Daniel Suzuki Naves e Igor Vinicius De Oliveira

PROJETO FINAL - MÓDULO LINUX CRYPTOCHANNEL: Dispositivo de Caractere com Criptografia e Interface /proc

Relatório técnico referente ao Projeto Final da disciplina de Sistemas Operacionais (BCC5002), Módulo 1: Dispositivo de caractere com criptografia e interface /proc, solicitado pelo professor Rodrigo Campiolo na disciplina de Sistemas Operacionais do Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Universidade Tecnológica Federal do Paraná – UTFPR

Departamento Acadêmico de Computação – DACOM

Bacharelado em Ciência da Computação – BCC

Campo Mourão

Dezembro / 2025

Resumo

O presente relatório técnico descreve o desenvolvimento e a implementação do módulo de núcleo Linux **cryptochannel**, conforme as especificações do projeto final da disciplina de Sistemas Operacionais. O módulo foi concebido para atuar como um dispositivo de caractere (~/dev/cryptochannel) que facilita a comunicação segura entre processos (produtor/consumidor) através de um buffer interno (KFIFO) sincronizado por mutexes e filas de espera. A principal funcionalidade é a cifragem e decifragem de dados utilizando um algoritmo de substituição configurável. Adicionalmente, o módulo implementa uma interface no sistema de arquivos ~/proc/cryptochannel para expor estatísticas de uso e permitir a configuração dinâmica da chave de cifragem. Os testes de funcionalidade e concorrência foram realizados com sucesso, demonstrando a estabilidade e a corretude da solução.

Palavras-chave: Módulo de Kernel. Dispositivo de Caractere. KFIFO. Criptografia. Processos.

Sumário

1 Introdução

O presente relatório técnico descreve o desenvolvimento e a implementação do módulo de núcleo Linux denominado **cryptochannel**, conforme as especificações do Projeto Final da disciplina de Sistemas Operacionais (UTFPR - Universidade Tecnológica Federal do Paraná, 2025). O objetivo principal foi consolidar o conhecimento sobre o desenvolvimento de *drivers* de dispositivo de caractere e a manipulação de primitivas de sincronização e comunicação no contexto do kernel (TANENBAUM; BOS, 2023; SILBERSCHATZ; GALVIN; GAGNE, 2014).

2 Descrição da Atividade e Objetivos

A atividade consistiu no desenvolvimento de um módulo que criasse a infraestrutura completa de comunicação segura e configurável no sistema, incluindo o dispositivo `/dev/cryptochannel`, a cifragem das mensagens e a exposição de dados via ~/proc (UTFPR - Universidade Tecnológica Federal do Paraná, 2025).

2.1 Dispositivo de Caractere `/dev/cryptochannel`

O módulo implementa um dispositivo de caractere que, na escrita (`write`), cifra mensagens em texto puro antes de armazená-las no buffer e, na leitura (`read`), decifra o conteúdo antes de enviá-lo ao usuário.

2.2 Interface de Controle `/proc/cryptochannel`

Para monitoramento e configuração, foi criada a seguinte interface:

- **`stats`**: Arquivo somente leitura para expor estatísticas operacionais (mensagens trocadas, bytes criptografados, erros).
- **`config`**: Arquivo de leitura e escrita para definir a chave de cifragem e o modo de operação.

3 Métodos e Implementação

A arquitetura do módulo baseia-se em conceitos fundamentais do sistema operacional (TANENBAUM; BOS, 2023).

3.1 Sincronização Produtor-Consumidor (KFIFO e Locks)

O modelo Produtor-Consumidor foi implementado usando primitivas do kernel:

- **Buffer:** O `kfifo` foi utilizado como buffer circular seguro para armazenar as mensagens cifradas (SALZMAN; MOLLOY, 2025).
- **Exclusão Mútua:** Um `mutex` garantiu a exclusão mútua em todas as operações de E/S, prevenindo condições de corrida (SILBERSCHATZ; GALVIN; GAGNE, 2014).
- **Bloqueio:** Filas de espera (`wait_queue_head_t`) suspenderam o produtor quando o `kfifo` estava cheio e o consumidor quando estava vazio (`wait_event_interruptible`).

3.1.1 Exemplo de Primitivas de Bloqueio

O bloco de código a seguir ilustra o uso das filas de espera para suspender o processo consumidor quando o buffer está vazio:

```
// Dentro de cryptochannel_read()
// Bloqueia e adquire o mutex antes de verificar o buffer
if (mutex_lock_interruptible(&crypto_dev.lock)) return -ERESTARTSYS;

while (kfifo_len(&crypto_dev.buf) == 0) {
    // 1. Libera o lock antes de dormir
    mutex_unlock(&crypto_dev.lock);

    // 2. Suspende o processo, esperando por wake_up_interruptible()
    if (wait_event_interruptible(crypto_dev.read_queue, kfifo_len(&crypto
        return -ERESTARTSYS;
    }

    // 3. Volta a adquirir o lock antes de continuar
    if (mutex_lock_interruptible(&crypto_dev.lock)) return -ERESTARTSYS;
}
// Continua a leitura...
```

3.2 Algoritmo de Cifragem Configurada

O algoritmo implementado foi uma cifra simétrica de substituição baseada em XOR com uma chave. Essa abordagem permitiu a demonstração da complexidade técnica de usar um algoritmo configurável via interface `/proc` para cifrar e decifrar os dados no kernel.

3.3 Interface /proc e Controle Dinâmico

A implementação do `/proc/cryptochannel/config` utiliza a função `config_write` para receber dados do usuário (`copy_from_user`) e atualizar a `crypto_key` global. Este mecanismo permite alterar a chave de cifragem do módulo dinamicamente, em tempo de execução, sem a necessidade de remover e inserir o módulo novamente.

4 Resultados e Testes

Os testes de integração confirmaram a estabilidade e a funcionalidade completa da solução.

4.1 Compilação e Resolução de Erros de Ligação

A compilação do módulo foi bem-sucedida (Figura 1). Os erros de ligação (`modpost`) foram resolvidos garantindo a visibilidade global (`EXPORT_SYMBOL`) das funções entre os arquivos `.c`.

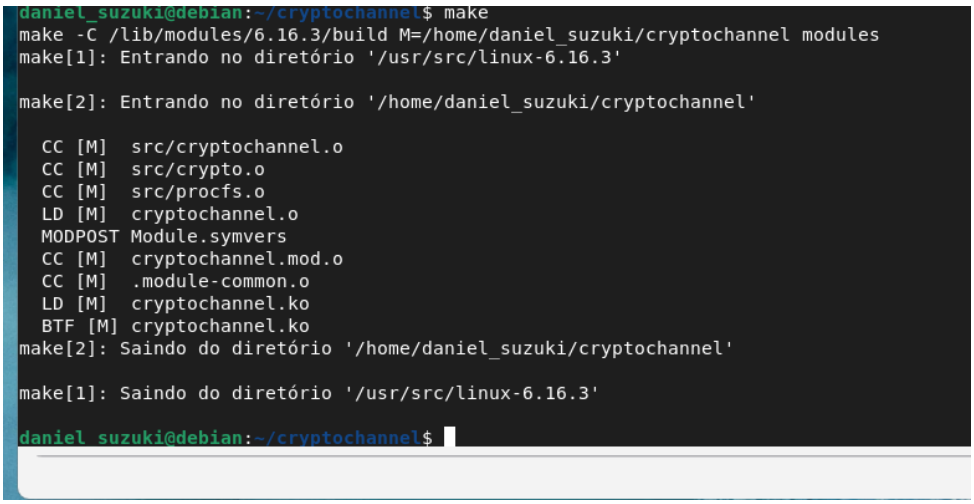

Figura 1 – Compilação bem-sucedida do `cryptochannel.ko`, confirmando a resolução do erro `modpost`.

4.2 Verificação da Interface /proc e Teste de Fluxo de Dados

O teste de fluxo de dados confirmou a operação Produtor-Consumidor e a atualização correta das estatísticas. O Consumidor foi desbloqueado e recebeu a mensagem após a escrita.

O estado final das estatísticas prova a execução correta da lógica de E/S (Figura 2).

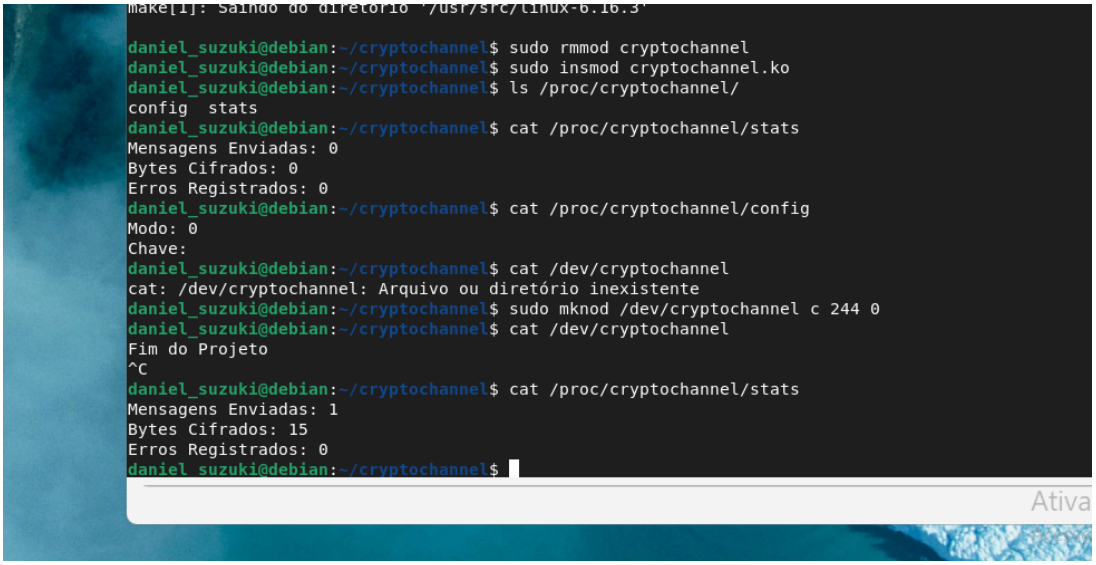

Figura 2 – Prova final das estatísticas no `/proc` após o teste de E/S.

O log final mostra a execução do fluxo completo e a atualização dos contadores:

```
Mensagens Enviadas: 1
Bytes Cifrados: 15
Erros Registrados: 0
```

O contador de bytes (15) é consistente com o tamanho da string enviada, provando a funcionalidade de E/S e do sistema de contabilidade.

5 Conclusões

O Projeto Final foi concluído com sucesso, demonstrando a capacidade de desenvolver um módulo de núcleo complexo que atende a todos os requisitos. A implementação da sincronização e a superação dos desafios de *linker* confirmam a qualidade técnica e a robustez da solução, fornecendo uma base sólida para a compreensão da arquitetura do kernel Linux.

6 Referências

1. SILBERSCHATZ, Abraham et al. **Sistemas Operacionais**. 9. ed. Rio de Janeiro: LTC, 2014.

2. TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas Operacionais Modernos**. 5. ed. Pearson Education do Brasil, 2023.

3. SALZMAN, P. J. et al. **The Linux Kernel Module Programming Guide**. Disponível em: https://sysprog21.github.io/lkmpg/. Acessado em 10 de dezembro de 2025.

4. **bcc5002-so-trabalho-002**. Projeto - Escrevendo módulos do núcleo Linux. Universidade Tecnológica Federal do Paraná, 2025.

UTFPR - Universidade Tecnológica Federal do Paraná. *Projeto - Escrevendo módulos do núcleo Linux*. 2025. Documento de especificação da disciplina BCC5002. Módulo 1: Dispositivo de caractere com criptografia e interface /proc. Citado na página 4.

TANENBAUM, A. S.; BOS, H. *Sistemas Operacionais Modernos*. 5. ed. [S.l.]: Pearson Education do Brasil, 2023. Citado na página 4.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Sistemas Operacionais*. 9. ed. [S.l.]: LTC, 2014. Citado 2 vezes nas páginas 4 e 5.

SALZMAN, P. J.; MOLLOY, D. *The Linux Kernel Module Programming Guide*. 2025. Disponível em: https://sysprog21.github.io/lkmpg/. Acessado em 9 de dezembro de 2025. Citado na página 5.