

Projeto de Pesquisa e Implementação

Implementação de Núcleo de Sistemas Operacionais

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

2025

Introdução

Módulos do kernel representam uma abordagem poderosa e flexível no design de Sistemas Operacionais. Um módulo do kernel é um fragmento de código que pode ser carregado e descarregado dinamicamente no kernel em tempo de execução. Isso permite que o kernel seja estendido sem a necessidade de recompilar todo o kernel.

Módulos do Kernel são tipicamente escritos em C ou C++ e compilados diretamente para o espaço de endereço do kernel. Essa modularidade aumenta significativamente a adaptabilidade, reduz o tempo de desenvolvimento para recursos especializados e fornece um ambiente controlado para testar e depurar alterações em serviços críticos do sistema operacional.

No Linux, você pode executar `lsmod` para listar todos os módulos carregados no sistema:

```
$ lsmod
Module                Size  Used by
tls                   110592  0
binfmt_misc           24576   1
intel_rapl_msrmr       20480   0
intel_rapl_common      40960   1 intel_rapl_msrmr
snd_hda_codec_generic 102400   1
ledtrig_audio          16384   1 snd_hda_codec_generic
kvm_intel              421888   0
...
```

E você pode usar `modinfo` para exibir informações sobre um módulo. Por exemplo, exiba as informações do módulo `tls`:

```
$ modinfo tls
filename:      /lib/modules/5.19.12-os-0816164/kernel/net/tls/tls.ko
alias:         tcp-ulp-tls
alias:         tls
license:       Dual BSD/GPL
description:   Transport Layer Security Support
author:        Mellanox Technologies
srcversion:    CA655CA00B96B66949E2221
...
```

Uma coisa para ter em mente é que um módulo do kernel existe no espaço do kernel e não pode ser escrito da mesma forma que um programa C normal. Isso ocorre porque as funções da biblioteca padrão do C, como

`printf` e `fopen`, não existem no kernel. Da mesma forma, estruturas como `FILE` e `wchar_t` também não estão disponíveis no kernel.

Objetivos de Aprendizagem

Ao concluir este projeto, você será capaz de:

- Compreender a arquitetura de um módulo do kernel Linux.
- Implementar módulos personalizados que realizam tarefas diversas.
- Compreender e aplicar os mecanismos de carregamento/descarregamento dos módulos do kernel Linux.
- Colaborar eficazmente em um contexto de equipe para alcançar um objetivo complexo de desenvolvimento de software.

Descrição

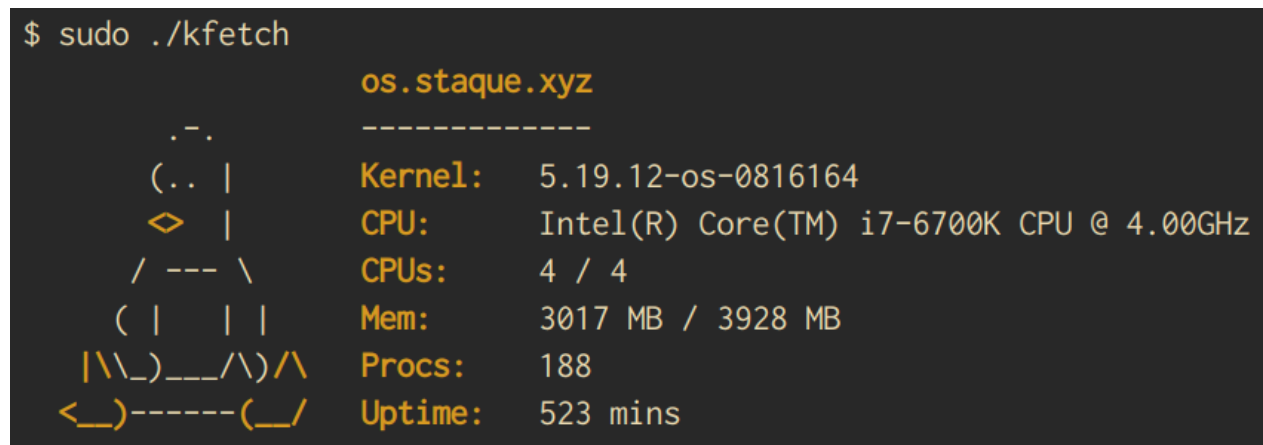
Neste projeto, você desenvolverá módulos customizados para o ambiente Ubuntu/Linux, explorando mecanismos de carregamento/descarregamento que sustentam o design moderno dos sistemas operacionais. O objetivo principal é preencher a lacuna entre o conhecimento teórico e a aplicação prática em programação de nível de sistema.

Para aprender a implementar um módulo do kernel, leia o livro *The Linux Kernel Module Programming Guide*, o qual fornece um guia abrangente para escrever módulos do kernel.

Módulo 1: Informações do Sistema

Você deve implementar um módulo do kernel chamado `kfetch_mod`: um driver de dispositivo de caractere que cria um dispositivo chamado `/dev/kfetch`. O programa do espaço do usuário `kfetch` pode recuperar as informações do sistema lendo deste dispositivo.

O módulo deve produzir uma saída semelhante a:



```
$ sudo ./kfetch

      .-.
     (.. |
      <  |
    / --- \
   ( |   | |
   ||\_)___/\) /\
  <__)------(_/

                                os.staque.xyz
                                -----
                                Kernel:   5.19.12-os-0816164
                                CPU:      Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
                                CPUs:     4 / 4
                                Mem:      3017 MB / 3928 MB
                                Procs:    188
                                Uptime:   523 mins
```

Figura 1: Módulo Customizado `kfetch_mod`

A lista das informações que seu módulo do kernel deve recuperar é a seguinte:

- Kernel: A versão do kernel
- CPU: O nome do modelo da CPU
- CPUs: O número de núcleos da CPU, no formato `<# de CPUs online> / <# de CPUs totais>`
- Mem: As informações de memória, no formato `/` (em MB)
- Proc: O número de processos

- Uptime: Quanto tempo o sistema tem sido em execução, em minutos.

Máscara de Informação Kfetch

Uma máscara de informação Kfetch é uma máscara de bits que determina quais informações exibir. Cada peça de informação é atribuída a um número, que corresponde a um bit em uma determinada posição.

```
#define KFETCH_NUM_INFO 6

#define KFETCH_RELEASE (1 << 0)
#define KFETCH_NUM_CPUS (1 << 1)
#define KFETCH_CPU_MODEL (1 << 2)
#define KFETCH_MEM (1 << 3)
#define KFETCH_UPTIME (1 << 4)
#define KFETCH_NUM_PROCS (1 << 5)

#define KFETCH_FULL_INFO ((1 << KFETCH_NUM_INFO) - 1);
```

A máscara é definida usando operações bit a bit OR nos bits relevantes. Por exemplo, para exibir o nome do modelo da CPU e as informações de memória, um usuário definiria a máscara assim: `mask = KFETCH_CPU_MODEL | KFETCH_MEM`.

Operações de dispositivo

Seu driver de dispositivo deve suportar quatro operações: `open`, `release`, `read` e `write`.

```
const static struct file_operations kfetch_ops = {
    .owner    = THIS_MODULE,
    .read     = kfetch_read,
    .write    = kfetch_write,
    .open     = kfetch_open,
    .release  = kfetch_release,
};
```

Para a operação de leitura, você precisa retornar um buffer que contenha o conteúdo do logotipo e as informações para o espaço do usuário. Isso permite que o usuário acesse e use os dados do dispositivo.

```
static ssize_t kfetch_read(struct file *filp,
                          char __user *buffer,
                          size_t length,
                          loff_t *offset)
{
    /* recuperando as informações */

    if (copy_to_user(buffer, kfetch_buf, len)) {
        pr_alert("Falha ao copiar dados para o usuário");
        return 0;
    }

    /* limpeza */
}
```

Para a operação de escrita, um único inteiro representando a máscara de informação que o usuário deseja definir é passado para o driver de dispositivo. As operações de leitura subsequentes usarão esta máscara para determinar quais informações devem ser retornadas ao usuário. Isso permite que o usuário especifique quais informações eles desejam receber e o driver de dispositivo pode usar a máscara para garantir que apenas as informações especificadas sejam retornadas.

```
static ssize_t kfetch_write(struct file *filp,
                          const char __user *buffer,
```

```

        size_t length,
        loff_t *offset)
{
    int mask_info;

    if (copy_from_user(&mask_info, buffer, length)) {
        pr_alert("Falha ao copiar dados do usuário");
        return 0;
    }

    /* definindo a máscara de informação */
}

```

Para as operações **open** e **release**, você precisa configurar e limpar as proteções, pois em um ambiente multithread, o acesso concorrente à mesma memória pode levar a condições de corrida. Essas proteções podem assumir a forma de bloqueios, semáforos ou outros mecanismos de sincronização que garantam que apenas uma thread possa acessar as variáveis ao mesmo tempo.

Requisitos

- As operações **open** e **release** devem configurar e limpar as proteções adequadamente.
- A operação de escrita deve definir a máscara de informação no módulo, que determina quais dados são retornados pela operação de leitura.
- A operação de leitura deve retornar dados que incluam:
 - Um logotipo de sua escolha
 - A primeira linha é o nome do host da máquina, que é obrigatório e não pode ser desabilitado
 - A próxima linha é uma linha separadora com um comprimento igual ao nome do host
 - As linhas restantes dependem da máscara de informação
- O suporte a cores é opcional.
- Você deve liberar recursos como memória alocada e o número principal/menor do dispositivo quando o módulo é removido.

Dica

- O Linux usa o sistema de arquivos **proc** para exportar informações sobre o sistema, localizado no diretório **/proc**.
- Por exemplo, as informações de memória podem ser encontradas em **/proc/meminfo**.
- O código-fonte pode ser encontrado em **fs/proc**. Você pode visualizar o código-fonte para ver como as informações são recuperadas.
- Para o nome do host e a liberação, você pode querer ver **uts_namespaces(7)**.
- Para o número de threads, você pode querer ver a variável **nr_threads**.

Módulo 2: Sistema de Pontuação de Comportamento de Processos

Implemente um módulo do kernel projetado para monitorar continuamente as principais métricas para cada processo em execução: uso da CPU, chamadas de sistema, atividade de E/S, potencialmente tráfego de rede, etc. O módulo então analisará essas métricas para atribuir uma nota de risco – Baixo, Médio ou Alto. Essa avaliação de risco seria acionada pelo exceder limites de recursos ou exibir comportamento anômalo.

Observação: Você deve propor o seu próprio algoritmo de avaliação de risco baseado nas informações que puder coletar. Seja criativo!

O módulo deve produzir os resultados da avaliação de risco – incluindo a nota de risco atribuída e as métricas relevantes – em um arquivo específico dentro do sistema de arquivos **/proc**. Isso permite uma fácil visualização do comportamento dos processos usando ferramentas padrão como **cat /proc/<pid>/stat**.

Entrega

- **Relatório do Projeto:** Um relatório detalhado documentando o projeto, incluindo:
 - Racional de design por trás das escolhas do módulo.
 - Detalhes técnicos da implementação.
 - Resultados dos testes e procedimentos de validação.
 - Discussão de desafios encontrados e soluções implementadas.
- **Código Fonte:** O código fonte completo para o módulo do kernel, devidamente documentado com comentários.
- **Instruções & README:** Instruções claras sobre como construir (build) e carregar o módulo em um sistema Ubuntu Linux.

Critérios de Avaliação

Qualidade do Relatório: Clareza, organização e qualidade geral do relatório escrito.

Qualidade do Código: Correção, manutenibilidade e adesão aos padrões de codificação.

Funcionalidade & Testes: Implementação bem-sucedida da funcionalidade principal e resultados abrangentes de testes.

Avaliação entre Pares: Evidência de colaboração eficaz e contribuição para o esforço da equipe (avaliada por meio de avaliações entre pares).