Laboratório de Fundamentos em TIC

Introdução a Sistemas Operacionais e Linux

Prof. Gabriel Resende Machado

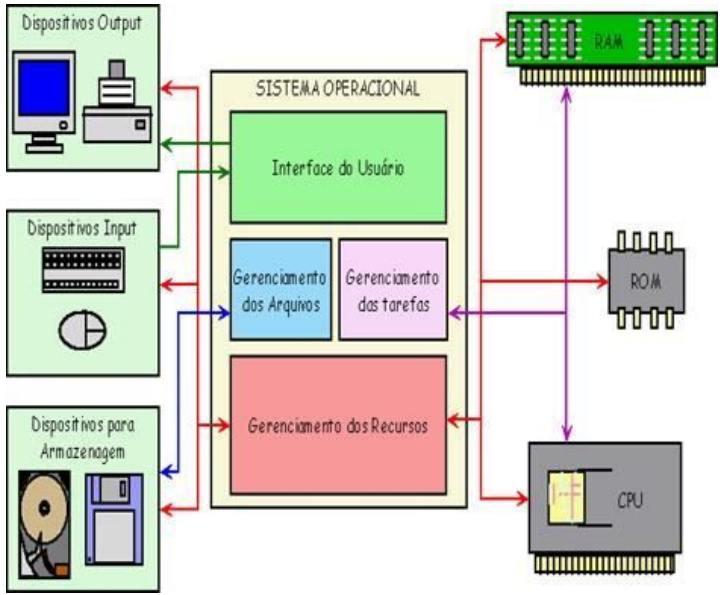
gabrielmachado@unifeso.edu.br

https://www.linkedin.com/in/machadogabriel

https://github.com/UNIFESO-Gabriel/fundamentos-em-tic

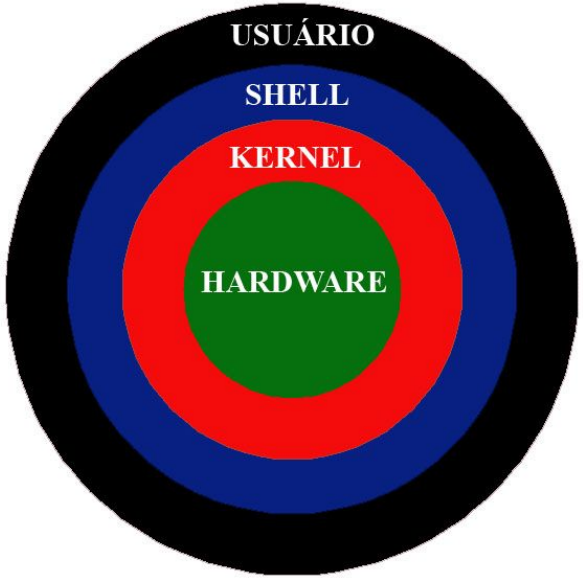
O que é um Sistema Operacional?

- Um Sistema Operacional (S.O.) é um programa que conversa diretamente com o *hardware* do computador e gerencia automaticamente aplicações executadas pelo usuário ou sistema;
- Sem o S.O., programadores e usuários precisariam levar em consideração o funcionamento de todos os componentes de *hardware* para executar suas tarefas;
- Tanenbaum (https://amzn.to/4c5NEPw) justifica que o sistema operacional surgiu para intermediar o diálogo entre a máquina e o usuário.



Componentes do Sistema Operacional

- Os componentes de um Sistema Operacional (S.O.) servem a várias funções cruciais que garantem o funcionamento eficiente, seguro e gerenciável do sistema;
- **O *Kernel*** orquestra grande parte desses componentes, de forma que trabalhem juntos para gerenciar recursos de *hardware* e *software*, fornecer serviços aos usuários e aplicativos, e garantir a segurança e estabilidade do sistema.



Principais Componentes de um S.O. (I)

- **Gerenciamento de Processos:** controlado pelo *Kernel*, envolve a criação, execução e terminação de processos (programas em execução), bem como a manutenção do estado dos processos em um sistema multitarefa;
- O gerenciamento de processos realizado pelo *Kernel* decide quais processos serão executados;
- Cada um dos processos em execução pode entrar e sair várias vezes do processador em um mesmo segundo, dando lugar a outro processo. O *Kernel* é responsável por decidir que processos serão alocados no processador;
- Como a alternância entre processos ocorre de forma muito rápida, um programa pode continuar sendo executado mesmo não estando no processador. **Os principais modos de acesso são:**
    - **Modo de usuário:** o modo de usuário é considerado um modo não privilegiado. Todos os *softwares* desse modo devem fazer requisições ao *Kernel* para poder executar instruções privilegiadas como, por exemplo, a criação de processos;
    - **Modo *Kernel*:** é considerado privilegiado, pois tem acesso a todo o computador. Quando a CPU está em modo *Kernel*, isso indica que ela está executando um *software* confiável e que está apta a executar quaisquer instruções.

Principais Componentes de um S.O. (I)

- **Outras responsabilidades do Gerenciamento de Processos**:
  - **Criação e Destruição de Processos:** O S.O. cria novos processos para executar programas e termina processos que concluíram sua execução ou que precisam ser interrompidos;
  - **Estados de Processos:** Gerencia os estados de processos (novo, pronto, executando, esperando, terminado) e transições entre esses estados;
  - **Escalonamento de Processos:** Decide qual processo deve ser executado pela CPU em um dado momento, usando algoritmos de escalonamento como *Round-Robin*, Prioridade, *First-Come First-Served* (FCFS), etc.;
  - **Troca de Contexto:** Alterna entre processos, salvando o estado do processo atual e restaurando o estado do próximo processo a ser executado;
  - **Comunicação entre Processos (IPC):** Facilita a comunicação e sincronização entre processos usando mecanismos como pipes, filas de mensagens, semáforos e memória compartilhada;
  - ***Deadlock* e *Starvation*:** Gerencia e resolve situações de *deadlock* (onde processos ficam esperando indefinidamente por recursos) e *starvation* (onde processos nunca recebem os recursos necessários).

Principais Componentes de um S.O. (II)

- **Gerenciamento de Memória:** Também controlado pelo *Kernel,* o gerenciamento de memória é responsável por controlar e coordenar o uso da memória principal (RAM) do sistema. Principais responsabilidades:
  - **Alocação e Liberação de Memória:** Distribui memória para processos quando necessário e libera memória quando não é mais necessária;
  - **Memória Virtual:** Implementa técnicas como paginação e segmentação para expandir a memória disponível além da memória física, permitindo que programas usem mais memória do que a fisicamente disponível;
  - **Proteção de Memória:** Garante que um processo não acesse a memória alocada para outro processo, prevenindo erros e problemas de segurança;
  - **Gerenciamento de Cache:** Utiliza a memória cache para armazenar temporariamente dados frequentemente acessados, melhorando o desempenho do sistema;
  - ***Swapping*:** Move processos entre a memória principal e o disco (*swap space*) para liberar memória quando necessário.

Principais Componentes de um S.O. (III)

- **Sistema de Arquivos:** Sistema de arquivos organiza e armazena dados em dispositivos de armazenamento, proporcionando uma maneira estruturada de acessar e gerenciar arquivos. Principais responsabilidades:
  - **Organização de Arquivos:** Define como os arquivos são armazenados, organizados e gerenciados no disco, usando estruturas como diretórios e subdiretórios;
  - **Controle de Acesso:** Define e aplica permissões de acesso a arquivos e diretórios, determinando quem pode ler, escrever ou executar um arquivo;
  - **Operações de Arquivo:** Facilita operações como criação, leitura, escrita, modificação e deleção de arquivos;
  - **Gerenciamento de Espaço em Disco:** Acompanha a utilização do espaço em disco, alocando e liberando espaço conforme necessário;
  - **Integridade e Recuperação de Dados:** Implementa mecanismos para garantir a integridade dos dados e fornecer ferramentas de recuperação em caso de falhas.
  - **Sistemas de arquivos:** S.O.s podem trabalhar com diferentes métodos de organização e armazenamento de dados em dispositivos de armazenamento, como discos rígidos, SSDs e pendrives. Os mais comuns são FAT32 (compatível, mas limitado em desempenho e tamanho), NTFS (Windows) e EXT4 (Linux).

Principais Componentes de um S.O. (IV)

- **Gerenciamento de Dispositivos:** gerenciado pelo *Kernel,* controla e coordena o uso de dispositivos de *hardware*, como impressoras, discos rígidos, teclados, etc. Principais responsabilidades:
  - ***Drivers* de Dispositivo:** Carrega e gerencia drivers que permitem ao S.O. comunicar-se com dispositivos de hardware específicos;
  - **Operações de Entrada/Saída (I/O):** Gerencia as operações de leitura e escrita em dispositivos, garantindo que os dados sejam transferidos de maneira eficiente e correta;
  - ***Buffering* e *Spooling*:** Utiliza *buffers* para armazenar temporariamente dados durante transferências I/O e *spoolers* para gerenciar a fila de impressão e outras operações assíncronas;
  - **Gerenciamento de Recursos:** Coordena o acesso aos dispositivos entre múltiplos processos, evitando conflitos e garantindo o uso eficiente dos recursos de *hardware*.

Principais Componentes de um S.O. (V)

- **Interface com o Usuário:** a parte de um S.O. que permite aos usuários interagirem com o sistema, proporcionando comandos e *feedback*;
- **Principais tipos de interfaces:**
  - **Interface de Linha de Comando (CLI):** Permite interação baseada em texto, onde o usuário digita comandos. Exemplo: *Bash* no Linux, *Prompt* de Comando no Windows.
  - **Interface Gráfica de Usuário (GUI):** Oferece uma interface visual com janelas, ícones, menus e dispositivos apontadores (como o mouse). Exemplo: Windows, macOS, GNOME no Linux;
- **Principais responsabilidades:**
  - **Execução de Comandos:** Permite que os usuários executem programas e comandos;
  - **Gerenciamento de Janelas:** Coordena a abertura, fechamento, redimensionamento e movimentação de janelas em uma GUI;
  - **Feedback Visual e Sonoro:** Fornece informações visuais e sonoras em resposta às ações do usuário;
  - **Acessibilidade:** Oferece opções de acessibilidade para usuários com necessidades especiais, como leitores de tela e atalhos de teclado.

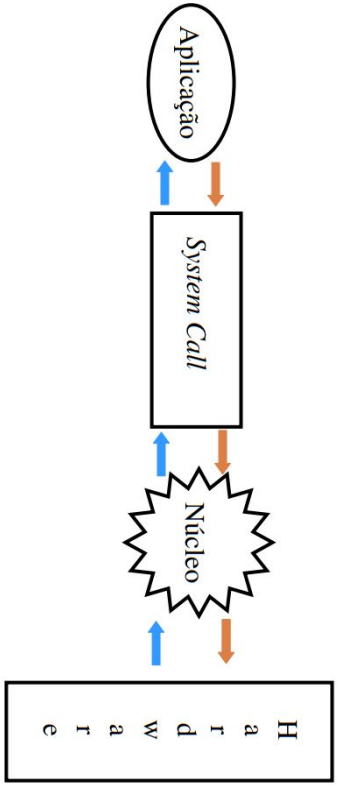
Principais Componentes de um S.O. (VI)

- **Segurança e Controle de Acesso:** protegem o sistema contra acessos não autorizados e ameaças, garantindo a integridade, confidencialidade e disponibilidade dos dados. Principais responsabilidades:
  - **Autenticação:** Verifica a identidade dos usuários, geralmente através de senhas, tokens ou sistemas biométricos.
  - **Autorização:** Define e aplica políticas de acesso, determinando o que os usuários autenticados podem fazer.
  - **Controle de Acesso:** Utiliza listas de controle de acesso (ACLs) e outros mecanismos para gerenciar permissões de usuários e processos;
  - **Proteção contra *Malwares*:** Implementa medidas para detectar e neutralizar vírus, *worms*, *trojans* e outros tipos de *malware*;
  - **Auditoria e Monitoramento:** Registra e monitora atividades no sistema para detectar e responder a comportamentos suspeitos;
  - **Criptografia:** Utiliza criptografia para proteger dados sensíveis em trânsito e em repouso, garantindo a confidencialidade e integridade.

Kernel de um S.O.

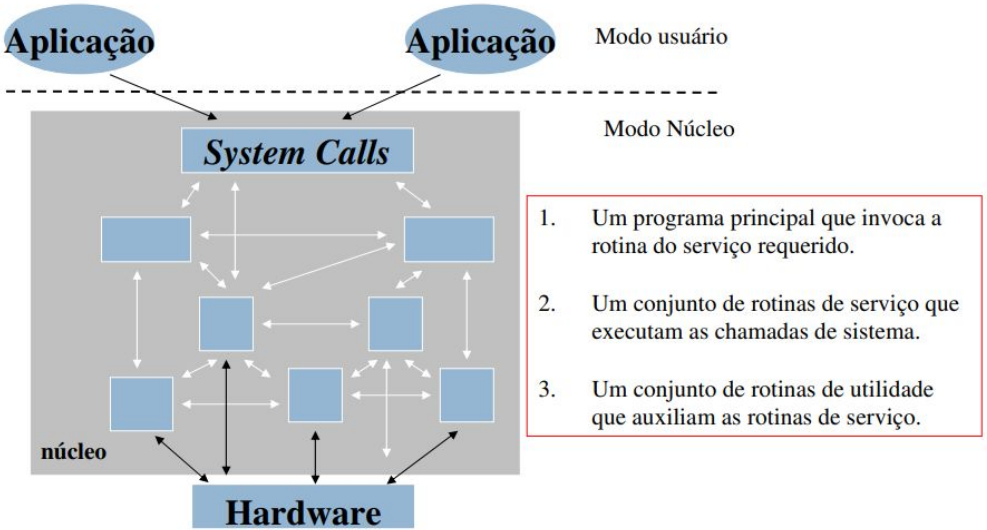
- *Kernel* representa o núcleo do Sistema Operacional, que é o principal *software* de um computador, executado logo após a BIOS *(Basic Input/Output System)*;
    - A BIOS é um *firmware* localizado na memória ROM da placa-mãe. Realiza o reconhecimento e análise dos componentes do *hardware* (POST), carrega o *bootloader* do S.O. e permite alterações de configuração no *hardware;*
    - Termina sua execução assim que o *Kernel* do S.O. inicia e é armazenado na memória RAM;
- A comunicação entre aplicações em **modo usuário** e o *Kernel* do S.O. faz-se, principalmente, por meio de *System Calls:*

Windows	Linux	Descrição
CloseHandle	close	Fechar arquivo
CreateFile	open	Criar arquivo
DeleteFile	unlink	Apagar arquivo
ExitProcess	exit	Terminar um processo e todos os seus segmentos
GetLocalTime	time	Recuperar local, data e tempo atuais



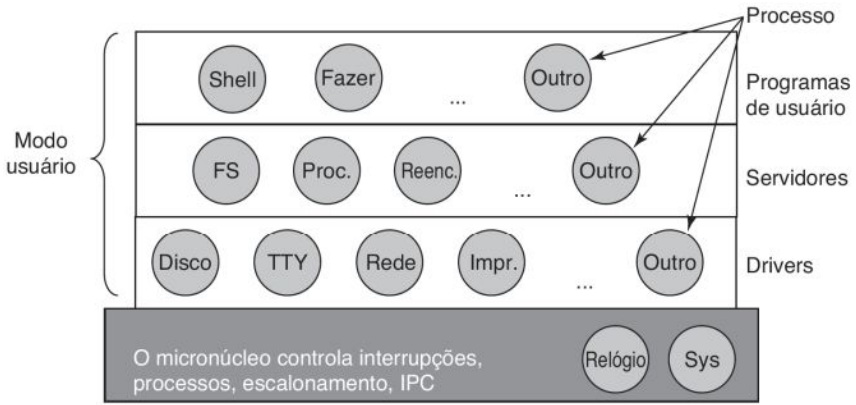
Tipos de *Kernels* (I)

- **Monolítico (Linux, BSD, MS-DOS, Solaris):**
  - Os controladores de dispositivos e as extensões de núcleo são executadas no espaço de núcleo, com acesso completo ao *hardware*;
  - Pode ser comparada com uma aplicação formada por vários procedimentos que são compilados separadamente e depois referenciados, formando um grande e único programa executável;
  - Como todos os módulos são executados em um mesmo espaço de endereçamento, se houver ocorrência de erro em um desses espaços, todo o sistema pode ser afetado.



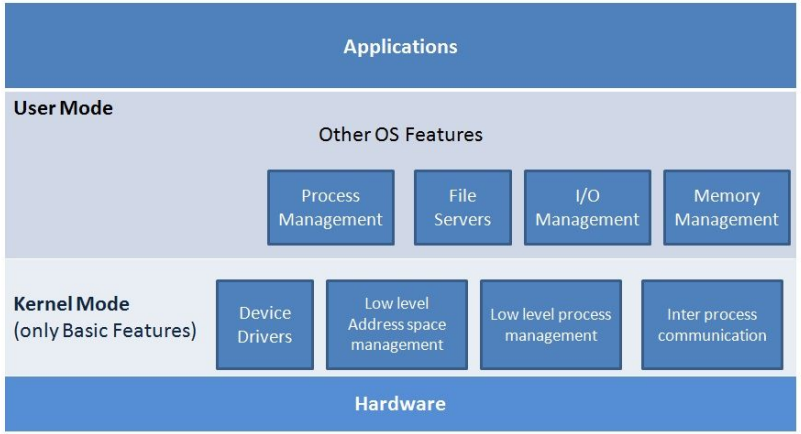
Tipos de *Kernels* (II)

- ***Micro-Kernel* (BeOS, L4, Mach, Minix, MorphOS, QNX, VSTa e Integrity, K42, PikeOS, Symbiam, MINIX3):**
  - A idéia básica por trás do projeto do *micro-kernel* é alcançar alta confiabilidade por meio da divisão do sistema operacional em módulos pequenos, bem definidos, e apenas um desses módulos (*micro-kernel*) é executado no modo *Kernel* e o restante é executado como processos de usuário;
  - Quando há execução de cada *driver* de dispositivo e cada sistema de arquivo como processo separado, um erro em um deles pode quebrar aquele componente, mas não pode quebrar o sistema inteiro, como na arquitetura monolítica;
  - *Micro-kernels* são comuns em aplicações de tempo real, industriais, aviônica e militares, que são cruciais e têm requisitos de confiabilidade muito altos.



Tipos de *Kernels* (III)

- **Híbridos (AmigaOS, Android, Chrome, Macintosh, webOS, Windows, OSX e Xinu):**
  - O híbrido combina a estabilidade e a segurança do *micro-kernel* com o desempenho do monolítico;
  - **Vantagens dos *Kernels* Híbridos:**
    - **Desempenho e Eficiência:** Ao manter serviços críticos no espaço do *kernel*, um *kernel* híbrido pode oferecer desempenho comparável ao de um kernel monolítico.
    - **Modularidade e Flexibilidade:** A capacidade de carregar e descarregar módulos dinamicamente oferece flexibilidade para atualizar e expandir funcionalidades.
    - **Estabilidade e Segurança:** A modularidade permite isolar componentes, o que pode aumentar a estabilidade e segurança do sistema.



Tipos de Sistemas Operacionais

- Os sistemas operacionais podem ser classificados de várias maneiras com base em sua finalidade, funcionalidade, arquitetura e ambiente de uso. Abaixo estão alguns dos principais tipos de sistemas operacionais e suas características:

- **Sistemas Operacionais de Tempo Real (*Real-Time Operating Systems* - RTOS):** (i) Projetados para aplicações que requerem respostas rápidas e previsíveis; (ii) Utilizados em ambientes onde o tempo de resposta é crítico, como em sistemas de controle industrial, aeronaves e dispositivos médicos. Exemplos: VxWorks, QNX, FreeRTOS;

- **Sistemas Operacionais Embarcados:** (i) Projetados para dispositivos específicos, com funções limitadas e otimização para o hardware em que operam. (ii) Encontrados em dispositivos como smartphones, televisores, eletrodomésticos, sistemas de navegação e equipamentos industriais. Exemplos: Android, iOS, FreeRTOS, Contiki;

- **Sistemas Operacionais Móveis:** (i) Projetados para dispositivos móveis, como *smartphones* e *tablets*. (ii) Otimizados para eficiência de energia, interface de usuário amigável e integração com sensores e outras funcionalidades móveis. Exemplos: Android, iOS, Windows Phone (descontinuado);

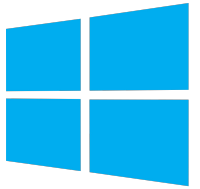
- **Sistemas Operacionais de Desktop:** (i) Usados em computadores pessoais, *laptops* e estações de trabalho. (ii) Projetados para fornecer uma interface de usuário gráfica amigável e suporte a uma ampla variedade de aplicações de produtividade e entretenimento. Exemplos: Windows, macOS, Linux (Ubuntu, Fedora, etc);

- **Sistemas Operacionais de Servidor: (i)** Otimizados para fornecer serviços a outros computadores na rede, como servidores web, servidores de banco de dados e servidores de arquivos. (ii) Focam na estabilidade, segurança e capacidade de gerenciamento. Exemplos: Windows Server, Linux (CentOS, Ubuntu Server), FreeBSD.

Breve Contexto Histórico dos S.Os.

- **Resumo da Evolução:**
  - 1950s: Computação em *batch* sem sistemas operacionais complexos;
  - 1960s: Surgimento dos primeiros sistemas operacionais com multiprogramação;
  - 1970s: Expansão do UNIX e primeiros sistemas operacionais para microcomputadores;
  - 1980s: Revolução dos PCs com MS-DOS, MacOS e evolução do UNIX;
  - 1990s: Diversificação com Windows, Linux e sistemas operacionais móveis emergentes;
  - 2000s: Era da internet com Windows XP, Mac OS X, Android e iOS;
  - 2010s: Convergência de dispositivos, computação em nuvem e virtualização;
  - 2020s: Avanços em IA, integração na nuvem e novas versões de OS populares.

Breve Contexto Histórico dos S.Os.



- **Resumo da Evolução:**
  - **1950s:** Computação em *batch* sem sistemas operacionais complexos;
  - **1960s:** Surgimento dos primeiros sistemas operacionais com multiprogramação;
  - **1970s:** Expansão do UNIX e primeiros sistemas operacionais para microcomputadores;
  - **1980s:** Revolução dos PCs com MS-DOS, Mac OS e evolução do UNIX;
  - **1990s:** Diversificação com Windows, Linux e sistemas operacionais móveis emergentes;
  - **2000s:** Era da internet com Windows XP, Mac OS X, Android e iOS;
  - **2010s:** Convergência de dispositivos, computação em nuvem e virtualização;
  - **2020s:** Avanços em IA, integração na nuvem e novas versões de OS populares.

MacOS

Apresentação do Linux

- Na década de 80, os sistemas operacionais eram inteiramente proprietários, comandados por empresas como Microsoft e IBM;
- Nesse ínterim, Richard Stallman lança o **Projeto GNU** em 1983, com o propósito de criar um sistema operacional completamente livre, semelhante ao UNIX, mas sem as restrições de licenciamento;
- O Projeto GNU progrediu significativamente, desenvolvendo muitos dos componentes necessários para um sistema operacional completo, como compiladores, editores de texto, e bibliotecas. No entanto, faltava o ***kernel*** do sistema operacional;

- Linus Torvalds lançou a primeira versão do kernel, 0.01, em setembro de 1991. O código fonte foi disponibilizado posteriormente sob a Licença Pública Geral GNU (GPL), alinhando-o com a filosofia do software livre defendida pelo Projeto GNU;
- O lançamento do *kernel* Linux sob a GPL incentivou a colaboração de programadores de todo o mundo. Desenvolvedores contribuíram com código, correções de *bugs* e novas funcionalidades, tornando o *kernel* mais robusto e funcional;
- Atualmente, o *kernel* Linux é utilizado no desenvolvimento de diversos S.Os. como as distros Linux (Ubuntu, Fedora, Red Hat, etc), como também o Android.

Principais Distros do Linux

- **Distribuições para Desktop:**
  - **Ubuntu** (e suas variantes como Kubuntu, Xubuntu, etc.): Popular para uso em desktops devido à sua facilidade de uso e vasta disponibilidade de software.
  - **Linux Mint**: Baseado no Ubuntu, é conhecido por sua facilidade de uso e design elegante.
  - **elementary OS:** Conhecido por sua beleza e simplicidade, é uma ótima escolha para usuários que procuram uma experiência de desktop elegante.
- **Distribuições de Servidor:**
  - **CentOS/RHEL (Red Hat Enterprise Linux):** CentOS é uma versão comunitária do RHEL, enquanto o RHEL é uma distribuição empresarial com suporte comercial. Ambos são amplamente utilizados em ambientes de servidor.
  - **Ubuntu Server:** Uma variante do Ubuntu otimizada para uso em servidores.
  - **openSUSE:** Uma distribuição robusta e estável, com uma forte orientação para servidores.
- **Distribuições para Segurança e Penetração:**
  - **Kali Linux:** Projetado para testes de segurança e auditoria de redes.
  - **Parrot Security OS:** Uma distribuição Linux baseada no Debian, projetada para testes de penetração, hacking ético e avaliação de vulnerabilidades.
  - **BackBox:** Outra distribuição Linux focada em segurança e testes de penetração.

Laboratório de Fundamentos em TIC

Introdução a Sistemas Operacionais e Linux

Prof. Gabriel Resende Machado

gabrielmachado@unifeso.edu.br

https://www.linkedin.com/in/machadogabriel

https://github.com/UNIFESO-Gabriel/fundamentos-em-tic