Bom dia, Boa Tarde, Boa Noite,

meu caro aluno, antes de iniciarmos nossa disciplina gostaria de me apresentar.

Me chamo Julio Cesar Pereira Lobtchenko, sou formado em Ciência da Computação pelo Centro Universitário da Grande Dourados – UNIGRAN, também sou formado em Ciências Biológicas (licenciatura) pela Universidade Federal da Grande Dourados – UFGD. Sou mestre em Biologia Geral também pela Universidade Federal da Grande Dourados – UFGD.

Você deve estar pensando, duas áreas bem parecidas né, kkk. São duas áreas magnificas e uma depende da outra, muitas coisas utilizadas na ciência da computação é baseado na biologia do ser humano, utilizando como referencia as ciências biológicas, e quanto a esta ultima é totalmente dependente da tecnologia e suas inovações, assim como o ser humano está cada vez mais totalmente dependente da tecnologia.

Assim precisamos entender o funcionamento destas tecnologias, e aí que entra nossa disciplina onde estudaremos um pouco sobre Redes de Computadores.

A seguir você irá ver explicações de cada tema, dicas, e reflexões no decorrer do conteúdo, para que sua aprendizagem seja significativa e você tenha condições de aplicar os conhecimentos adquiridos.

Aproveito a aqui para lembrar a vocês que sua dedicação é seu comprometimento são fundamentais. Leia o conteúdo com atenção, responda aos exercícios solicitados e aproveite as dicas e os recursos educacionais disponibilizados sobre os assuntos pertinentes à sua área de atuação.

Desejo a você um bom aprendizado!

O que estudaremos na nossa disciplina?

Estudaremos no decorrer da nossa matéria os seguintes conteúdos:

- Redes de Computadores (LAN, MAN, WAN);
- Modelo OSI;
- Camadas no modelo TCP/IP;
- Arquitetura de Redes TCP/IP;
- Protocolos;
- Interligação de redes;
- Planejamento e estruturação de uma rede;
- Princípios e Serviços de Sistemas Operacionais de Redes de Computadores;
- Conceitos de gerência de redes de computadores baseadas em TCP/IP.

E as avaliações?

Nossas avaliações serão em forma de exercícios e trabalhos, sendo assim não teremos prova. Entretanto cada exercício proposto será contabilizado na sua nota final.

Além disso teremos duas recuperações, sendo uma para o primeiro bimestre e outra para o segundo bimestre.

Agora damos início a nossa aula propriamente dita, como falei no começo desta aula chegamos em um ponto que não conseguimos viver sem computadores para realizar nossas tarefas do dia a dia. Dependemos de diversos programas rodarem para realizarem nossas tarefas diárias.

Porém, vamos voltar um passo e tentar entender o que é um computador. Segundo o Wikipedia:

> "Computador é uma máquina capaz de variados tipos de tratamento automático de informações ou processamento de dados. Um computador pode possuir inúmeros atributos, dentre eles armazenamento de dados, processamento de dados, cálculo em grande escala, desenho industrial, tratamento de imagens gráficas, realidade virtual, entretenimento e cultura."

Baseado na descrição acima, um computador então é um conjunto de dispositivos físicos que tem a capacidade de realizar processamento e armazenamento de dados. Para realizar nossas tarefas, precisamos de programas que digam para a máquina como processar e armazenar esses dados. O problema é que programas, em sua maioria, não conseguem atuar nesta parte física de forma direta.

Então, precisamos de uma camada intermediária para fazer controle e se comunicar tanto com a parte física quanto aos aplicativos. Quem faz este intermédio é o sistema operacional.

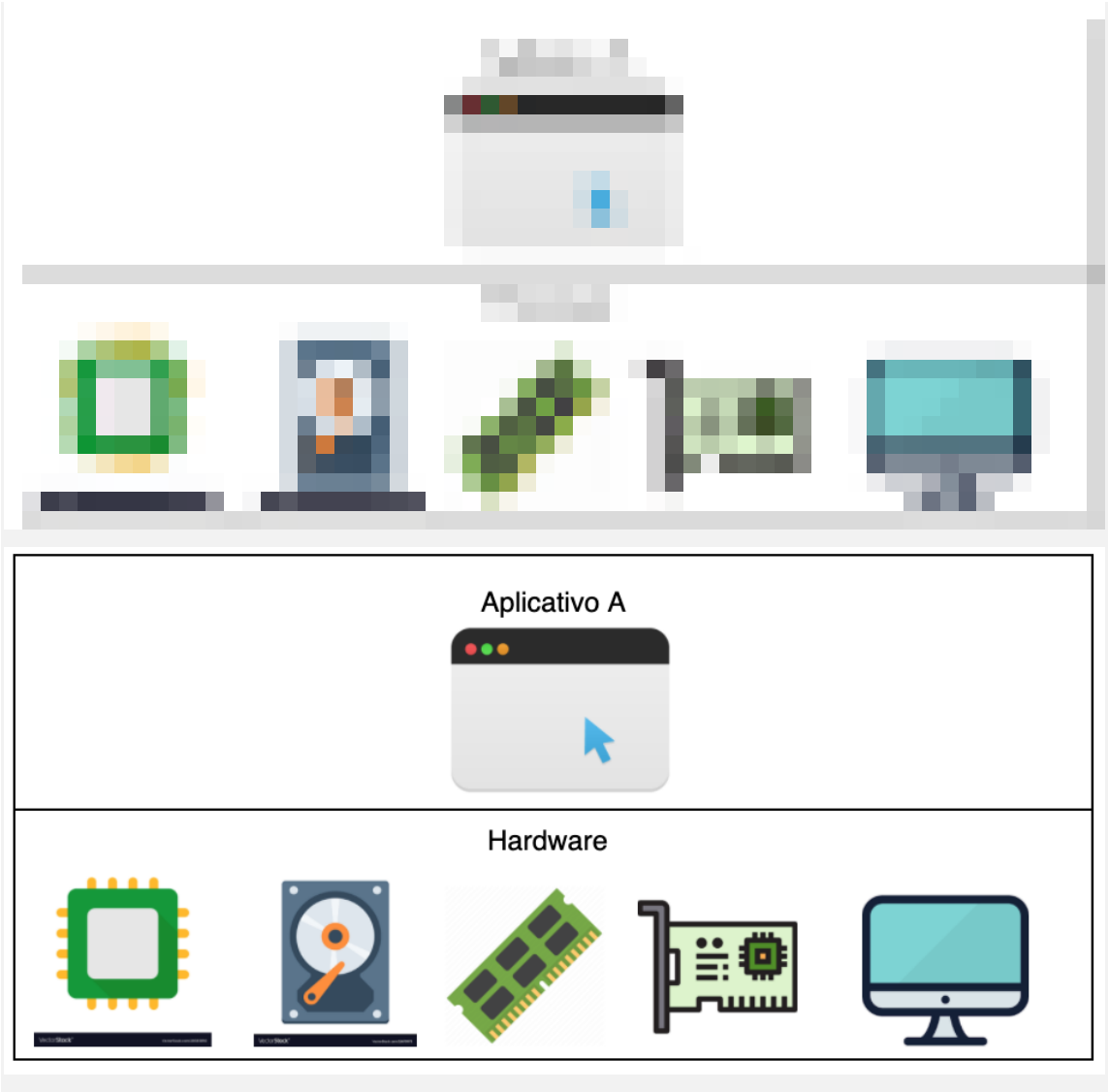
Tópicos que serão abordados

Esta aula falará sobre alguns tópicos introdutórios envolvendo sistemas operacionais:

- A Importância de Sistemas Operacionais
- Objetivos de um sistema operacional: fornece abstrações e gerenciar recursos
- Tipos de Sistemas Operacionais
- Elementos de um sistema operacional;
- Proteção do Sistema Operacional (system calls: chamadas de sistema)

A Importância de Sistemas Operacionais

Para entender a importância de sistemas operacionais, vamos imaginar que o mesmo nunca tenha existido. Logo, os aplicativos precisam rodar diretamente em cima do hardware.



Exemplo de cenário sem a existência de Sistemas Operacionais: O programa precisaria se comunicar diretamente com o hardware para realizar suas tarefas existem dois pontos principais que um desenvolvedor de aplicativo precisa se preocupar sem ter a presença de um sistema operacional.

Um deles é a complexidade de escrever código preocupando-se com toda a parte física da máquina.

Cada máquina possui uma arquitetura específica. Falar em arquitetura é o mesmo que dizer que uma máquina tem um conjunto de instruções (comandos que a CPU pode executar), organização da memória, dispositivos Entrada/Saída, dentre outros. Por exemplo, um dispositivo móvel não roda os mesmos comandos de CPU que um notebook.

Tudo isso teria de ser levado em conta quando o programador desenvolvesse uma aplicação. Mesmo com o uso de drivers fornecendo uma interface mais amigável para se comunicar com os dispositivos, seria bem complexo. Por exemplo, para escrever ou ler um bloco de dados de um disco rígido SATA, o desenvolvedor precisaria entender como funciona toda a interface de disco (sua documentação tem centenas de páginas).

Além disso, na maioria dos casos, queremos rodar muitos aplicativos na máquina ao mesmo tempo. Ao desenvolvermos aplicativos, teríamos de pensar numa forma de permitir que seu aplicativo rodasse em conjunto com t outros, compartilhando todos os recursos de hardware necessários para seu funcionamento: CPU, memória, dispositivos entrada/saída etc. Isso além de ser uma tarefa extremamente complexa, está muito propenso a erro. Abaixo seguem alguns dentre muitos pontos:

- Isolamento: como garantir que dois programas rodando não utilizem um mesmo espaço em memória?
- Compartilhamento: como será controlado a prioridade do acesso de recursos pra cada aplicação? Por exemplo, se um conjunto de programas solicitar a utilização da CPU ao mesmo tempo, como faremos pra dizer qual tem mais prioridade naquele momento?
- Políticas de Segurança: como garantir que certos aplicativos não tenham acesso privilegiado a alguns recursos?

Baseado nos pontos citados acima, ou seja, programas tivessem que lidar com tudo isso, a quantidade de programas desenvolvidos que temos hoje cairia drasticamente.

Pronto! São argumentos suficientes para entender que o sistema operacional é um componente essencial. Sem ele, a vida dos desenvolvedores e até usuários seria muito mais difícil!

Baseado no que vimos até agora, um sistema operacional tem dois papeis essenciais de forma resumida: fornecer abstrações para programas e gerenciar recursos. Vamos entendê-los a seguir.

Fornece Abstrações de Alto Nível

Vamos pensar, que tipo de relacionamento o sistema operacional tem com nossos aplicativos?

O sistema operacional precisa então saber lidar com toda a complexidade de Hardware, ou seja, isso precisa ser totalmente desacoplado dos aplicativos. Em outras palavras: os aplicativos devem rodar sem se preocupar com detalhes do que está acontecendo com o hardware.

Acima foi citado um exemplo de abstração (arquivos), porém existem várias outras abstrações de alto nível que valem mencionar: processos, memória, E/S dentre outros.
Todas essas abstrações são relativamente fáceis de implementar em um programa e tornam os problemas mais fáceis de solucionar.

Por exemplo, se quiser ler um arquivo, você não precisa se preocupar que tipo de arquivo que é, qual e como chegar a um determinado local da memória aonde está o arquivo, dentre outras questões que não envolvem necessariamente o objetivo do programa. O sistema operacional faz todo este trabalho "sujo".

Gerenciar Recursos

Como mencionado acima, um sistema operacional também precisa controlar o compartilhamento de como os dispositivos físicos tais como CPU, memória, dispositivos de E/S será feito com os programas. Em poucas palavras: controlar a multiplexação.

Vamos utilizar a CPU como exemplo. Existem dezenas de programas podem estar rodando na sua máquina: o navegador, serviços (como daemons) do próprio sistema operacional, algum aplicativo de música dentre outros. Em contrapartida, existem muito menos unidades de processamento na máquina. O sistema operacional precisa revezar a utilização das CPUs entre estes de uma forma justa e considerando a prioridade que cada processo tem em cada momento. Ou seja, a multiplexação é feita por tempo.

Outro exemplo similar que vale citar é a memória, a qual cada programa precisa ter seu próprio espaço privado para utilizar. O sistema operacional precisa manter estes isolados, sem que um afete o outro(um programa não pode ter acesso a um espaço de outro programa). Aqui, a multiplexação é feita por espaço.

Tipos de Sistemas Operacionais

A maioria das pessoas acreditam que o sistema operacional é apenas o que roda no seu computador pessoal, mas existe uma infinidade de tipos de sistemas operacionais. Vamos conhecer alguns dos principais tipos.

Sistemas Operacionais de grande porte

Apesar de ser considerado um tipo de sistema legado, muitas empresas (principalmente bancos) ainda utilizam esse tipo de sistema. Estes rodam em mainframes. Servem para realizar o processamento de muitas tarefas simples de forma simultânea, especialmente tarefas que envolvem grande entrada e saída e dados (E/S).

Esse tipo de sistema permite que muitos usuários se conectem ao mesmo tempo (timesharing) para realização de tarefas em batch, como transações bancárias. Um exemplo de sistema deste tipo é o IBM System/390. Sistemas Operacionais para Servidores Equivale a um desktop com "superpoderes" (capacidade bem maior). A ideia é fornecer serviços por meio da rede para múltiplos usuário, como impressão e compartilhamento de arquivos na rede.

Sistemas Operacionais para Desktops

Estes tipos de sistemas são os mais conhecidos por usuários. Basicamente, eles dão suporte a multiprogramação (vários programas compartilhando uma mesma CPU) oferecendo suporte as nossas tarefas do dia a dia.

Exemplos que valem citar: Windows 7/8 (Microsoft), Linux, FreeBSD, OS X. Sistemas Operacionais para portáteis se você tem um smartphone ou tablet, é esse tipo de sistema que roda nele. Atualmente, apesar de outras opções, os principais são: Android Google e o iOS, que gerenciam CPUs, memória, sensores, câmera dentre outros em favor dos muitos apps que instalamos nele.

Sistemas Operacionais embarcados

Estes geralmente rodam em dispositivos que diferentemente dos citados até agora, possuem geralmente muito poucas funcionalidades, somente o necessário para o dispositivo (como TVs, microondas dentre outros) em si.

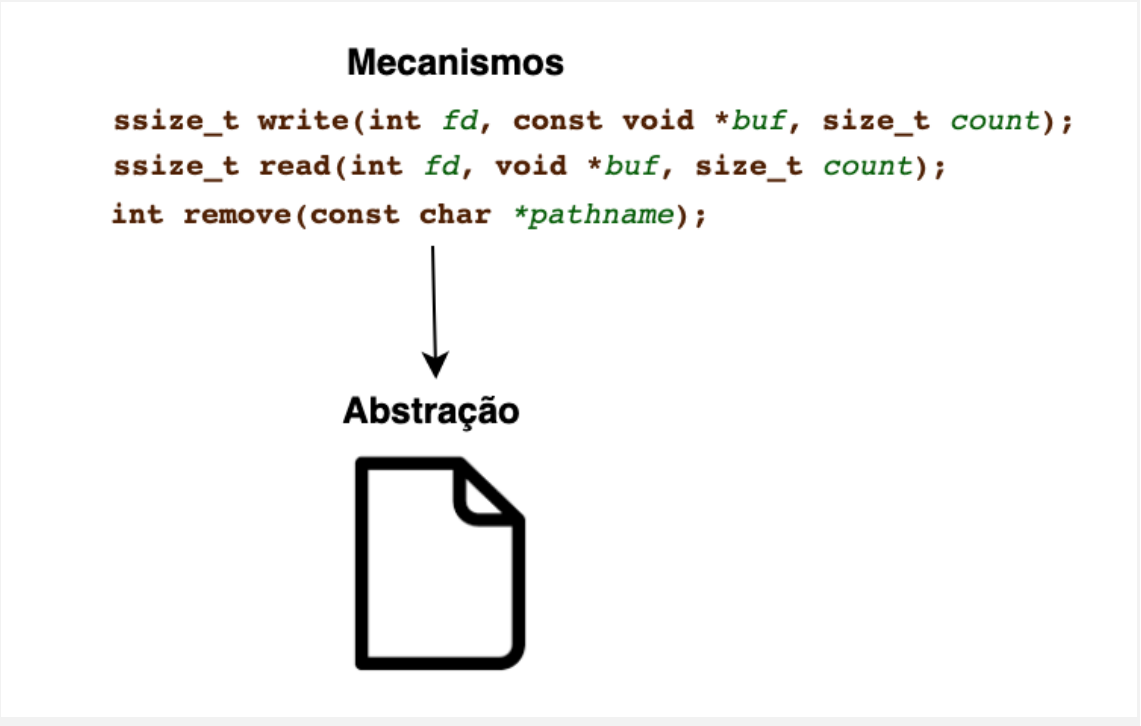
Isso implica que nenhum software adicional terceiro pode ser instalado nele, pois são gravados em um tipo de memória que permite a escrita somente uma vez (memória ROM). Alguns exemplos que valem citar: Embedded Linux, QNX e Vx Works.

Elementos de um Sistema Operacional

Para conseguir fazer os papeis que citamos acima, o sistema operacional precisa implementar basicamente três elementos que interagem entre si: abstrações, mecanismos e políticas.

O sistema operacional possui suporte a abstrações de alto nível, no qual mecanismos executam em cima destas. Além disso, deve-se definir a forma de como os mecanismos serão utilizados por meio de políticas.

Vamos ver um exemplo mais concreto de como isso funciona na prática. Um exemplo de abstração é um arquivo. O arquivo é uma abstração de uma sequência de conteúdo armazenado em algum lugar na memória, contendo informações para o sistema operacional ou aplicativo. Não importa o tipo de arquivo (imagem, documento, texto e dentre outros).



Exemplo de como funciona a relação entre mecanismos e abstrações: Os mecanismos realizam operações em cima de abstrações. Em cima da abstração de arquivo, podemos fazer algumas ações como ler, escrever e apagar seu conteúdo. A essas

operações denominamos mecanismos. Vale ressaltar novamente: esses mecanismos rodam em cima de abstrações, logo não importam os detalhes como tipo de arquivo.

Porém, não faz sentido que todo usuário ou programa possa acessar todos os arquivos do sistema operacional. Precisamos ter um nível de segurança e proteção e quando fizer alguma operação em um arquivo, verificar se o usuário ou aplicativo tem as devidas permissões para tal. Este é um exemplo de política. Em outras palavras: a política define como esses mecanismos atuarão.

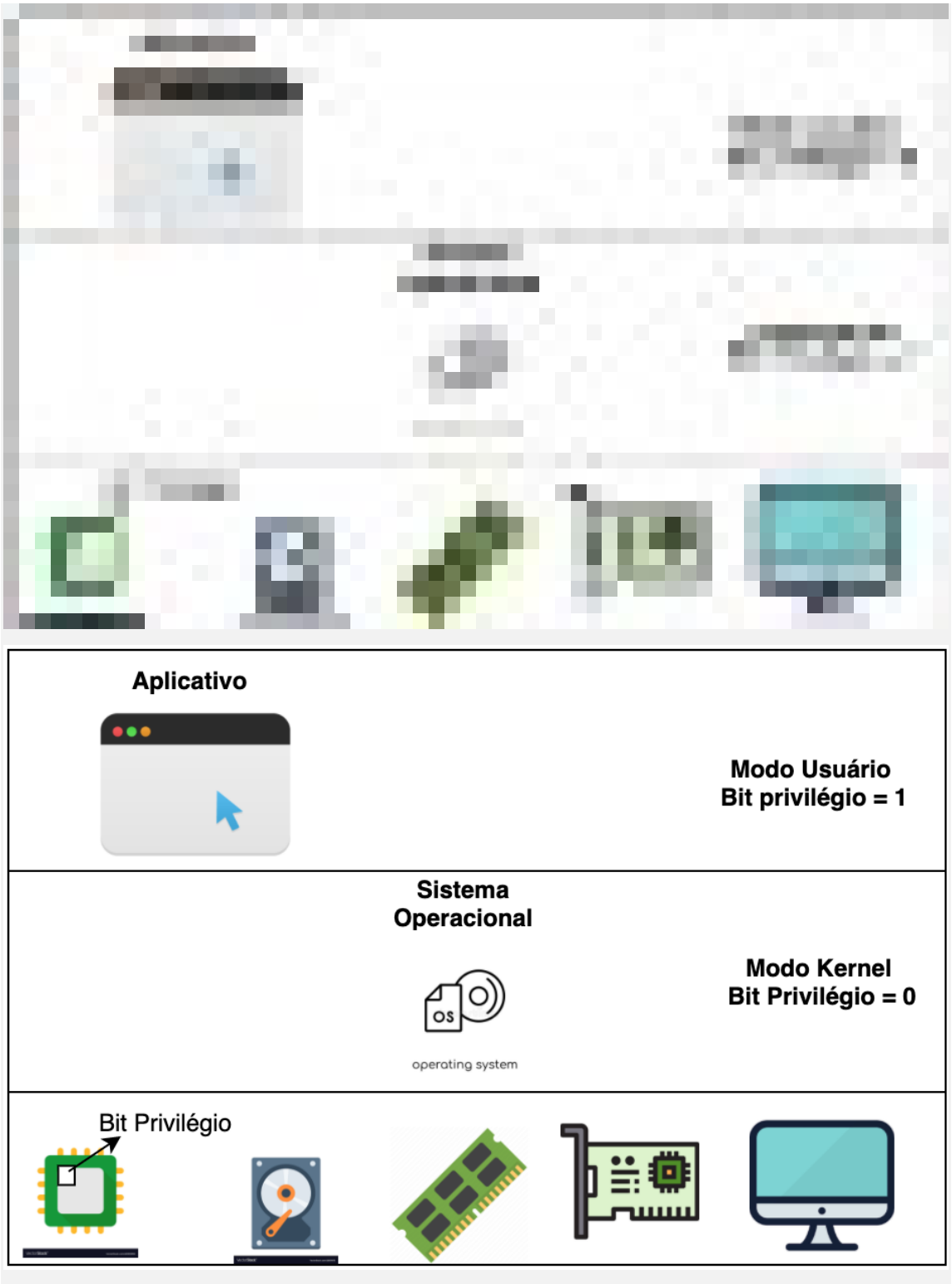
Proteção

Plataformas de computadores em geral dão suporte para ao menos dois modos: kernel e usuário. Isto pode ser feito, por exemplo, por um bit de controle fica dentro de um registrador do processador.

Observação: Registrador é um tipo de memória, porém com acesso bem mais rápido do que uma memória principal (RAM), sem delay para buscar os dados. Isso porque fica localizada dentro da CPU e é feita do mesmo material.

Modo Kernel/Supervisor (privilegiado): Neste modo, existe acesso irrestrito de utilização de recursos de hardware, podendo executar qualquer instrução. Como o kernel do sistema operacional é a camada que se comunica com o hardware, o mesmo opera neste modo. Somente algumas aplicações críticas podem ser exceções como drivers de dispositivos, por exemplo.

Modo Usuário (não privilegiado): Neste modo, a permissão ao hardware é bem restrita, só podendo executar alguns tipos de instruções. Os programas e até mesmo alguns aplicativos que auxiliam as tarefas do sistema operacional rodam neste modo.



Proteção: Sistema Operacional tem privilégios para executar quaisquer instruções por estar em modo Kernel

Porém, programas em algum momento precisam fazer operações, mesmo que indiretamente no hardware. Seguem abaixo alguns exemplos:

- Alocar memória para algum objeto;
- Ler/Escrever em um arquivo;
- Enviar dados para alguma máquina remota.

Logo, precisamos de uma interface simples para realizar essas ações. Para isso, os sistemas operacionais disponibilizam chamadas de sistema.

Chamadas de Sistema

Como mencionado, em geral, somente o sistema operacional e alguns programas críticos talvez, possuem permissão para executar informações privilegiadas no hardware.

Porém, o sistema operacional precisa fornecer uma camada para que os programas consigam realizar certas operações, conforme descrito acima.

Para isso, o sistema operacional fornece chamadas de sistema (system calls / syscalls), a qual permite que o mesmo cumpra seu objetivo de fornecer umas abstrações mais simples para os programas de usuário.

Elas são geralmente escritas em linguagem C ou de máquina e funcionam da mesma maneira ao chamar uma função dentro de um programa, porém com uma diferença:

A chamada de sistema precisa entrar no modo Kernel para ser executada, enquanto uma rotina dentro do programa não precisa.

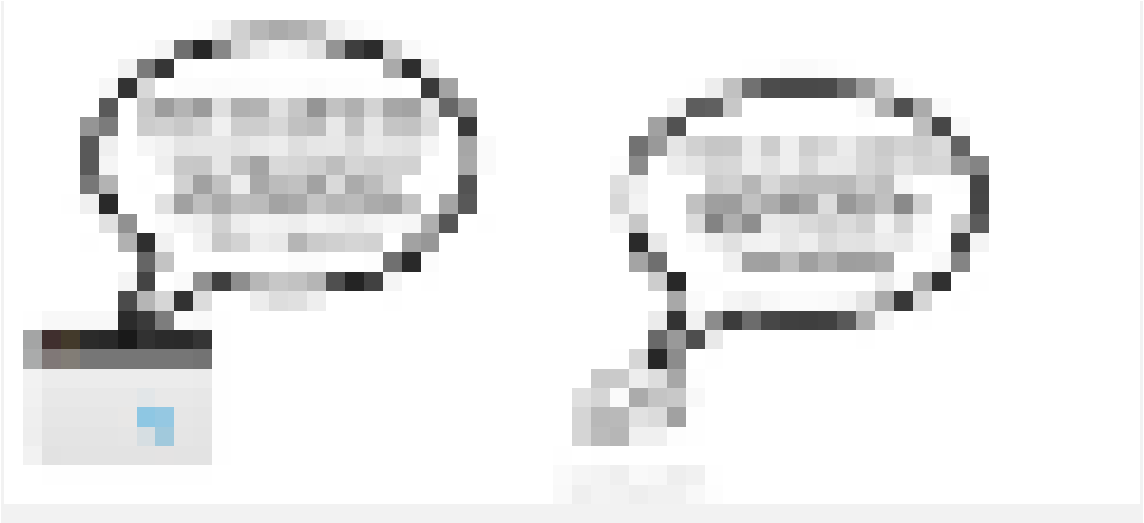
Porém, lembre-se: os programas rodam em modo Usuário. Como é magicamente feita essa alternância do modo Usuário para o modo Kernel?

Isso será explicado a seguir, mas antes vale citar alguns exemplos de chamadas de sistemas comuns em sistemas operacionais.

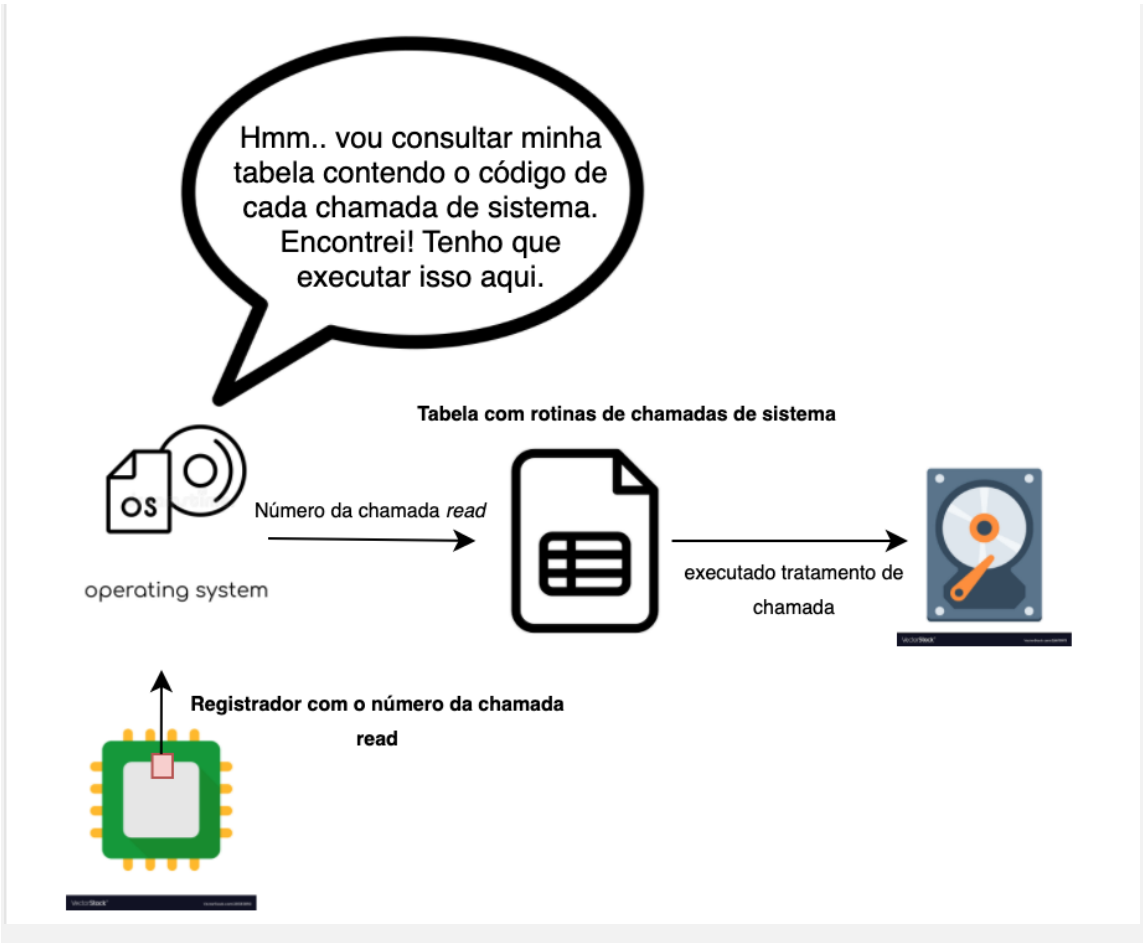
Observação: não vamos entrar em detalhes sobre quais parâmetros devem ser passados e o nome da rotina pode variar para cada sistema operacional. A ideia é mostrar que essas operações servem como mecanismos para operar diversas abstrações do sistema (conforme explicamos acima), tais como: processos, arquivos, dentre outros.

- read(): Lê dados de um arquivo.
- write(): Escreve dados em um arquivo.
- open(): Abre um arquivo (leitura/escrita)
- close(): Fecha um arquivo aberto.
- fork(): Cria um processo filho com as mesmas informações do pai.
- exit(): Sai do programa.
- mkdir(): Cria um novo diretório

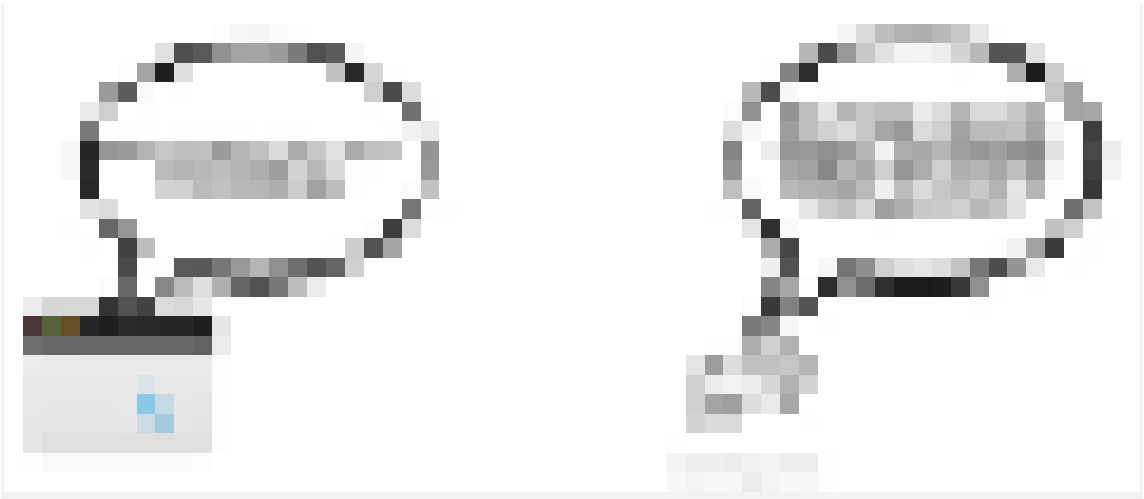
Vamos entender o que acontece por trás dos panos, inicialmente com a ilustração abaixo para simplificar a ideia e depois cada passo descrito mais detalhadamente.

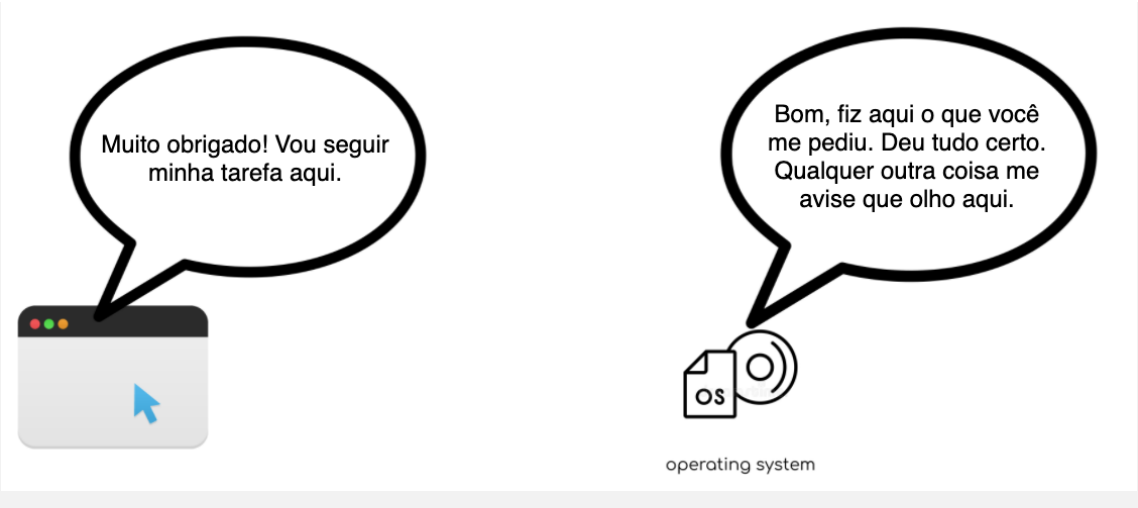


Exemplo do que acontece em um chamada de sistema: O programa faz a chamada para biblioteca, com os parâmetros necessários para mesma, similar ao que ocorre quando chamamos um método em nosso programa. Sistema Operacional entra em modo núcleo. Aqui também o sistema operacional coloca o número da chamada de sistema em um registrador da CPU.



Exemplo do que acontece em um chamada de sistema: O sistema operacional, operando ainda em modo Kernel, faz um despacho para o tratador correto (no caso a rotina) adequada baseado no número da chamada read() na tabela.





Exemplo do que acontece em um chamada de sistema: Por fim, a chamada read retorna o controle para o programa em modo usuário.

1. O programa faz uma chamada de sistema chamando uma biblioteca. No exemplo, utilizamos a chamada read, que serve para ler um arquivo. Os parâmetros são inicialmente empilhados e aí sim de fato, a chamada de sistema é feita.
2. Isso gera uma instrução TRAP (armadilha), que faz com o que entre em modo Kernel, passando o controle para o sistema operacional poder decidir o que fazer.
3. O sistema operacional empilha os parâmetros (como a chamada de uma função de um programa) "Deixa ver… o que esse processo quer?"
4. Se tudo estiver correto (caso não esteja, o programa pode ser terminado, caso por exemplo, tente executar algo ilegal) é feito o despacho: com o número da chamada, o sistema procura em uma tabela pelo tratamento de rotina adequado.
5. É feito cocada número de chamada sistema contendo qual rotina (geralmente em linguagem de máquina) deve chamar.
6. Quando a rotina terminar se ser executada, o sistema operacional retorna o resultado conforme uma chamada de função faz.
7. Por fim, o controle é devolvido ao programa e o mesmo precisa limpar a pilha (desempilhar os parâmetros empilhados do passo 1) após a chamada e seguir a execução do programa.

O exemplo acima foi com o exemplo da chamada read, mas os passos são similares as outras chamadas de sistema.

Conclusão

Um sistema operacional é indispensável nos dias de hoje para ser o intermediador entre os recursos físicos e programas de um computador. O mesmo tem dois objetivos principais: gerenciar os recursos físicos, os compartilhando com as aplicações e fornece abstrações amigáveis para estas poderem realizar suas tarefas de forma simplificada.

Para conseguir atingir estes objetivos, o sistema operacional atua em conjunto com três elementos: abstrações, mecanismos que rodam em cima destas abstrações e políticas para de que forma estes mecanismos serão utilizados.

Programas rodam em modo usuário, que possui acesso restrito a diversas operações. Para isso o sistema operacional, que opera em modo Kernel e possui acesso total ao operações envolvendo o hardware, fornece uma interface amigável para programas solicitarem que estas operações sejam realizadas pelo sistema operacional. Essas são denominadas chamadas de sistema.