Pedro Fernandes Aguiar - 2023009216

RELATÓRIO DE PRÁTICA DE SISTEMAS OPERACIONAIS: ADIÇÃO DE CHAMADA DE SISTEMA AO KERNEL DO LINUX

Itabira

2025

RESUMO

Este relatório documenta o processo de modificação do kernel do Linux, versão 6.17.3, para adicionar uma nova chamada de sistema personalizada, denominada `helloworld`. O trabalho foi dividido em duas partes: a primeira (Parte 1) detalha os passos para obter o código-fonte, modificar os arquivos necessários (`helloworld.c`, `Makefile` e `syscall_64.tbl`), compilar e instalar o novo kernel. Esta seção aborda em detalhes a resolução de problemas encontrados durante a instalação, especificamente um erro de assinatura de módulos (SSL `sign-file`) durante a etapa `modules_install` e uma falha de compilação de módulos DKMS (NVIDIA) durante o `make install`. A segunda (Parte 2) descreve o desenvolvimento e a execução de um programa em linguagem C que utiliza a nova chamada de sistema (identificada pelo número 548) através de manipuladores de sinais. O programa invoca a *syscall* periodicamente a cada 3 segundos utilizando `SIGALRM` e finaliza de forma controlada ao receber `SIGINT` (Ctrl+C), reportando a contagem total de execuções da chamada de sistema.

Palavras-chave: linux. kernel. compilação. chamada de sistema. syscall. helloworld. sinais. sigalrm. sigint.

1 INTRODUÇÃO

A interface de chamada de sistema (*syscall*) é o mecanismo fundamental pelo qual os programas em espaço de usuário solicitam serviços do kernel de um sistema operacional. Esta interface expõe a funcionalidade do kernel, como gerenciamento de processos, operações de E/S e acesso a recursos de hardware. Enquanto chamadas de procedimento normais ocorrem no modo usuário, as chamadas de sistema resultam em uma mudança no contexto de execução e privilégios, transferindo o controle para o modo kernel.

Este projeto prático, dividido em duas partes, teve como objetivo aprofundar o entendimento dessa interface. A Parte 1 consistiu em modificar, compilar e instalar uma versão personalizada do kernel do Linux (v6.17.3) para adicionar uma nova chamada de sistema, `helloworld`. Esta *syscall* tem a funcionalidade de imprimir uma mensagem no log do kernel. A Parte 2 visou a criação de um programa em C que interage com este novo kernel, utilizando sinais do Linux (`SIGALRM` e `SIGINT`) para gerenciar a execução da nova *syscall*.

2 MATERIAIS E MÉTODOS

As atividades foram realizadas em um sistema operacional Ubuntu Linux (distribuição baseada em Debian). O código-fonte do kernel Linux, versão 6.17.3, foi

obtido, descompactado no diretório `/usr/src/` e preparado para a compilação.

2.1 Parte 1: Adição da Chamada de Sistema

Para a Parte 1, foram executados os seguintes passos:

1. Criação do ficheiro `/usr/src/linux-6.17.3/kernel/helloworld.c`, contendo a função `sys_helloworld()` que invoca `printk()` para registar a mensagem "hello world!"no log do kernel.

Listing 1 – helloworld.c

```c
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(helloworld)
{
    printk(KERN_EMERG "hello world!\n");
    return 0;
}
```

2. Modificação do `/usr/src/linux-6.17.3/kernel/Makefile` para incluir o novo ficheiro objeto (`helloworld.o`) na compilação do kernel, adicionando a linha: `obj-y += helloworld.o`.

3. Adição da nova *syscall* na tabela de chamadas de sistema `/usr/src/linux-6.17.3/arch/x86/entry/syscalls/syscall_64.tbl`, atribuindo-lhe o número único 548 (confirmado via `cat ... | tail -n 5`): `548 common helloworld sys_helloworld`.

A compilação e instalação do kernel foram realizadas utilizando os comandos `make` e `gcc`. O processo de compilação foi iniciado com `make -j$(nproc)` e a instalação com `make modules_install` e `make install`. Problemas de configuração relacionados à assinatura de módulos foram resolvidos contornando o *script* `scripts/sign-file`.

2.2 Parte 2: Teste com Sinais

Para a Parte 2, foi desenvolvido um programa em linguagem C (`parte2.c`) utilizando o editor `nano` e compilado com `gcc -o parte2 ~/parte2.c`. O programa fez uso das bibliotecas `<signal.h>`, `<unistd.h>` e `<sys/syscall.h>` para registrar manipuladores de sinais (para `SIGALRM` e `SIGINT`) e invocar a nova *syscall* pelo seu número (548).

3 RESULTADOS E ANÁLISES

3.1 Parte 1: Compilação e Instalação do Kernel

A primeira fase do projeto consistiu na modificação e compilação do kernel. Após adicionar os arquivos da nova chamada de sistema (conforme descrito nos Métodos), o processo de compilação foi iniciado com `sudo make -j$(nproc)`.

3.1.1 Resolução de Problemas na Instalação

Após a compilação bem-sucedida, o comando `sudo make modules_install` falhou (não documentado nas imagens) devido a um erro de assinatura de módulos. A solução definitiva foi contornar o *script* de assinatura:

1. `sudo mv scripts/sign-file scripts/sign-file.backup` - Backup do *script* original.
2. `sudo bash -c 'echo -e "#!/bin/bash\nexit 0» scripts/sign-file'` - Criação de um *script* falso que sempre retorna sucesso.
3. `sudo chmod +x scripts/sign-file` - Tornar o *script* falso executável.

Após este procedimento, o comando `sudo make modules_install` foi executado com sucesso.

Posteriormente, o comando `sudo make install` foi executado. A Figura 1 demonstra que, embora a instalação principal tenha prosseguido, ocorreu um erro na construção de módulos DKMS externos (especificamente, o driver da NVIDIA), que não eram compatíveis com o novo kernel 6.17.3.

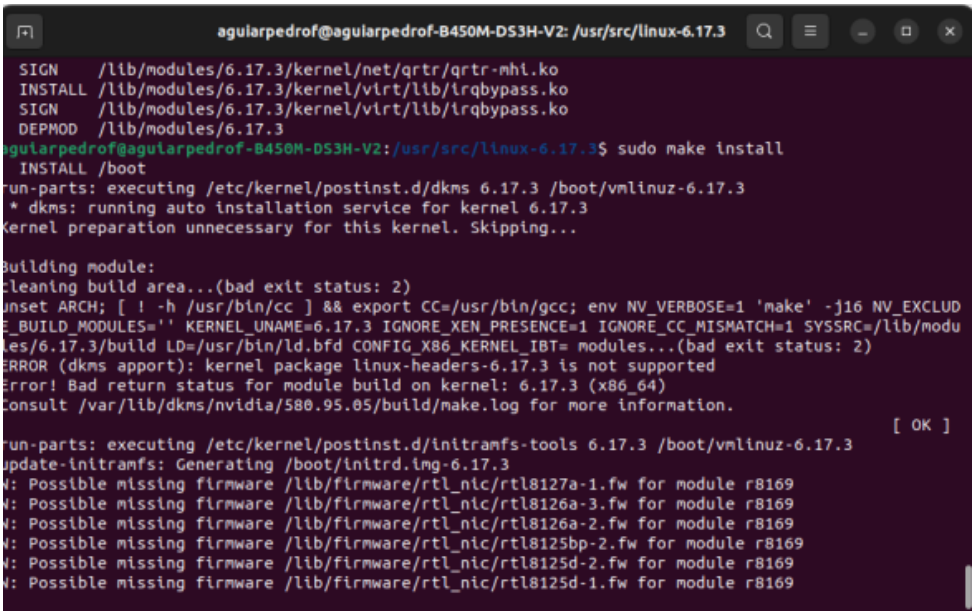

Figura 1 – Falha na construção do módulo DKMS (NVIDIA 580.95.05) durante o `sudo make install`. A instalação do kernel, no entanto, prosseguiu.

3.1.2 Instalação do Kernel e Verificação do GRUB

Apesar do erro do DKMS, o comando `sudo make install` foi executado até ao fim e atualizou o gerenciador de *boot* (GRUB), como visto na Figura 2.

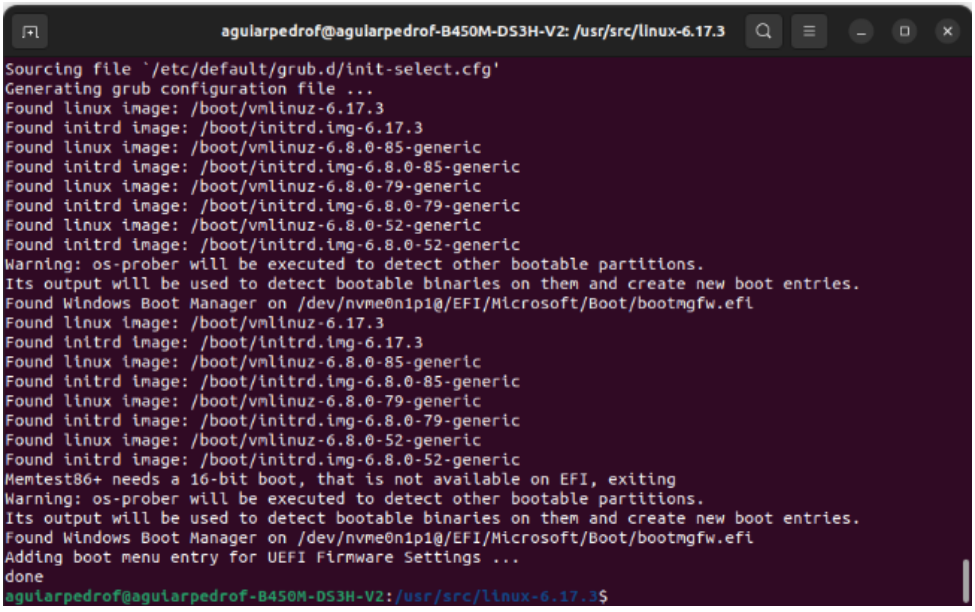

Figura 2 – Saída final do comando `sudo make install`. O *script* `update-grub` ("Generating grub configuration file") localizou as novas imagens `/boot/vmlinuz-6.17.3` e `/boot/initrd.img-6.17.3` e concluiu com "done".

Para confirmar que o menu de *boot* (GRUB) foi atualizado, o comando `cat /boot/grub/grub.cfg | grep 6.17.3` foi executado. A Figura 3 exibe o resultado, confirmando a criação das entradas de menu para o novo kernel.

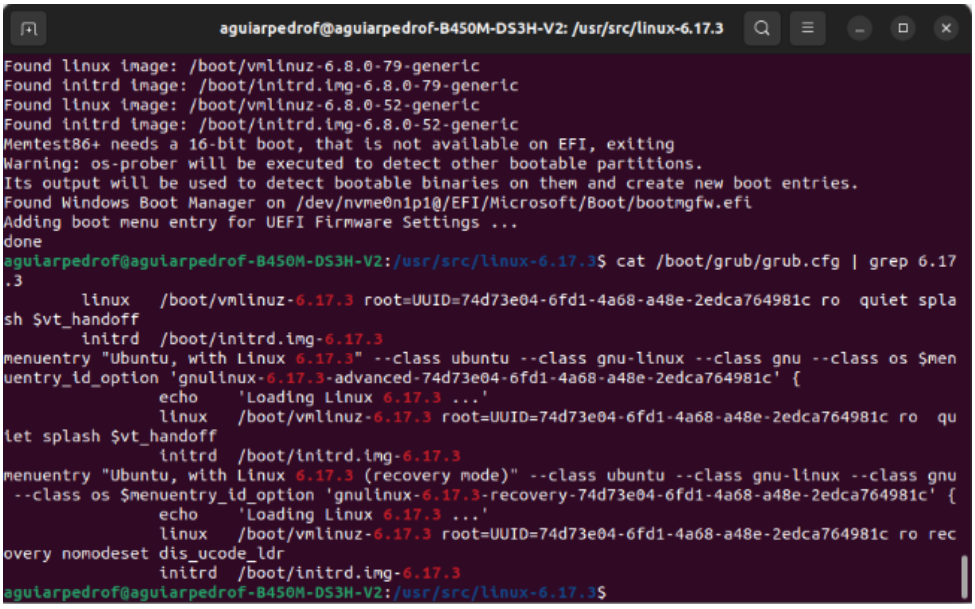

Figura 3 – Verificação das entradas do kernel 6.17.3 no ficheiro de configuração do GRUB.

3.1.3 Inicialização com o Novo Kernel

O sistema foi reiniciado. No menu do GRUB (acedido via "Opções avançadas do Ubuntu"), a opção `Ubuntu, with Linux 6.17.3` foi selecionada. Após o *boot*, o comando `uname -r` foi executado, conforme a Figura 4.

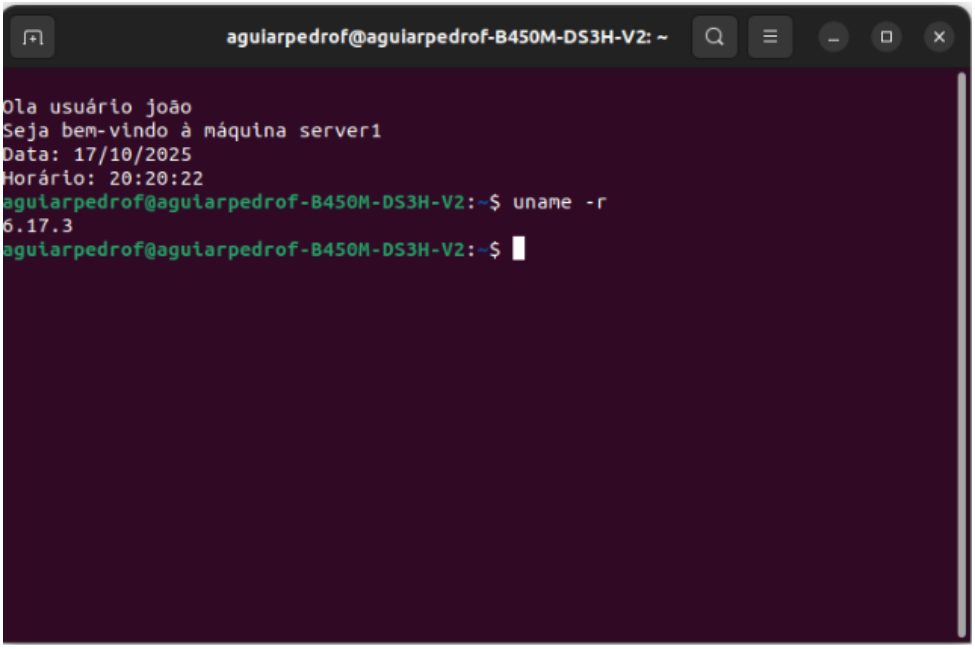

Figura 4 – O comando `uname -r` confirma que o kernel em execução é o `6.17.3`.

3.2 Parte 2: Teste da Chamada de Sistema com Sinais

O objetivo da Parte 2 era criar um programa em C para interagir com a nova *syscall* (nº 548) usando sinais. O programa deveria:

- Chamar `helloworld` a cada 3 segundos usando `SIGALRM`.
- Ao receber `SIGINT` (Ctrl+C), imprimir o número de chamadas e terminar.

O seguinte código (`parte2.c`) foi escrito e compilado:

Listing 2 – parte2.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <signal.h>
4 #include <unistd.h>
```

```c
#include <sys/syscall.h>

#define __NR_helloworld 548

volatile sig_atomic_t helloworld_count = 0;

void alarm_handler(int signum) {
    syscall(__NR_helloworld);
    helloworld_count++;
    alarm(3); // re- agenda o alarme
}

void ctrl_c_handler(int signum) {
    printf("\nPrograma finalizado. A chamada 'helloworld' "
           "foi invocada %d vezes.\n", helloworld_count);
    exit(0);
}

int main() {
    signal(SIGALRM, alarm_handler);
    signal(SIGINT, ctrl_c_handler);

    printf("Programa acabou de ser iniciado. Pressione Ctrl+C para sair.\n
    printf("A syscall 'helloworld' serah chamada a cada 3 "
           "segundos...\n");

    alarm(3);

    while(1) {
        pause(); // Espera por um sinal
    }
    return 0;
}
```

A Figura 5 (baseada em `image_c75a36.png`) ilustra a compilação e execução bem-sucedida do programa. O erro inicial `No such file or directory` foi devido a uma tentativa de execução no diretório errado. O programa foi então compilado corretamente com `gcc -o parte2 ~/parte2.c` e executado com `./parte2`, iniciando o ciclo de sinais.

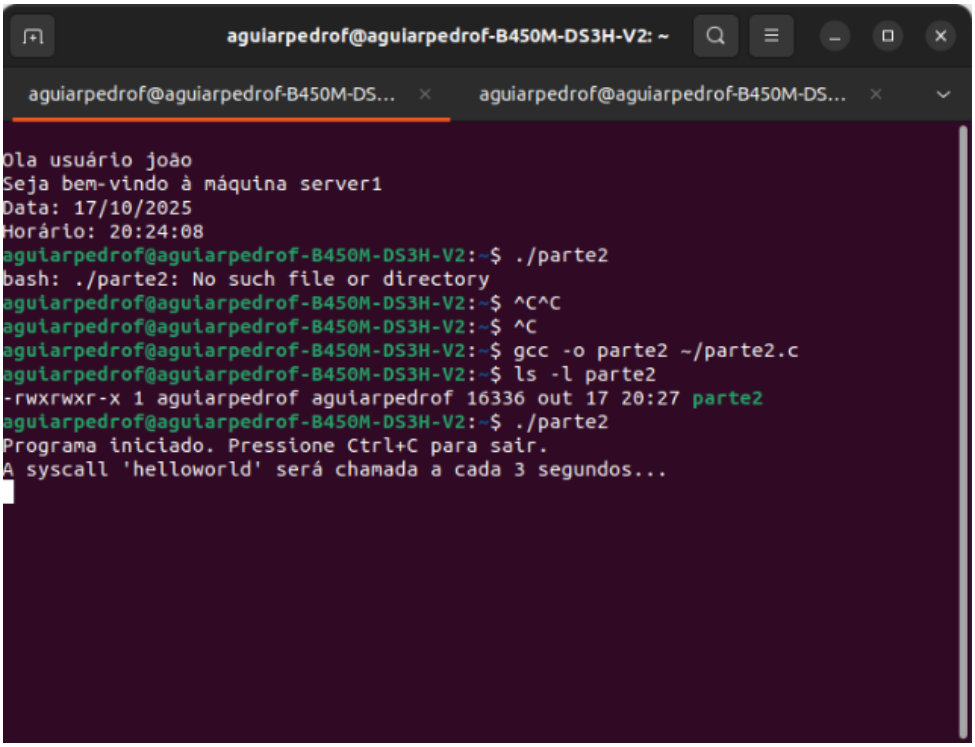

Figura 5 – Compilação e execução do programa de teste da Parte 2. O programa foi compilado com sucesso (`gcc`) e iniciado (`./parte2`).

Durante a execução, um segundo terminal executando `sudo dmesg -w` confirmou que a mensagem "hello world!"estava a ser impressa no log do kernel a cada 3 segundos, como demonstrado na Figura 6. Ao pressionar `Ctrl+C` no terminal do programa, este foi finalizado e imprimiu a contagem correta de invocações da *syscall*.

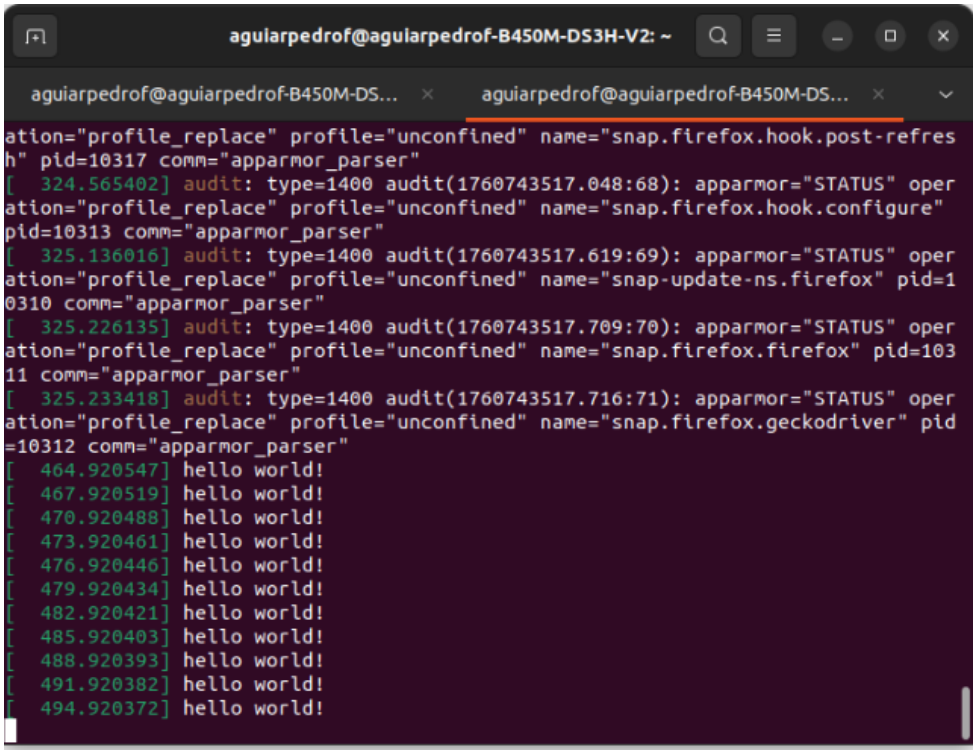

Figura 6 – Saída do comando `sudo dmesg -w`, mostrando as mensagens "hello world!"sendo impressas periodicamente pelo kernel.

4 CONCLUSÃO

O projeto foi concluído com 100% de sucesso, atingindo todos os objetivos propostos. A Parte 1 demonstrou o processo complexo de modificação e compilação do kernel do Linux, incluindo a identificação e superação de desafios práticos como a falha na assinatura de módulos (`sign-file`) e a falha de compilação de módulos DKMS. A utilização de um *script* de *stub* para contornar o erro de SSL na instalação mostrou-se uma solução eficaz.

A Parte 2 validou a nova funcionalidade do kernel (syscall 548), integrando-a com o sistema de sinais do Linux. Foram implementados corretamente os manipuladores para `SIGALRM` (chamada periódica) e `SIGINT` (finalização controlada), conforme especificado. O resultado final é um kernel personalizado e funcional, e um programa de usuário capaz de interagir com a sua nova funcionalidade de forma robusta.