

Sistemas Operacionais

Ciência da Computação

Robson Costa, Dr.
robson.costa@ifsc.edu.br

Instituto Federal de Santa Catarina
Campus Lages

Versão: 6 de fevereiro de 2023

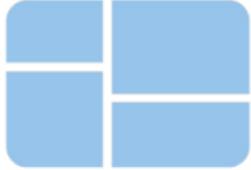
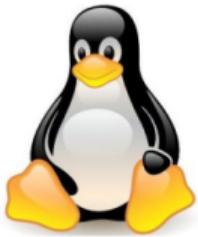
Sumário

Unidade 1 – Introdução aos Sistemas Operacionais

- 1.1 - Definições básicas e contextualização
- 1.2 - Por que estudar sistemas operacionais?
- 1.3 - História: surgimento dos SOs e principais características
- 1.4 - Principais conceitos de SOs
- 1.5 - Conclusões

Definições básicas e contextualização

O que é um Sistema Operacional?



iOS

Definições básicas e contextualização

O que é um Sistema Operacional?

- O Sistema Operacional (SO) pode ser considerado como uma camada de *software* alocada entre o *hardware* e os programas que executam as tarefas para os usuários;
- Ele é o responsável pelo acesso aos recursos possibilitando uma distribuição justa e eficiente do mesmo;



Definições básicas e contextualização

O que é um Sistema Operacional?

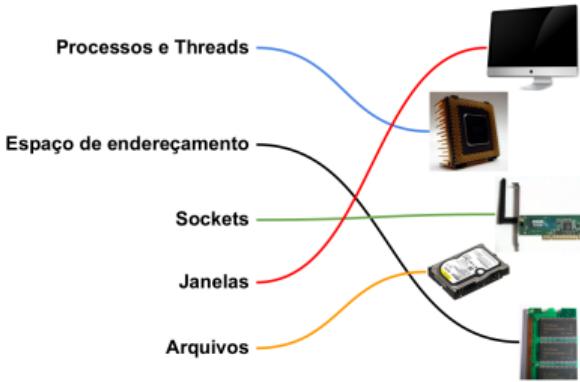


(Um sistema de computação - Tanenbaum e Woodhull, 2008)

Definições básicas e contextualização

O que é um Sistema Operacional?

- O SO procura tornar a utilização do computador, ao mesmo tempo, mais eficiente (+ com o mesmo *hardware*) e mais conveniente (redução de custos de desenvolvimento);
- Para cada tipo de *hardware* “gerido” pelo SO, haverá algum tipo de abstração oferecida para sua utilização;



Definições básicas e contextualização

O que é um Sistema Operacional?

- Para atingir os objetivos, o SO oferece diversos tipos de serviços (dependendo do seu foco), os básicos são:
 - Gerência de processos;
 - Gerência de memória;
 - Sistema de arquivos;
 - Gerência de entrada/saída;

Definições básicas e contextualização

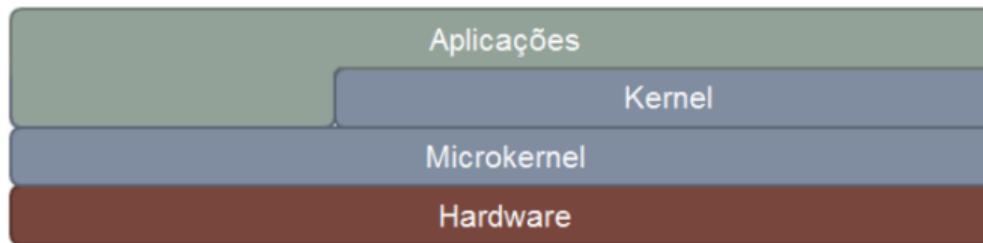
Visão do Usuário

- A arquitetura do SO corresponde à imagem que o mesmo tem do sistema;
- Essa imagem é definida pela interface através da qual o usuário acessa os serviços do sistema operacional;
- Essa interface, assim como a imagem, é formada pelas chamadas de sistema e pelos programas de sistema;
- As chamadas de sistema são a forma como os programas solicitam serviços ao SO;
- A parte do SO responsável por implementar as chamadas de sistema é normalmente chamada de **núcleo** ou ***kernel***;
- Os principais componentes do *kernel* de qualquer SO são:
 - Gerência de processador;
 - Gerência de memória;
 - Sistema de arquivos;
 - Gerência de entrada/saída;

Definições básicas e contextualização

Visão do Usuário

- Em função da complexidade de um *kernel* completo, muitos SOs são implementados em camadas;
- Primeiro um pequeno componente de *software* chamado **micronúcleo** ou **microkernel** implementa os serviços mais básicos do SO;
- Sobre o *microkernel*, usando os seus serviços, o *kernel* propriamente dito implementa os demais serviços;



Definições básicas e contextualização

Programas de Sistema

- Fornecidos juntamente com o SO, também chamados de utilitários;
- São programas normais executados fora do *kernel*;
- Implementam tarefas básicas para a utilização do SO e muitas vezes são confundidos com o mesmo (ex.: utilitários para manipulação de arquivos);
- Na década de 60, os compiladores também eram considerados programas de sistemas, mas com a expansão da indústria de microcomputadores, compiladores passaram a ser programas normais;
 - Pode-se utilizar o SO de um fornecedor e o compilador de outro;
- O mais importante programa de sistema é o interpretador de comandos, também conhecido como ***shell***;
- Semelhante ao *shell*, outro programa de sistema é a **interface gráfica de usuário (GUI – Graphical User Interface)**;

Definições básicas e contextualização

Visão de Projeto

- Neste caso, o mais importante é a organização interna do SO;
- O foco é obter maior eficiência na gestão dos recursos e também maior conveniência na interface de acesso à estes recursos;
- Normalmente a CPU executa programas de usuário, somente quando ocorre algum evento especial (chamada de sistema ou interrupção de periférico) o SO é ativado;
- Uma chamada de sistema corresponde a uma solicitação de serviço, que deve:
 - Verificar a sua legalidade;
 - Envia comandos ao controlador do periférico (quando aplicável) para executar uma tarefa e avisa o SO através de uma interrupção;
 - Quando uma interrupção ocorre, a CPU redireciona o fluxo de execução para a rotina de tratamento de interrupções;
 - O SO retorna o resultado ao programa origem;
- A unidade de gerência de memória (*MMU – Memory Management Unit*) constitui outro mecanismo importante para SO ditos “modernos”;

Por que estudar sistemas operacionais?

- **Alguns** precisarão projetar ou construir SOs, módulos ou bibliotecas para SOs;
- **Muitos** irão criar sistemas que utilizam conceitos centrais existentes em SOs, mesmo que desenvolvendo *hardware* e/ou *software*;
- **Todos** trabalharão direta ou indiretamente na criação de aplicações que utilizam os recursos dos SOs: quanto melhor você entende o SO, melhor você faz uso;

História - década de 1940

- Não existiam SOs;
- O programador era também o operador do computador;
- Existia uma planilha de horário onde cada programador poderia utilizar o computador, e este controlava completamente a máquina;
- A primeira modificação foi a criação de **operadores profissionais**;
- Neste caso, o programador entregava ao operador o seu ***job***;
- O ***job*** era formado pelos programas e seus dados a serem processados;

História - década de 1940

- Após a execução do *job*, o programador recebia uma listagem com a saída;
- Em caso de erro, o operador enviava uma cópia do conteúdo da memória;
- Isto diminuiu o atraso inerente da planilha de horário, mas o tempo de preparação (inserção dos cartões e/ou fitas) ainda continuava grande;
- Para otimizar isto, criou-se **sistema em lote (*batch*)**, onde programas com necessidades semelhantes eram organizados e executados serialmente;
- Mesmo assim a passagem de um *job* para outro era feita manualmente;

História - década de 1950

- Criação de **bibliotecas de sistema (*system libraries*)**;
 - Geram abstração de *hardware* flexibilizando o acesso aos periféricos;
 - Estas rotinas deram início ao conceito de **chamadas de sistema**;
- Criação dos **monitores de sistemas (*system monitor*)**;
 - Automatizam a execução de um programa após o término de outro;
- Para saber qual programa executar, criou-se cartões de controle;
 - Informam qual o próximo programa a ser executado;
 - Informam de qual usuário este programa pertence;
 - Isto é a origem da **linguagem de comando** utilizada atualmente;
- A partir do *system monitor* surgiu o conceito de **multiprogramação**;
- Anteriormente, apenas um programa era executado de cada vez;
 - Se este acessasse um periférico, a CPU ficaria parada esperando o seu retorno;
 - Isto resultava em um problema no tempo de execução de cada *job*;

História - década de 1950

- Com multiprogramação, enquanto um programa acessa um periférico, outro pode utilizar a CPU otimizando assim o seu tempo;
- Duas inovações de *hardware* permitiram isto:
 - O uso de **interrupções**, para avisar a CPU que a tarefa no periférico foi finalizada;
 - O surgimento de **discos magnéticos**, permitindo a leitura não sequencial dos *jobs*;
- O conceito de multiprogramação é essencial aos SOs e também é a origem da maioria dos temas básicos da área: gerência de memória, escalonamento de processos e sistema de arquivos;
- Como não havia mais a necessidade de se reunir *jobs* com características semelhantes, o termo *batch* passou a significar um sistema na qual não existe interação entre o usuário e a execução do programa;
 - Mais recentemente, este termo foi substituído por **execução em background**;

História - 1961

- Um grupo de pesquisa do MIT anuncia o desenvolvimento do CTSS (*Compatible Time-Sharing System*), o primeiro SO com compartilhamento de tempo;
 - Neste caso, além de multiprogramação cada usuário possui um terminal que lhes permitem acesso simultâneo ao sistema operacional;
 - Desta forma o usuário tem a impressão de possuir o computador apenas para os seus programas;
 - Isto é possível devido ao tempo de utilização reduzido da CPU nestes ambientes;
 - Um exemplo é um editor de texto, o tempo entre a digitação dos caracteres é suficiente para que a CPU execute milhares de instruções de máquina de outros programas residentes na memória principal;

História - 1964

- Baseado na ideia do CTSS surgiu o MULTICS (*MULTIplexed Information and Computing Service*);
 - Ideia: uma máquina enorme fornecendo poder de processamento para todos na região de Boston;
 - Introduziu muitas idéias embrionárias;
 - Sucesso misto: projetado para suportar centenas de usuários e pouco mais rápido que um Intel 80386;
 - Bell Labs e General Electric desistiram do projeto;
 - MIT manteve e fez o MULTICS funcionar: utilizado por cerca de 80 empresas, tais como GE, Ford e Agência de Segurança Americana;
 - O último MULTICS foi desligado em outubro de 2000, no Departamento de Defesa Canadense;

História - de 1965 até os tempos atuais

- **Em 1965** a IBM lança o OS/360, era extremamente complexo, possuía milhões de linhas em *assembly* (consequentemente milhares de *bugs*) e tinha suporte ao compartilhamento de tempo e um excelente suporte a discos (para a época);
- **Em 1969** Ken Thompson e Dennis Ritchie, pesquisadores da Bell Labs, criaram a primeira versão do Unix (*System V* da AT&T) escrito em *assembly*;
- **Em 1973** o Unix foi reescrito utilizando-se a linguagem de programação C;
- **Em 1978** foi lançado o BSD (*Berkeley Software Distribution*), que ainda era um *add-on* para o Unix 6;
- **Em 1981** a Microsoft lança o MS/DOS, um SO comprado da empresa Seattle Computer Products em 1980;
- **Em 1984** a Apple lança o SO Macintosh OS 1.0, o primeiro a ter uma interface gráfica totalmente incorporada ao sistema;

História - de 1965 até os tempos atuais

- **Em 1985** a Microsoft lança o MS-Windows 1.0, primeira tentativa da empresa no campo de sistemas operacionais com interface gráfica;
- **Em 1987** Andrew Tanenbaum, um professor de computação holandês, desenvolve um SO didático simplificado, mas respeitando a API do Unix, que foi batizado de Minix;
- **Ainda em 1987** a IBM e a Microsoft apresentam a primeira versão do OS/2, um SO multitarefa destinado a substituir o MS-DOS e o Windows. Posteriormente as empresas rompem o acordo, a Microsoft fica com o Windows e a IBM com o OS/2;
- **Em 1991** Linus Torvalds, um estudante de graduação finlandês, inicia o desenvolvimento do Linux, lança na rede Usenet o núcleo 0.01, logo abraçado por centenas de programadores ao redor do mundo todo;
- **Em 1993** a Microsoft lança o Windows NT, o primeiro SO 32 bits da empresa;
- **Ainda em 1993** é realizado o lançamento dos Unix de código aberto FreeBSD e NetBSD;

História - de 1965 até os tempos atuais

- **Em 2001** a Apple lança o MacOS X, um SO derivado da família Unix/BSD;
- **Em 2007** a Apple lança o iOS, um SO para dispositivos móveis como *smartphones* e *tablets*;
- **Em 2008** a Google lança o Android (adquirido por ela em 2005);
- **Em 2010** a Microsoft lança a sua versão de SO para dispositivos móveis, o Windows Phone;

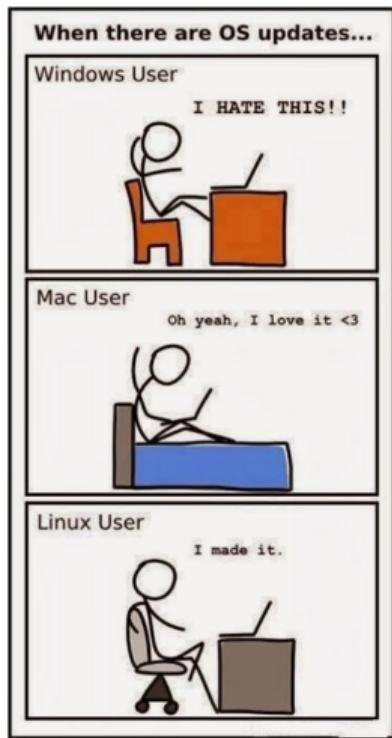
Competitividade...

Imagen de <http://8bitgnome.deviantart.com/art/Windows-vs-mac-vs-linux-Sp-131171690>

Windows vs **mac** vs **linux**

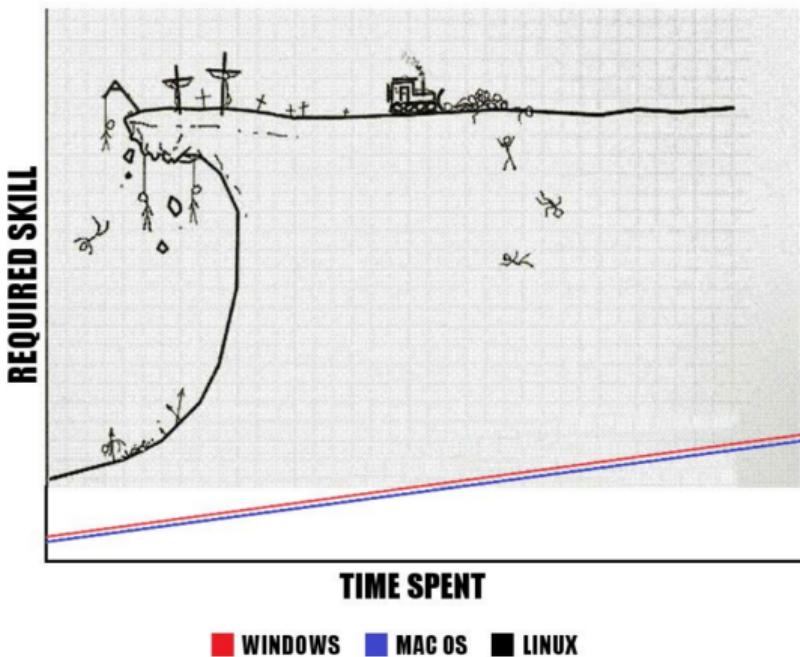


Atualizações...

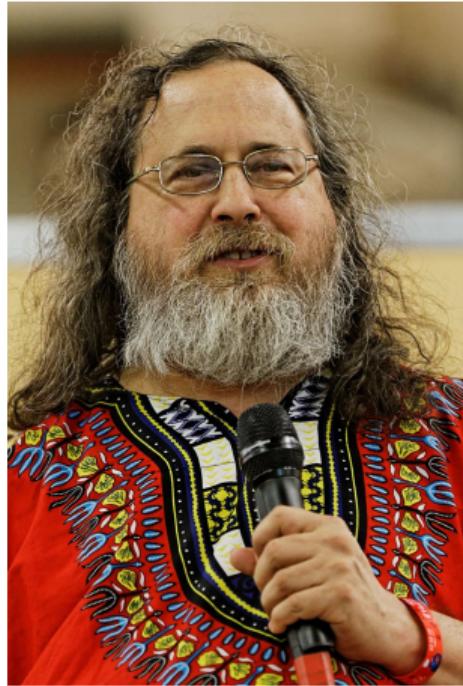


Curva de aprendizado...

OS LEARNING CURVE



Relação barba e SO (meio machista... mas tem uma explicação)



Principais conceitos de SOs - Processos

- Um **processo** é a execução de um programa: uma **instância de um programa**;
- Um programa pode ser visto como um conjunto de instruções passivas, armazenadas em um arquivo no computador;
- Um programa será carregado da memória secundária (ex. HD) para a memória principal (ex. RAM) onde também terá espaço para armazenar informações adicionais sobre a sua execução;
- Cada processo é associado a um **espaço de endereçamento**:
 - Definido por endereço mínimo e máximo (similar a um vetor);
 - Este espaço contém:
 - o programa executável;
 - dados do programa;
 - a pilha do processo;

Principais conceitos de SOs - Processos

- Do ponto de vista da memória RAM, podemos ter vários processos coexistindo ao mesmo tempo, desde que respeitado o espaço disponível nela;
- Do ponto de vista de um núcleo do processador, podemos ter apenas um único processo sendo executado, de fato, ao mesmo tempo;
- Além de instruções em memória, os processos geralmente precisam acessar:
 - arquivos;
 - dispositivos de entrada/saída;
 - interfaces de comunicação;
- Para utilizar cada um destes recursos, informações precisarão ser armazenadas na memória principal;

Principais conceitos de SOs - Arquivos

- O **arquivo** é uma **abstração para um conjunto de dados**;
- Arquivos são armazenados em dispositivos físicos, usualmente não voláteis;
 - Ex.: HD, SSD, CD, DVD, Blueray, Pendrive, Fita magnética;
- Da perspectiva do usuário, um arquivo é a menor unidade de armazenamento lógico secundário, ou seja, dados só existem na memória secundária através de arquivos;



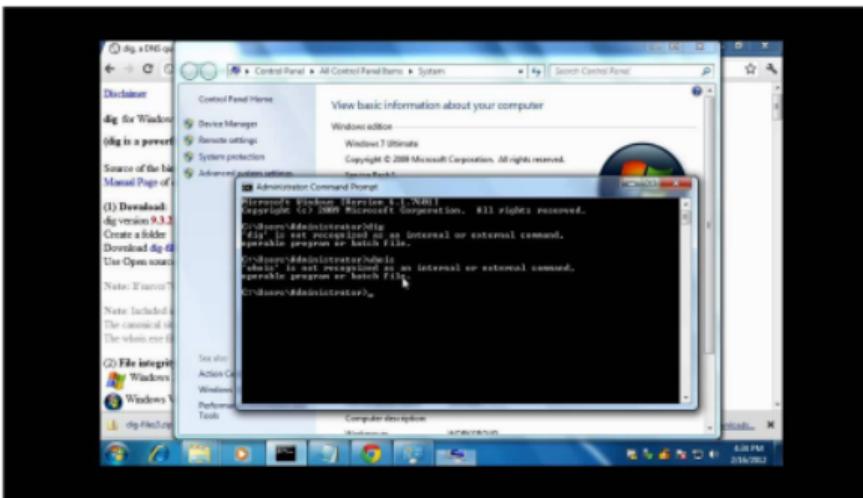
Principais conceitos de SOs - Arquivos

- **Silberschatz et al., 2013:**

- *“Normalmente, os arquivos representam programas e dados. Arquivos de dados podem ser numéricos, alfabéticos, alfanuméricicos ou binários. Os arquivos podem ter forma livre, como os arquivos de texto, ou podem ser formatados rigidamente. Em geral, um arquivo é uma sequência de bits, bytes, linhas ou registros, cujo significado é definido pelo criador do arquivo e pelo seu usuário. O conceito de arquivo é, portanto, extremamente geral.”*

Principais conceitos de SOs - Shell

- **Shell** é uma **interface** que disponibiliza interatividade entre o usuário e o sistema operacional;
- Pode ser em modo texto ou gráfico:
 - CLI (*Command Line Interface*);
 - GUI (*Graphical User Interface*);



Principais conceitos de SOs - Shell

Linha de Comando

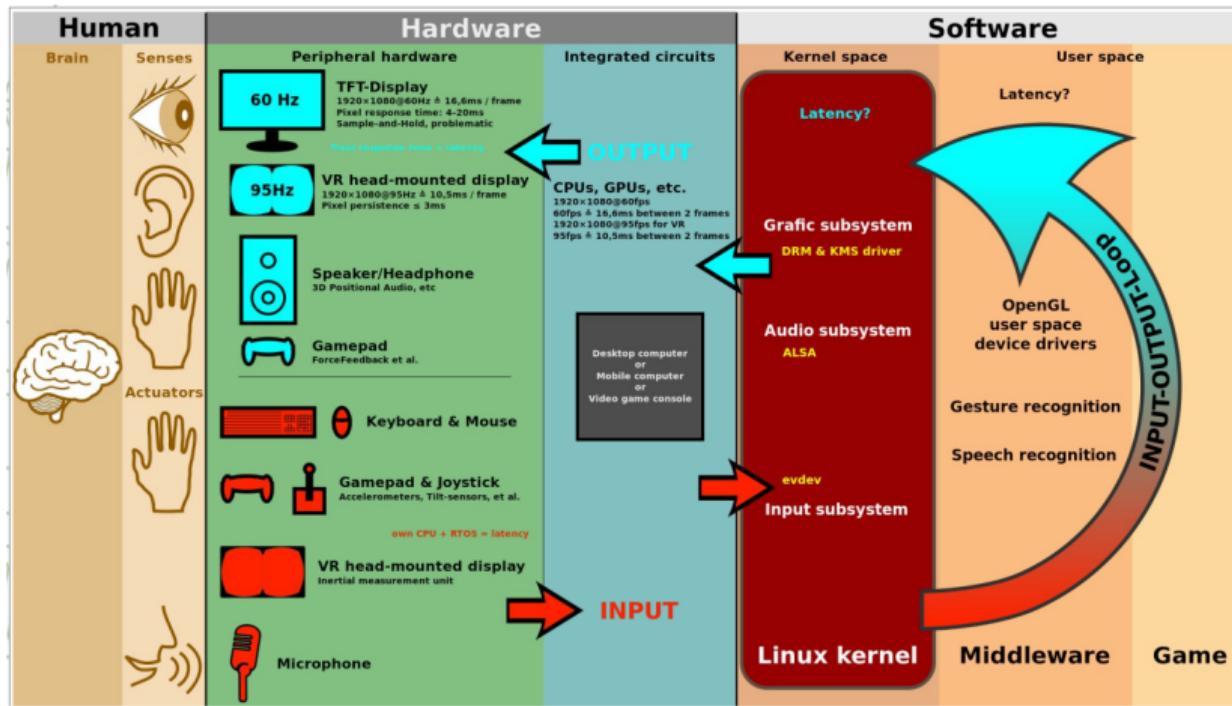
```
lame -m s -a --preset 64 --lowpass 3.4 --highpass 0.42 /Volumes/+10RAID2000/Users/localadmin/Music/iTunes/Media/Music/Andreas Illiger/Unknown/Album/01\ Tiny\ Wings\ Theme.mp3 out.mp3
ID3v2 found. Be aware that the ID3 tag is currently lost when transcoding.
LAME 3.99 64bits (http://lame.sf.net)
Autoconverting from stereo to mono. Setting encoding to mono mode.
Resampling: input 44.1 kHz output 8 kHz
Using polyphase highpass filter, transition band: 387 Hz - 484 Hz
Using polyphase lowpass filter, transition band: 3387 Hz - 3484 Hz
Encoding /Volumes/+10RAID2000/Users/localadmin/Music/iTunes/Media/Music/Andreas Illiger/Unknown Album/01 Tiny Wings Theme.mp3
to out.mp3
Encoding as 8 kHz single-ch MPEG-2.5 Layer III [2x] average 64 kbps qval=3
      Frame          | CPU time/estim | REAL time/estim | play/CPU |    ETA
      2966/2968 [100%] | 0:01/     0:01| 0:02/     0:02| 122.89x| 0:00
      8 [  2] *
      16 [   0]
      24 [   0]
      32 [   0]
      40 [   1] *
      48 [ 886] ****
      56 [2056] ****
*****
      64 [ 21] **
-----
-----
      kbps      mono %      long switch short %
      53.6      100.0       97.6    1.4    0.9
Writing LAME Tag...done
ReplayGain: -7.3dB
```

Principais conceitos de SOs - Shell

Interface Gráfica



Principais conceitos de SOs - Shell



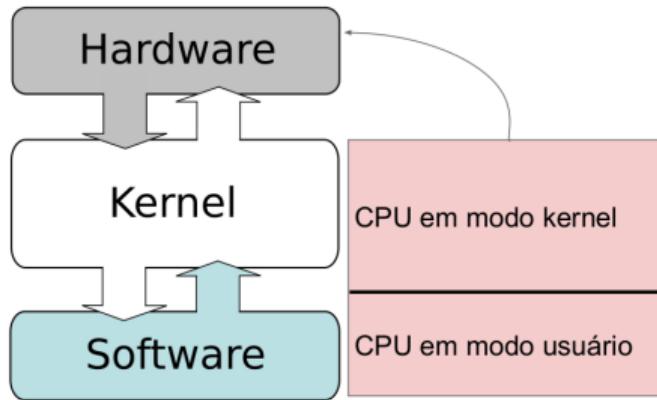
By Shmuel Csaba Otto Traian, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=31418026>

Principais conceitos de SOs - Chamadas de sistema

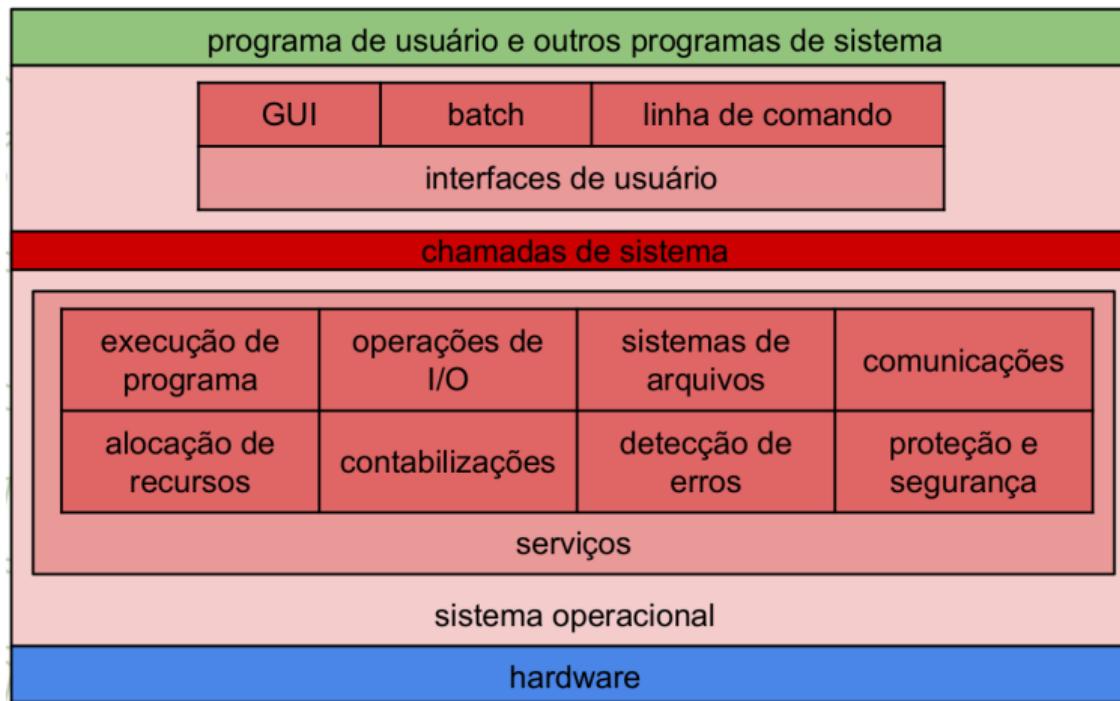
- Os programas precisam utilizar recursos tanto do SO quanto do *hardware*;
- A utilização destes recursos é feita através de solicitações conhecidas como **chamadas de sistema**;
- Chamadas de sistema podem ser vistas como **funções disponibilizadas pelo SO**, as quais são **invocadas pelo software**;
- As chamadas de sistema fornecem uma interface com os serviços e recursos disponibilizados pelo SO;
- São solicitações ao *kernel* do SO e geralmente estão disponíveis como rotinas implementadas em C, C++ ou Assembly;
- Acessadas/utilizadas através de uma API (*Application Programming Interface*);
 - As funcionalidades da API do SO vão além da realização de chamadas de sistema;

Principais conceitos de SOs - Chamadas de sistema

- Para entender chamadas de sistema, é interessante lembrar dois principais aspectos sobre SOs:
 - Eles mantêm conjuntos de serviços que provêm diferentes funcionalidades necessárias para execução dos demais aplicativos;
 - É necessária uma forma de comunicação entre os aplicativos executados em um SO e seus respectivos serviços;



Principais conceitos de SOs - Chamadas de sistema



Uma visão dos serviços do sistema operacional - Silberschatz et al, 2013

Principais conceitos de SOs - Chamadas de sistema

- As três APIs de SOs mais comuns são:
 - [Win32 API](#) - Windows API;
 - [POSIX API](#) - inclui “virtualmente” todas as versões do UNIX, Linux e Mac OS X;
 - o padrão POSIX (*Portable Operating System Interface*) surgiu na década de 70/80 como uma iniciativa para tornar os programas mais compatíveis entre vários sistemas operacionais;
 - definido pela família de normas [IEEE 1003](#);
 - [Java API](#) - para a máquina virtual Java;

Principais conceitos de SOs - Arquitetura de SOs

- Um sistema tão grande e complexo como um sistema operacional moderno deve ser construído cuidadosamente;
- Várias abordagens diferentes podem ser utilizadas, desde conceitos monolíticos até a modularização;
- A seguir, veremos um pouco sobre as seguintes arquiteturas de um SO:
 - Estrutura simples;
 - Abordagem em camadas;
 - *Microkernels*;
 - Módulos;

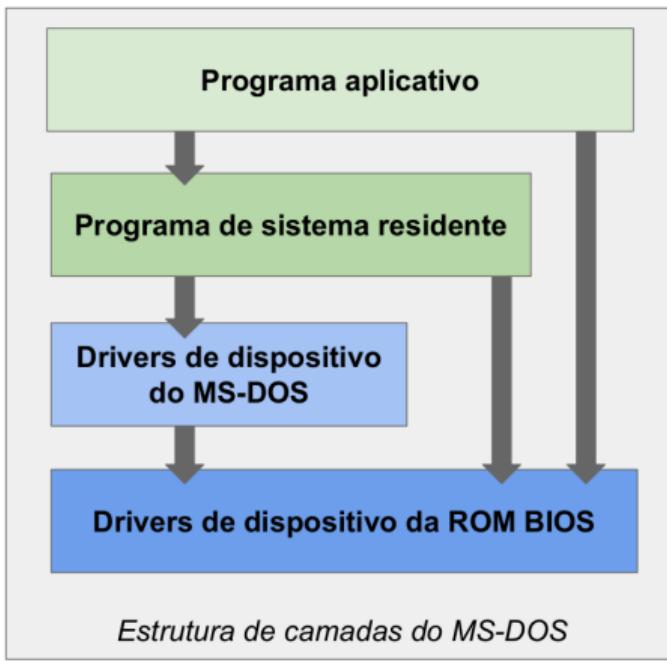
Principais conceitos de SOs - Arquitetura de SOs

Estrutura Simples

- Os primeiros SOs surgiram a partir de conceitos mais simples, e acabaram crescendo para além do seu escopo original (ex.: MS-DOS e Unix);
- **MS-DOS**
 - Foi projetado para fornecer o máximo de funcionalidade com o menor espaço;
 - Não foi planejado de forma modular;
 - O *software* aplicativo era capaz de realizar acessos de forma direta ao *hardware* (ex.: discos, I/O, tela);
 - Vale lembrar que quando o MS-DOS foi lançado, o processador para o qual ele foi escrito (Intel 8088) ainda não oferecia modo *dual* e proteção de *hardware*;
 - Deste modo, não havia outra forma de disponibilizar os recursos sem correr o risco de comprometer a segurança do sistema;

Principais conceitos de SOs - Arquitetura de SOs

Estrutura Simples

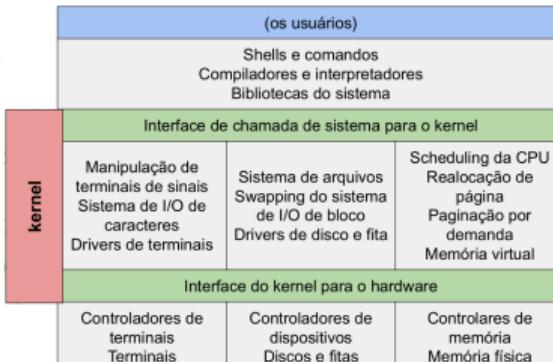


Principais conceitos de SOs - Arquitetura de SOs

Estrutura Simples

- **Unix original**

- Como no MS-DOS, era limitado pelas funcionalidades do *hardware*;
- Era composto por duas partes separadas:
 - **kernel**: separado em uma série de interfaces e *drivers* de dispositivos;
 - **programas do sistema**: compiladores, *shell* e bibliotecas;
- Estas estruturas **monolíticas** eram difíceis de se implementar e se manter;



Estrutura tradicional do sistema UNIX

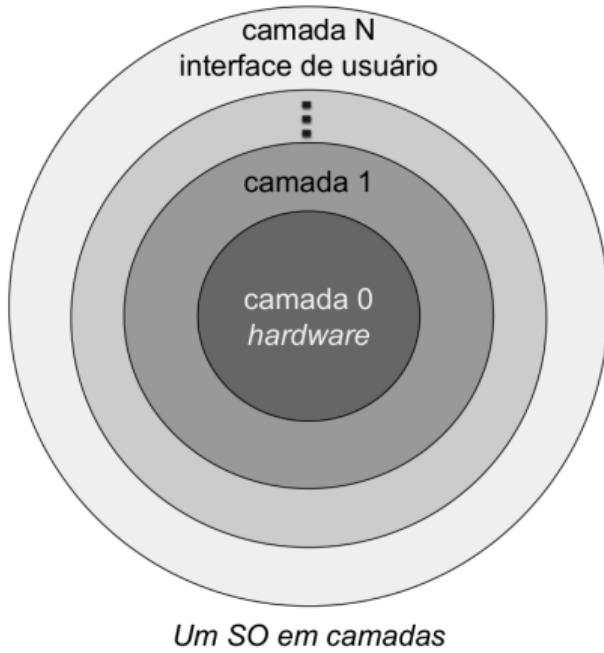
Principais conceitos de SOs - Arquitetura de SOs

Abordagem em Camadas

- Com a chegada do suporte de *hardware* adequado, surgiram novas possibilidades para os SOs;
- Os SOs passaram a serem divididos em partes menores e mais específicas;
- Esta modularidade trouxe inúmeras possibilidades, mas principalmente a criação de SOs que se adaptam melhor aos diferentes tipos de *hardware* e necessidades dos usuários e programadores;
- A abordagem por camadas permitia a divisão do SO em partes menores;
 - A ideia é imaginar o sistema como um todo como sendo composto por várias camadas. Cada camada serviria como acesso às camadas mais internas;
 - A camada inferior, chamada de camada 0 (zero), é o *hardware*;
 - A camada mais alta, chamada de camada N , é a interface com o usuário;

Principais conceitos de SOs - Arquitetura de SOs

Abordagem em Camadas



Principais conceitos de SOs - Arquitetura de SOs

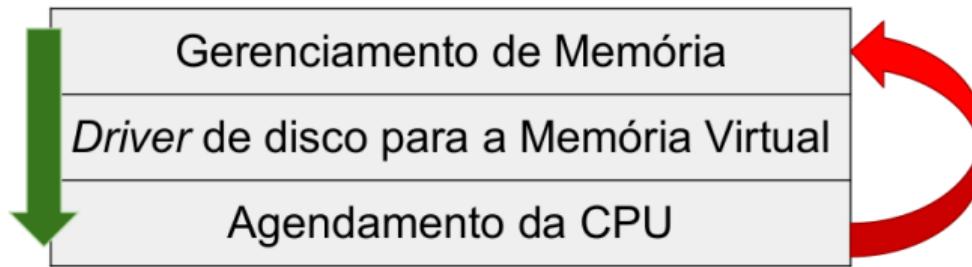
Abordagem em Camadas

- Cada camada é uma implementação de estruturas de dados e um conjunto de rotinas que podem ser chamadas por camadas de nível superior;
- Para implementar uma nova camada, só poderão ser utilizadas funcionalidades disponibilizadas pela camada imediatamente inferior à ela;
- A principal vantagem é a facilidade de construção e de depuração;
- Como principais desvantagens, podemos destacar:
 - **Definição apropriada para cada camada:** Um *driver* que será utilizado por uma camada, deve ser implementado em uma camada inferior (ex.: o driver de disco para memória virtual deve estar em uma camada inferior ao de gerenciamento da memória principal). O módulo agendador da CPU fica abaixo do driver de disco, mas ele mesmo pode precisar do gerenciamento de memória;

Principais conceitos de SOs - Arquitetura de SOs

Abordagem em Camadas

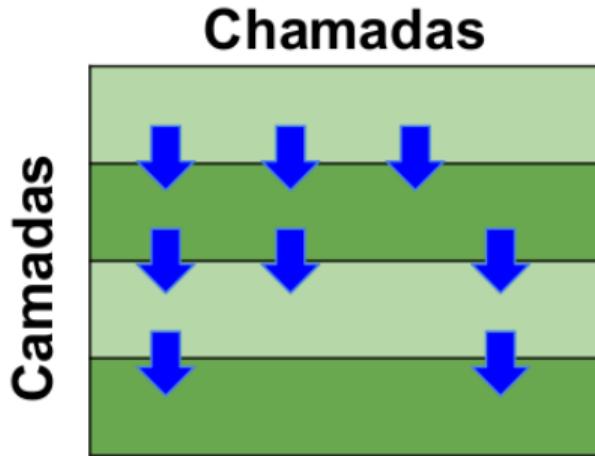
- Dificuldade na resolução de dependências entre as camadas



Principais conceitos de SOs - Arquitetura de SOs

Abordagem em Camadas

- Menor eficiência: Gera um *overhead* de chamadas de sistema com tantas transições entre diferentes camadas. Isto torna o sistema lento;



Principais conceitos de SOs - Arquitetura de SOs

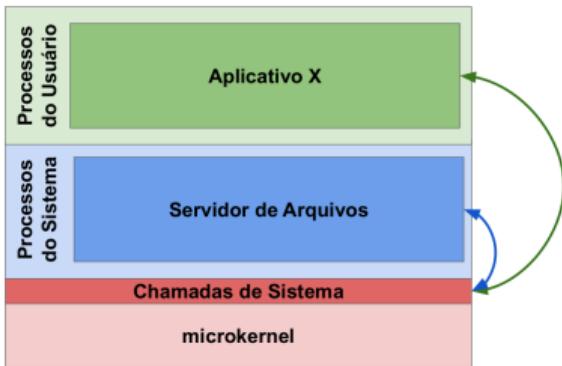
Microkernel

- Surgiu nos anos 80, na Universidade de Carnegie Mellon, para o SO Mach;
- A ideia básica é que o não essencial deve ser removido do *kernel* e implementado em programas de usuário e de sistema;
- Desta forma, o *kernel* fica pequeno, mas há pouco consenso sobre quais serviços devem ser mantidos nele;
- Normalmente o *microkernel* fornece:
 - Gerenciamento mínimo de processos;
 - Gerenciamento mínimo da memória;
 - Recurso de comunicação entre processos;

Principais conceitos de SOs - Arquitetura de SOs

Microkernel

- A principal tarefa do *kernel* passa a ser o fornecimento de recursos de comunicação entre os diferentes processos executados;
- Esta comunicação é feita através de um mecanismo de troca de mensagens;
 - Se um processo quiser acessar um arquivo, ele deve interagir com o processo “servidor de arquivos”. Esta interação nunca é feita de forma direta, mas sempre através do *kernel*;



Principais conceitos de SOs - Arquitetura de SOs

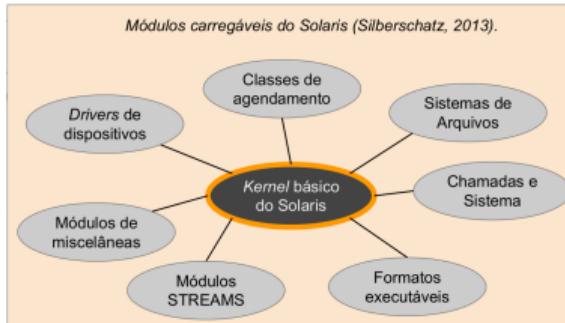
Microkernel

- Entre as vantagens do *microkernel* estão:
 - Facilidade de extensão do SO, uma vez que podem ser adicionados inúmeros serviços no espaço do usuário;
 - Raramente há a necessidade de se alterar o *kernel*;
 - O SO é mais facilmente portado para outras plataformas de *hardware*;
 - Maior segurança e confiabilidade uma vez que os serviços são executados como processos de usuário, ou seja, se algo falhar, o resto permanece funcionando;
- A principal desvantagem do *microkernel* é o fato de poder sofrer um comprometimento do seu desempenho devido ao *overhead* de mensagens e funções trocadas pelo sistema;
- Exemplos de SOs baseados em *microkernel*:
 - Tru64 UNIX (baseado no *kernel* Mach);
 - Mac OS X (também baseado no Mach);
 - QNX: SO de tempo real;

Principais conceitos de SOs - Arquitetura de SOs

Módulos

- Considerado como um dos melhores modelos;
- Usa técnicas de programação orientada a objetos para construir um *kernel* modular;
- O *kernel* tem apenas um conjunto de componentes básicos;
- A estratégia é carregar módulos dinamicamente ao próprio *kernel*;
- Basicamente, carrega 7 possíveis tipos de módulos:
 - Classes de agendamento, sistemas de arquivos, chamadas de sistema, formatos executáveis, módulos *streams*, miscelâneas e drivers de dispositivos;



Principais conceitos de SOs - Arquitetura de SOs

Módulos

- Novos módulos podem ser incorporados dinamicamente ao *kernel*, tanto para prover funcionalidades de *software* quanto para adicionar maior suporte a *hardware*;
- Parece com a abordagem por camadas, já que cada elemento pode incorporar dinamicamente novas partes, mas permite a comunicação direta entre as camadas que são diretamente associadas, diminuindo o *overhead* de mensagens;
- O *kernel*, em si, é uma abordagem de *microkernel*, pois contempla apenas funcionalidades básicas;
- O Mac OS X utiliza uma estrutura híbrida, com conceitos de módulos. Lembrando que este SO utiliza o *microkernel* Mach e o *kernel* BSD:
 - O Mach fornece gerenciamento de memória, suporte a chamadas de procedimento remotas e recursos de comunicação entre processos;
 - O BSD fornece suporte a conexão de rede, interface de linha de comando, sistema de arquivos e uma implementação de API POSIX;

Principais conceitos de SOs - Arquitetura de SOs

Módulos

- Vantagens:
 - O SO carregará apenas os módulos necessários;
 - O *kernel* é pequeno;
 - Ao precisar de um novo módulo, é só instalar;
- Desvantagens:
 - Pode perder estabilidade se algum módulo apresentar problemas;
 - Um módulo malicioso ou com falhas pode comprometer completamente a segurança do SO;

Conclusões

A problem has been detected and Windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS
PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

Conclusões

- Sistemas Operacionais (SOs):
 - formam uma peça fundamental na evolução tecnológica;
 - já evoluíram muito, mas ainda deverão evoluir muito mais;
 - existem em inúmeros “ sabores e aromas”, ou seja, para cada aplicação pode haver um SO que se adapte melhor;
- Processos são instâncias de programas;
- Arquivos são abstrações para conjuntos de dados;
- *Shell* é a interface do SO com o usuário;
- Chamadas de sistema permitem que os aplicativos façam solicitações para realizar tarefas que não possuem acesso direto;
- Para a implementação de SOs, vários tipos de arquitetura podem ser empregados em sua concepção, alterando drasticamente o seu funcionamento;