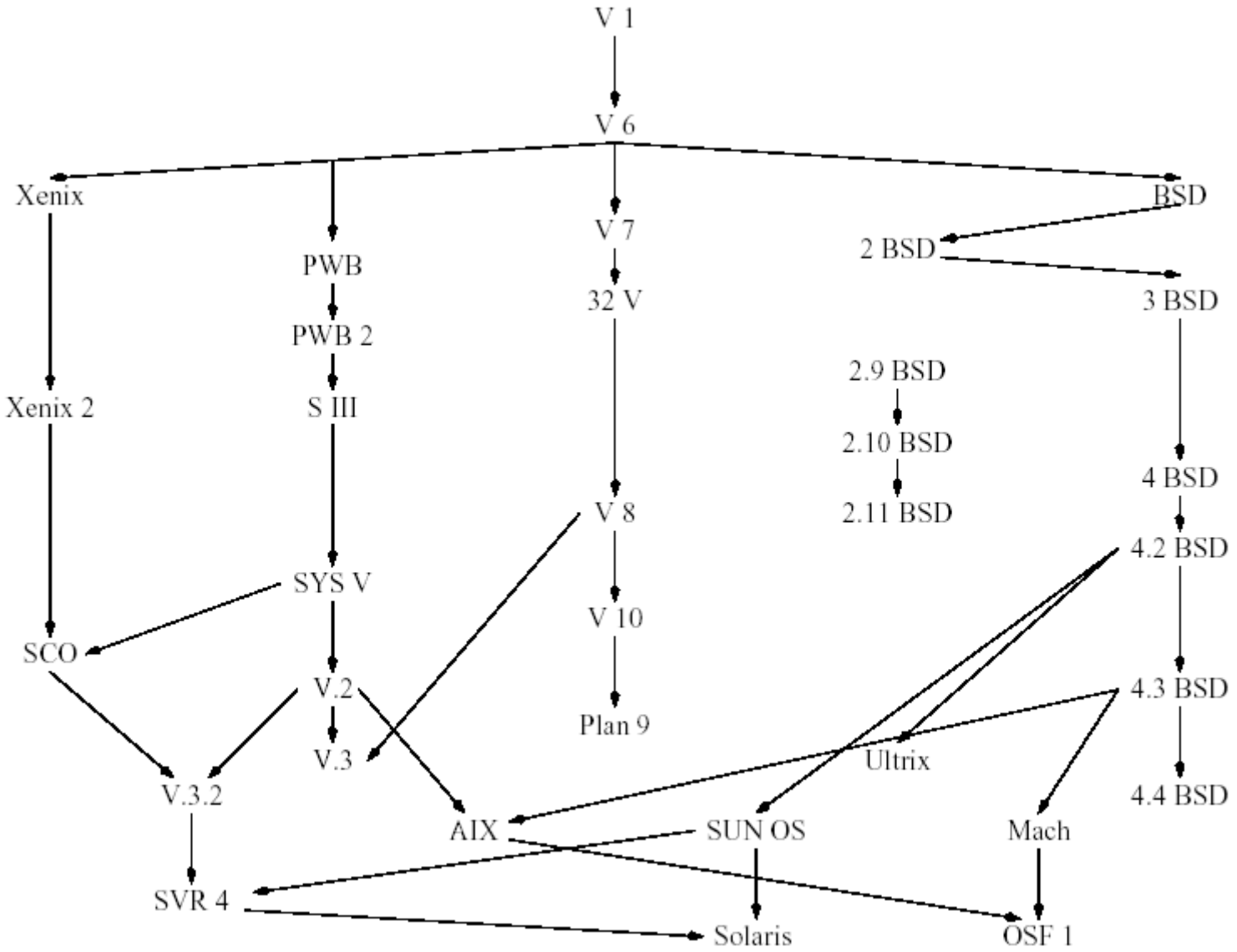
Dessas tentativas, a que pode ser considerada bem sucedida foi patrocinada pelo IEEE Standards Board, que definiu o padrão *POSIX*.

Esse padrão define basicamente o conjunto de funções de biblioteca que todo sistema que o adote deve oferecer aos programas de aplicação.

No início dos anos 90, Linus Torvalds se propôs a desenvolver um núcleo de sistema operacional que utilizasse os recursos de *modo protegido* oferecidos pelo processador Intel 386, baseado no sistema operacional *Minix*, construído para fins didáticos.

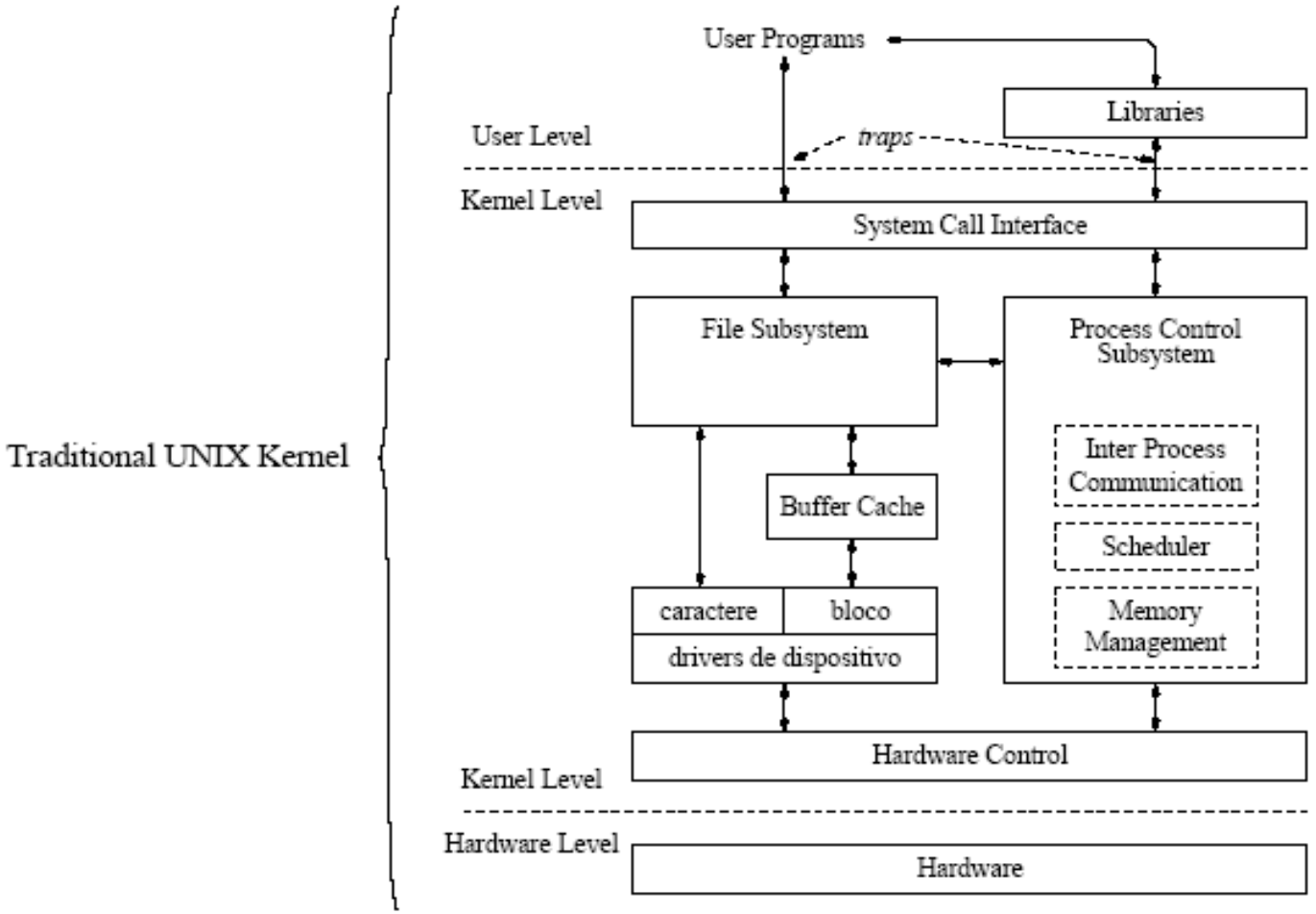
Esse sistema evoluiu para uma versão de UNIX distribuído pela rede na modalidade *open source*, o código fonte pode ser baixados e distribuído gratuitamente.

O sistema se disseminou rapidamente e hoje ocupa uma fatia considerável das instalações tanto domésticas quanto institucionais.


V 1
V 6
Xenix
PWB
V 7
BSD
2 BSD
3 BSD
PWB 2
32 V
Xenix 2
S III
2.9 BSD
2.10 BSD
2.11 BSD
4 BSD
V 8
4.2 BSD
SYS V
V 10
SCO
V.2
4.3 BSD
Plan 9
V.3
Ultrix
V.3.2
4.4 BSD
AIX
SUN OS
Mach
SVR 4
Solaris
OSF 1

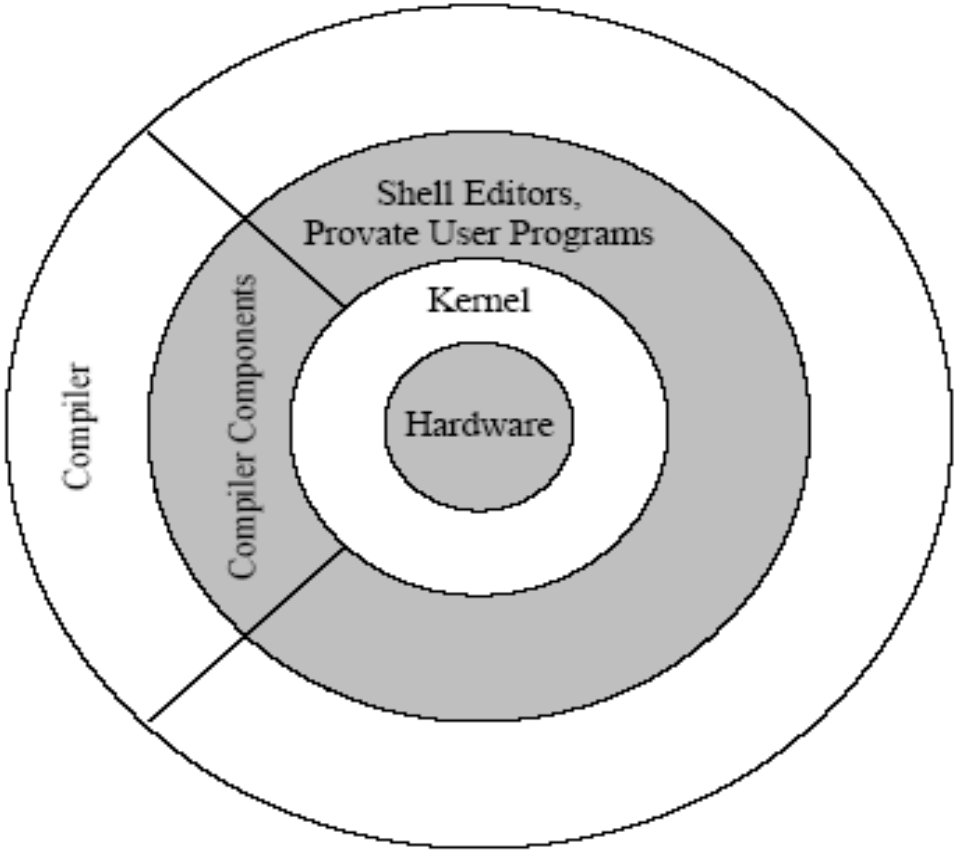
Estrutura do UNIX/LINUX

O Unix como o Linux é baseado em estrutura monolítica.



Estrutura do UNIX/LINUX

Visão da arquitetura do UNIX:



Estrutura do UNIX/LINUX

No kernel monolítico, todas as funcionalidades do kernel é visto como um bloco de código que executa em modo núcleo. Todos os componentes funcionais do kernel tem acesso a todas as rotinas e estruturas de dados interna.

No caso de mudanças em alguma parte do kernel, o mesmo deve ser reconstruído, suas bibliotecas re-linkadas e o sistema reiniciado para que as novas mudanças tenham efeito.

Como resultado de alguma modificação, tal como adicionar um novo driver de dispositivo ou alguma nova funcionalidade do sistema de arquivos, os sistema era reconstruído, dificultando muita esta tarefa.

Para resolver este problema, o Linux foi organizado como blocos relativamente independentes, chamados de **Loadable Modules**.

Os **Loadable Modules** tem duas principais características:

Estrutura do UNIX/LINUX

Dynamic linking: Um módulo do kernel pode ser “carregado” e linkado no kernel enquanto o kernel está na memória pronto e executando.

Stackable linking: Os módulos são arranjados em uma hierarquia, Módulos individuais servem como como bibliotecas quando eles são referenciados por módulos clientes

No Linux, o **Dynamic linking** facilita a tarefa de configuração do kernel. Um programa usuário, ou mesmo um usuário pode “carregar” ou “liberar” um módulo utilizando os comandos **insmod**, **modprobe** e **rmmod**.

Estrutura do UNIX/LINUX

A figura a seguir mostra a arquitetura em módulos: