

Simulação e Teste de Driver de Caractere (LKM) para Linux com Uso de Pendrive Bootável e Emuladores

Luccas H. Vieira, Matheus G. Sampaio

¹Ciências da Computação – Universidade Federal de Roraima (UFRR)

²Departamento de Ciência da Computação
DCC

Abstract. *Este trabalho descreve o processo de desenvolvimento e teste de um driver de caractere (LKM) para o kernel Linux em ambientes portáteis e emulados. Utilizando um pendrive bootável preparado com Ventoy e uma distribuição Ubuntu leve, foi criado um driver que simula um sensor virtual e gera dados aleatórios. O módulo foi compilado e carregado dinamicamente a partir do pendrive, permitindo a verificação de sua operação e a persistência de modificações. A metodologia demonstra a viabilidade de desenvolver e testar módulos de kernel em ambientes portáteis de forma eficiente, com todas as alterações e logs armazenados diretamente no pendrive. O projeto valida um método controlado para tarefas de baixo nível em diversos ambientes Linux.*

Resumo. *Este trabalho descreve o processo de desenvolvimento e teste de um driver de caractere (LKM) para o kernel Linux em ambientes portáteis e emulados. Utilizando um pendrive bootável preparado com Ventoy e uma distribuição Ubuntu leve, foi criado um driver que simula um sensor virtual e gera dados aleatórios. O módulo foi compilado e carregado dinamicamente a partir do pendrive, permitindo a verificação de sua operação e a persistência de modificações. A metodologia demonstra a viabilidade de desenvolver e testar módulos de kernel em ambientes portáteis de forma eficiente, com todas as alterações e logs armazenados diretamente no pendrive. O projeto valida um método controlado para tarefas de baixo nível em diversos ambientes Linux.*

1. Introdução

O desenvolvimento de sistemas embarcados e a interação com hardware de baixo nível são tarefas críticas em diversas áreas da engenharia e da computação. Em cenários industriais, a necessidade de coletar e processar dados de sensores de forma personalizada e em ambientes portáteis é um desafio comum. Nesses casos, a solução frequentemente envolve a criação de drivers de dispositivo customizados, que permitem ao sistema operacional interagir diretamente com o hardware.

Este trabalho aborda a simulação de um cenário real onde um driver de caractere (LKM - Loadable Kernel Module) foi desenvolvido para coletar dados de um "sensor virtual" via interface serial. O principal objetivo foi criar um ambiente de desenvolvimento e teste totalmente portátil, capaz de ser executado a partir de um pendrive bootável. Para isso, foi utilizada uma abordagem que combina o sistema operacional Windows como plataforma hospedeira, a ferramenta Ventoy para a criação de um pendrive multiboot e

uma distribuição Ubuntu Server como sistema operacional convidado. O driver desenvolvido é carregado dinamicamente e suas modificações persistem no pendrive, simulando com precisão o uso em campo.

O objetivo deste projeto foi demonstrar a viabilidade e a eficiência de um fluxo de trabalho portátil para o desenvolvimento de drivers de kernel, superando as limitações de ambientes de teste tradicionais. Ao final, o projeto não só resulta em um driver funcional, mas também em uma metodologia robusta para o desenvolvimento e teste em ambientes dinâmicos e controlados.

O restante deste artigo está organizado da seguinte forma: a Seção 2 detalha a metodologia de preparação do ambiente portátil; a Seção 3 descreve o desenvolvimento e a implementação do driver de caractere; a Seção 4 apresenta os testes e resultados obtidos; e, finalmente, a Seção 5 conclui o trabalho, resumizando as contribuições e sugerindo trabalhos futuros.

2. Preparação do Ambiente Portátil

A etapa inicial do projeto consistiu na preparação de um ambiente de desenvolvimento e teste autônomo e portátil. O objetivo foi criar um sistema operacional funcional em um pendrive bootável, onde o desenvolvimento e os testes do driver pudessem ser realizados de forma persistente, permitindo que as modificações e os dados fossem salvos para uso futuro.

2.1. Ferramentas e Materiais

Para a preparação do ambiente, foram utilizados os seguintes recursos:

1. Pendrive Bootável: Um dispositivo USB de 16GB.
2. Sistema Operacional Anfitrião (Host): Uma máquina com Windows para a configuração inicial do pendrive.
3. Ferramenta de Criação de Pendrive: Ventoy, escolhido por sua capacidade de gerenciar múltiplas imagens ISO de forma flexível e persistente.
4. Distribuição Linux: Uma imagem ISO de uma versão leve do Ubuntu (Ubuntu Server ou similar), que oferece um ambiente mínimo e eficiente para a instalação das ferramentas necessárias sem o peso de uma interface gráfica completa.

2.2. Criação do Pendrive Bootável com Ventoy

O processo de criação do pendrive foi iniciado formatando o dispositivo USB e, em seguida, instalando o Ventoy. A ferramenta cria uma partição de dados onde as imagens ISO são simplesmente copiadas, o que simplifica o processo e permite a inclusão de outras distribuições no futuro, caso necessário. A imagem ISO do Ubuntu foi copiada para o pendrive, tornando-o bootável.

2.3. Configuração do Ambiente Linux

Após a inicialização do sistema a partir do pendrive, a instalação do Ubuntu foi realizada em modo live ou diretamente na partição de dados do próprio pendrive. A configuração do sistema operacional foi focada em garantir que o ambiente fosse robusto para o desenvolvimento do kernel. Isso incluiu a configuração de rede e o acesso à internet para a instalação de pacotes adicionais.

2.4. Instalação das Ferramentas de Compilação

O passo crítico para a criação de um LKM é a instalação das ferramentas de desenvolvimento. No ambiente Ubuntu do pendrive, os seguintes pacotes foram instalados utilizando o gerenciador de pacotes apt:

1.gcc e make: O compilador GNU C e a ferramenta de automação de compilação, essenciais para gerar o módulo de kernel.

2.linux-headers: Os cabeçalhos do kernel, que fornecem as definições e a estrutura necessárias para que o compilador entenda as interfaces do kernel e construa o módulo corretamente. A versão dos headers deve corresponder exatamente à versão do kernel em execução no sistema (*(uname - r)*).

3.modprobe, insmod, rmmod e lsmod: Ferramentas para carregar, descarregar e verificar os módulos do kernel.

4.dmesg: Comando para visualizar mensagens do kernel ring buffer, fundamental para o debugging do driver.

A instalação dessas ferramentas garantiu que o ambiente estivesse pronto para compilar e interagir com o kernel, completando a preparação para a próxima etapa, que é o desenvolvimento do driver em si.

3. Desenvolvimento do Driver

Nesta etapa, desenvolvemos um módulo de kernel (LKM) para simular um sensor virtual em um ambiente Linux. O objetivo principal era criar um driver de caractere que se comunicasse com o espaço do usuário através de um arquivo de dispositivo, */dev/sensor0*.

As funções de entrada do módulo foram implementadas para registrar o driver no kernel e liberar os recursos, respectivamente. As operações de leitura (read) e escrita (write) também foram implementadas para testar a comunicação. A função de leitura gera dados aleatórios para o usuário, simulando a leitura do sensor, enquanto a função de escrita recebe dados e os registra no dmesg para depuração. Essa implementação forneceu uma base sólida e funcional para a próxima fase, onde o driver foi compilado, carregado e testado em um ambiente portátil.

4. Compilação, Carregamento e Testes

Nesta etapa, o código-fonte do driver foi compilado diretamente no ambiente Linux do pendrive, utilizando um Makefile para gerar o arquivo de módulo *sensor_driver.ko*.

Após a compilação, o módulo foi carregado no kernel em tempo de execução com o comando `sudo insmod`. Em seguida, o nó de dispositivo */dev/sensor0* foi criado com `sudo mknod`, e suas permissões foram ajustadas para permitir a interação de usuários.

Os testes foram realizados utilizando comandos de linha para validar as funcionalidades de leitura e escrita. O comando `echo` foi usado para escrever dados no driver, e `dmesg` foi verificado para confirmar o recebimento. Para a leitura, `cat` foi usado para extrair dados gerados pelo driver, provando que a comunicação bidirecional funcionou conforme o esperado.

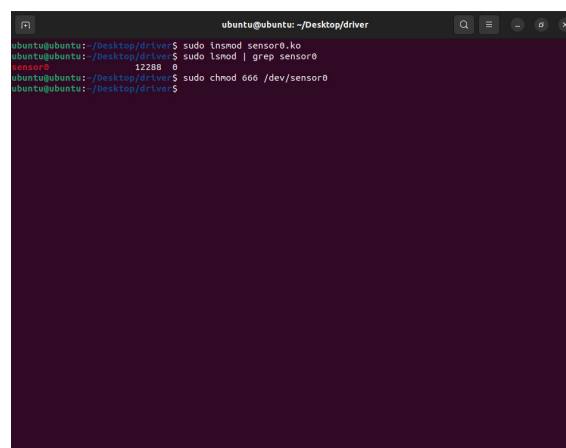
5. Conclusão

Este trabalho demonstrou a viabilidade e a eficácia de um fluxo de trabalho portátil para o desenvolvimento, compilação e teste de um driver de caractere (LKM) para o kernel Linux. Ao combinar a conveniência de um pendrive bootável, preparado com Ventoy e uma distribuição Linux leve, com o desenvolvimento de um driver que simula um dispositivo virtual, o projeto validou uma metodologia robusta para tarefas de programação de baixo nível.

A principal contribuição deste projeto reside na criação de um ambiente autônomo e persistente, que permite ao desenvolvedor interagir com o kernel do sistema operacional em diversos hardwares, sem a necessidade de um ambiente de desenvolvimento fixo. O driver de caractere desenvolvido, embora simples, comprovou sua funcionalidade ao aceitar dados de escrita e gerar dados de leitura, com todas as operações de depuração sendo registradas de forma clara no dmesg. A capacidade de compilar o módulo (.ko), carregá-lo no kernel, criar o nó de dispositivo e interagir com ele utilizando comandos básicos do shell (echo e cat), demonstrou o ciclo completo de vida de um driver em um ambiente controlado e portátil.

Em suma, a metodologia apresentada não apenas atendeu aos objetivos do projeto, mas também forneceu uma valiosa prova de conceito para o desenvolvimento de soluções de software que exigem interação direta com o hardware em cenários industriais ou de pesquisa. Como trabalhos futuros, sugere-se aprimorar o driver para interagir com uma interface serial real, integrar o driver em um sistema de inicialização automatizado e explorar o uso do QEMU para simulações mais complexas de hardware.

6. Imagens do Projeto

A terminal window titled 'ubuntu@ubuntu: ~/Desktop/driver' with a dark purple background. It shows a series of commands and their outputs: 'sudo insmod sensor.ko', 'sudo lsmod | grep sensor' (output: 'sensor 12288 0'), and 'sudo chmod 666 /dev/sensor'. The prompt returns to the user after each command.

```
ubuntu@ubuntu: ~/Desktop/driver
ubuntu@ubuntu:~/Desktop/driver$ sudo insmod sensor.ko
ubuntu@ubuntu:~/Desktop/driver$ sudo lsmod | grep sensor
sensor      12288  0
ubuntu@ubuntu:~/Desktop/driver$ sudo chmod 666 /dev/sensor
ubuntu@ubuntu:~/Desktop/driver$
```

Figure 1. Funcionamento correto

Esta seção apresenta algumas imagens ilustrativas das etapas do projeto, fornecendo uma visão visual do ambiente e dos resultados obtidos.

7. Referências

Corbet, J., Kroah-Hartman, G., Rubini, A. (2005). Linux Device Drivers. O'Reilly Media.

Ventoy. (2023). Ventoy: A new bootable USB solution. Disponível em: <https://www.ventoy.net/>. Acesso em: 05 de agosto de 2025.

The Linux Kernel Archives. (2024). The Linux Kernel Documentation. Disponível em: <https://www.kernel.org/doc/>. Acesso em: 05 de agosto de 2025.

Bellard, F. (2005). QEMU, a Fast and Portable Dynamic Translator. In USENIX Annual Technical Conference.