E.O.S - Um sistema operacional para a educação

Thiago Vandes Alkmim | Carlos Henrique Oliveira Dias | João Pedro Romanzini dos Santos
Jefferson Riper da Silva

ETEC de Guarulhos

Resumo

Nosso projeto parte de uma realidade comum no Brasil: muitos computadores apresentam dificuldades em executar sistemas pesados, como o Windows, que consomem recursos em excesso devido a processos constantes em segundo plano. Diante desse cenário, desenvolvemos a proposta de um **sistema operacional leve, construído a partir do Linux From Scratch (LFS) e Beyond Linux From Scratch (BLFS)**, com foco em **desempenho e acessibilidade**.

A iniciativa busca não apenas melhorar a performance de máquinas com hardware limitado, mas também oferecer um ambiente mais eficiente para o **processo de ensino-aprendizagem**, permitindo que estudantes explorem a tecnologia de maneira prática e fluida. Dessa forma, o projeto une **otimização técnica** e **impacto educacional**, mostrando como é possível criar soluções sob medida para os desafios reais enfrentados por usuários e instituições de ensino.

Introdução

Nossa equipe pretende desenvolver um sistema operacional de distribuição Linux voltado para educação em computadores de nível de entrada. Reconhecemos que além do acesso a hardware de qualidade ser dificil nas redes públicas de ensino do Brasil, também é perceptivel que a grande demanda de sistemas operacionais como Windows podem intensificar o desconforto ao uso e exigir mais por recursos que não serão utilizados no meio educacional.

Tendo em vista também a dificuldade burocráticas em solicitar e receber computadores mais potentes torne a situação ainda mais complicada. Por isso, pretendemos desenvolver um sistema operacional com base no guia Linux From Scratch e Beyond Linux From Scratch, que é um projeto sem fim lucrativos fundado em 1998 e atualizado constantemente para auxiliar na criação de sistemas operacionais personalizados além de expandir a compreensão do funcionamento de um sistema operacional.

Objetivo

O objetivo principal deste projeto é desenvolver um sistema operacional leve e funcional, construído, capaz de oferecer melhor desempenho em computadores com hardware limitado e, ao mesmo tempo, servir como uma ferramenta de apoio no ensino.

A pesquisa busca demonstrar que, ao compreender e construir um sistema a partir de seus fundamentos, é possível criar soluções otimizadas que atendam às necessidades de usuários e instituições de ensino, proporcionando não apenas ganhos técnicos em termos de desempenho, mas também uma experiência educacional enriquecedora.

Metodologia

A realização desta pesquisa seguiu uma abordagem **experimental e exploratória**, fundamentada na construção prática de um sistema operacional a partir do projeto **Linux From Scratch (LFS)** e sua extensão **Beyond Linux From Scratch (BLFS)**. O desenvolvimento foi conduzido em ambiente controlado de virtualização, com registros minuciosos de cada etapa para análise técnica e pedagógica.

A seguir, descrevem-se os métodos e procedimentos empregados, acompanhados dos materiais utilizados e das referências conceituais que nortearam a pesquisa.

a) Preparação do ambiente de desenvolvimento

O ambiente de desenvolvimento foi preparado utilizando o **Oracle VirtualBox** como ferramenta de virtualização, permitindo a criação de uma máquina virtual isolada. Essa escolha se justifica pela necessidade de realizar testes repetitivos, aplicar snapshots de restauração e evitar riscos de comprometimento do sistema host, conforme práticas recomendadas em pesquisas aplicadas em engenharia de software

b) Coleta de materiais e ferramentas

Foram utilizados os pacotes disponibilizados pelo projeto oficial do **Linux From Scratch (LFS, 2023)** e **Beyond Linux From Scratch (BLFS, 2023)**, baixados diretamente de seus repositórios oficiais. A coleta dos materiais incluiu:

- **Toolchain**: `binutils`, `gcc`, `glibc`, `linux-headers`.
- **Utilitários essenciais**: `bash`, `coreutils`, `grep`, `sed`, `findutils`, `tar`, entre outros.
- **Ferramentas de build**: `Meson`, `Ninja`, `make` e `autotools`.
- **Linguagens de suporte**: módulos Python e Perl, instalados como dependências de scripts e bibliotecas.
- **Bibliotecas gráficas**: Xorg, Mesa, Cairo, Pango, GTK, além do ambiente XFCE.

Todos os pacotes foram verificados por meio de checksums (MD5/SHA256) para garantir integridade, seguindo recomendações de segurança do projeto LFS (Beekmans et al., 2023).

c) Referências metodológicas

A pesquisa baseou-se em referências clássicas e contemporâneas para validação de seus métodos:

- Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems*. Pearson.
- Beekmans, G. et al. (2023). *Linux From Scratch Project*. Disponível em: http://www.linuxfromscratch.org
- Beyond Linux From Scratch (BLFS) (2023). *BLFS Book*. Disponível em: http://www.linuxfromscratch.org/blfs

Desenvolvimento

A motivação do projeto parte de dois eixos complementares: (1) um problema prático — muitos computadores no contexto brasileiro possuem hardware limitado e apresentam baixo desempenho com sistemas pesados — e (2) um objetivo educacional — promover aprendizado profundo sobre

sistemas operacionais ao construir um sistema “do zero”.
A fundamentação apoiou-se em estudos sobre footprint de sistemas operacionais, práticas de engenharia de software aplicada à construção de sistemas e metodologias de ensino prático (aprendizagem baseada em projetos). A escolha pelo **Linux From Scratch (LFS)** e **Beyond LFS (BLFS)** fundamentou-se na necessidade de controle total sobre cada componente e no caráter didático do processo.

Hipóteses principais

- Um sistema construído e otimizado a partir do LFS terá *menor consumo de memória e maior responsividade* em máquinas com hardware limitado quando comparado a sistemas comerciais genéricos.
- A experiência prática de montar o sistema do zero *melhora significativamente* a compreensão dos estudantes sobre arquitetura de sistemas operacionais.

Planejamento e ambiente de desenvolvimento

- **Host:** Gentoo Linux (forneceu flexibilidade para compilar pacotes e ferramentas necessárias).
- **Virtualização:** Oracle VirtualBox — VM usada para criar imagens experimentais, facilitar snapshots, testes e recuperação rápida de estados.
- **Controle de versões / documentação:** repositório com roteiro de build, checklist de dependências, scripts auxiliares e logs de compilação.
- **Ferramentas de build modernas:** Meson + Ninja para projetos que exigem build contemporâneo; `make`/`autotools` para pacotes clássicos.
- **Dependências de linguagem:** módulos Python e Perl instalados quando requeridos por scripts de build e utilitários.

Preparação inicial (host e VM)

1. **Configuração da VM:** alocação de CPU/cores, RAM e disco; escolha de adaptador de rede (NAT para fácil internet, bridge para testes em LAN), habilitação de aceleração VT-x/AMD-V.
2. **Snapshots:** criação de snapshots regulares (pontos de restauração antes de etapas arriscadas como remover pacotes ou modificar toolchain).
3. **Coleta de fontes:** download de tarballs da árvore LFS (toolchain, utilitários essenciais) e do BLFS (pacotes gráficos, Xorg, drivers, XFCE e dependências).

Construção do LFS - passo a passo técnico

a) Ambiente de trabalho e variáveis

- Criação de partição/volume virtual e montagem em `/mnt/lfs`.

- Definição de variáveis de ambiente (`LFS`, `LFS_TGT`, `PATH`) e criação de um ambiente isolado para compilação (chroot posteriormente).

b) Toolchain temporário (chroot temporário)

- Compilação sequencial de: `binutils`, `gcc` (fase inicial), `glibc` (headers e biblioteca básica), `linux-headers`.
- Objetivo: criar um *toolchain* auto-contido que seja independente do sistema host, evitando contaminação por bibliotecas do Gentoo.

c) Toolchain final e construção do sistema base

- Reconstrução de `binutils` e `gcc` agora usando o toolchain temporário para gerar um compilador final.
- Compilação de utilitários essenciais em ordem: `coreutils`, `file`, `findutils`, `sed`, `grep`, `bash`, `tar`, `gzip`, `util-linux`, `procps-ng`, entre outros.
- Instalação de man-pages, documentação e configuração de locale, fstab, timezone e hostname.

d) Kernel e bootloader

- Configuração e compilação do kernel Linux: seleção de drivers necessários (sistemas de arquivos, rede, drivers virtuais para VirtualBox), habilitação de módulos relevantes.
- Instalação/integração de bootloader (GRUB) e teste de boot em VM.

Integração BLFS e implementação da interface gráfica

A passagem para BLFS envolveu a construção de uma pilha gráfica e das bibliotecas contemporâneas. Principais etapas:

a) Xorg / pilha gráfica

- Compilação do `xorg-server` e bibliotecas X11 (libX11, xtrans, libXrandr, etc).
- Instalação de drivers gráficos compatíveis com o ambiente virtual (ex.: drivers vesafb/vmware/vboxvideo conforme suporte da VM).
- Configuração de `xorg.conf`/autodetect para detectar adaptador de vídeo do VirtualBox.

b) Bibliotecas gráficas e toolkits

- Compilação de `mesa` (opengl), `libdrm`, `libinput`, `xkeyboard-config`.
- Instalação de toolkits e bibliotecas de nível superior: `cairo`, `pango`, `atk`, `glib`, `gtk+` (e as versões necessárias do GTK2/Gtk3 conforme XFCE), além de `libxcb` e `pixman`.

c) Meson & Ninja

- Uso de Meson + Ninja para construir pacotes modernos (algumas bibliotecas do GNOME, componentes do XFCE ou dependências que adotam Meson).
- Necessidade de Python e módulos `meson`/`ninja` no ambiente de build; garantia de compatibilidade entre versões de Python e Meson.

d) Módulos Python e Perl

- Instalação de módulos Python (via `pip` local ou build de fontes) demandados por utilitários e scripts de build.
- Instalação de módulos Perl usados por ferramentas legadas; ambos (Python/Perl) eram frequentemente pré-requisitos para pacotes BLFS.

e) XFCE

- Compilação e instalação dos componentes do XFCE: `xfce4-panel`, `xfdesktop`, `xfwm4`, `thunar` (file manager) e plugins.
- Configuração de sessão e gerenciador de janelas; seleção de um display manager leve (ex.: `lightdm`), ou configuração para `startx` manual conforme opção escolhida.
- Ajustes visuais: temas, ícones e scripts de inicialização para proporcionar a identidade visual desejada.

f) Configuração de rede (IP estático e DHCP)

- **IP estático:** edição dos arquivos de configuração de rede (ex.: `/etc/network/interfaces`, scripts de netif, ou configs do NetworkManager alternativo) para definir IP, máscara, gateway e DNS; testes com `ip addr` e `route`.
- **DHCP:** instalação e configuração do cliente DHCP (`dhclient` ou `dhcpcd`), testes com ambientes com servidor DHCP e fallback para IP estático quando necessário.
- Testes de conectividade (`ping`, `curl`) e configuração de DNS (`resolv.conf`).

g) Gestão de usuários e permissões

- Criação de usuários padrão e grupos via `useradd`/`groupadd`, definição de senhas, e políticas de sudo (arquivo `sudoers`) para delimitar privilégios administrativos.
- Ajuste de permissões de diretório, `umask` padrão e propriedades de arquivos importantes (`/var`, `/home`, `/tmp`).

h) Otimizações e práticas para manter leveza

- Compilação com flags otimizadas e uso de `make -j$(nproc)` para acelerar builds.

- Remoção de pacotes e serviços desnecessários; desabilitação de daemons não utilizados no boot para reduzir I/O e memória.

- Stripping de binários quando apropriado (`strip`) e verificação de dependências redundantes.

- Criação de aliases e utilitários voltados para estudantes, tornando operações comuns mais simples (aliases em português, scripts de ajuda).

Testes, medição e avaliação

- **Métricas coletadas:** tempo de boot (cronômetro + systemd-analyze quando aplicável), uso de memória em idle (`free -m`), consumo de CPU durante tarefas típicas, tempo de abertura de aplicativos.

- **Testes comparativos:** execução de workloads leves (edição de texto, navegação web simples) em máquinas antigas vs. máquinas com Windows (quando disponível) para comparação qualitativa e quantitativa.

- **Avaliação educacional:** testes de usabilidade com estudantes, questionários sobre clareza das instruções e curva de aprendizado.

Principais problemas identificados e soluções aplicadas

- **Dependência circular / versões incompatíveis:** problema recorrente ao compilar bibliotecas interdependentes. *Solução:* seguir estritamente a ordem de build do LFS/BLFS, manter um repositório local com versões testadas e usar patches quando necessário.

- **Falhas em Meson/Ninja por falta de módulo Python:** resolvido instalando uma versão compatível de Python + Meson no ambiente.

- **Xorg não iniciava (problemas de driver):** ajuste das configurações do kernel (módulos de vídeo) e instalação dos drivers virtuais do VirtualBox ou fallback para driver VESA.

- **Problemas com chroot e variáveis de ambiente:** garantir montagem de `/proc`, `/sys`, `/dev` e export correto de `PATH` dentro do chroot.

- **Performance de compilação lenta:** otimização via `-O2 -march=native` quando aplicável e uso de SSD nos testes.

- **Gerenciamento de atualizações:** ausência de gerenciador de pacotes levou à necessidade de documentar passos de atualização manual e considerar, futuramente, um sistema de empacotamento.

Resultados e Discussões

O desenvolvimento do sistema operacional proposto resultou em um ambiente leve, funcional e otimizado, construído a partir da metodologia **Linux From Scratch (LFS)** e estendido pelo **Beyond**

Linux From Scratch (BLFS). O processo culminou na criação de um sistema com interface gráfica, navegação web funcional, configuração de rede (tanto com IP estático quanto por DHCP) e gerenciamento básico de usuários e permissões.

Resultados obtidos

Ao longo dos meses de desenvolvimento, foram alcançados os seguintes resultados:

- **Construção completa do LFS**: sistema base funcional, independente do host, contendo todos os utilitários essenciais.
- **Integração BLFS**: instalação da pilha gráfica (Xorg, Mesa, bibliotecas essenciais) e do ambiente de desktop XFCE.
- **Rede configurada**: suporte a endereços estáticos e dinâmicos via cliente DHCP, com conectividade validada.
- **Usuários e permissões**: definição de contas específicas e atribuição de privilégios com o uso de grupos e sudoers.
- **Desempenho otimizado**: medições iniciais mostraram consumo de memória significativamente inferior em comparação com sistemas comerciais como o Windows 10 e 11, especialmente em máquinas virtuais com recursos limitados.

Considerações Finais

A pesquisa realizada confirmou a viabilidade técnica e educacional da construção de um sistema operacional leve a partir do **Linux From Scratch (LFS)**, expandido pelo **Beyond Linux From Scratch (BLFS)**. Os resultados demonstraram que a abordagem proposta permite a criação de um sistema funcional, otimizado e adaptável, podendo servir como recurso pedagógico para o ensino de sistemas operacionais e infraestrutura computacional.

As **conclusões finais** indicam que:

- O processo de construção manual, apesar de complexo, reforça significativamente a compreensão sobre dependências, kernel, bibliotecas e serviços do sistema, configurando um diferencial educacional.
- A adoção de software livre e de metodologias abertas garante a replicabilidade do projeto e sua sustentabilidade técnica e econômica.

A **situação atual do projeto** corresponde a um sistema completo em ambiente virtualizado, com interface gráfica (XFCE), navegador funcional, configuração de rede por IP estático e DHCP, além de gerenciamento básico de usuários e permissões.

Os **próximos passos** projetados para a continuidade da pesquisa incluem:

a) **Implementação em hardware real**, a fim de validar o desempenho em computadores antigos ou de baixo custo.

b) Criação de uma imagem ISO instalável, facilitando a distribuição do sistema em larga escala.

c) Desenvolvimento de ferramentas de automação parcial, como scripts de instalação e atualização, visando reduzir a complexidade de manutenção.

d) Avaliação pedagógica formal, com aplicação em turmas de ensino técnico ou superior para medir o impacto do projeto no aprendizado dos estudantes.

O sistema desenvolvido demonstra que é possível aliar desempenho, baixo custo e potencial educativo, contribuindo tanto para a melhoria da experiência de ensino quanto para o aproveitamento sustentável de recursos computacionais.

Referências Bibliográficas

BLFS DEVELOPMENT TEAM. *Beyond Linux From Scratch*. Versão 12.0. Disponível em: http://www.linuxfromscratch.org/blfs/. Acesso em: 30 set. 2025.

GNU PROJECT. *GNU C Library*. Free Software Foundation, 2025. Disponível em: https://www.gnu.org/software/libc/. Acesso em: 30 set. 2025.

LFS DEVELOPMENT TEAM. *Linux From Scratch*. Versão 12.0. Disponível em: http://www.linuxfromscratch.org/lfs/. Acesso em: 30 set. 2025.

MESON BUILD SYSTEM. *Meson Build: Documentation*. 2025. Disponível em: https://mesonbuild.com/. Acesso em: 30 set. 2025.

ORACLE CORPORATION. *Oracle VM VirtualBox User Manual*. Versão 7.0. Disponível em: https://www.virtualbox.org/manual/. Acesso em: 30 set. 2025.

PERL FOUNDATION. *Perl Documentation*. 2025. Disponível em: https://perldoc.perl.org/. Acesso em: 30 set. 2025.

PYTHON SOFTWARE FOUNDATION. *Python Documentation*. 2025. Disponível em: https://docs.python.org/3/. Acesso em: 30 set. 2025.

XFCE DEVELOPMENT TEAM. *Xfce Desktop Environment*. 2025. Disponível em: https://www.xfce.org/. Acesso em: 30 set. 2025.