

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

```
1 #include <linux/kernel.h>
2 #include <linux/syscalls.h>
3
4 SYSCALL_DEFINE0(helloworld)
5 {
6     printk(KERN_EMERG "hello world!\n");
7     return 0;
8 }
```

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.

```
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: /usr/src/linux-6.17.3
SIGN /lib/modules/6.17.3/kernel/net/qrtr/qrtr-mhi.ko
INSTALL /lib/modules/6.17.3/kernel/virt/lib/lrqbypass.ko
SIGN /lib/modules/6.17.3/kernel/virt/lib/lrqbypass.ko
DEPMOD /lib/modules/6.17.3
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:/usr/src/linux-6.17.3$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/dkms 6.17.3 /boot/vmlinuz-6.17.3
* dkms: running auto installation service for kernel 6.17.3
Kernel preparation unnecessary for this kernel. Skipping...

Building module:
Cleaning build area...(bad exit status: 2)
unset ARCH; [ ! -h /usr/bin/cc ] && export CC=/usr/bin/gcc; env NV_VERBOSE=1 'make' -j16 NV_EXCLUDE_BUILD_MODULES='' KERNEL_UNAME=6.17.3 IGNORE_XEN_PRESENCE=1 IGNORE_CC_MISMATCH=1 SYSSRC=/lib/modules/6.17.3/build LD=/usr/bin/ld.bfd CONFIG_X86_KERNEL_IBT= modules...(bad exit status: 2)
ERROR (dkms apport): kernel package linux-headers-6.17.3 is not supported
Error! Bad return status for module build on kernel: 6.17.3 (x86_64)
consult /var/lib/dkms/nvidia/580.95.05/build/make.log for more information.

[ OK ]

run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.17.3 /boot/vmlinuz-6.17.3
update-initramfs: Generating /boot/initrd.img-6.17.3
W: Possible missing firmware /lib/firmware/rtl_nic/rtl8127a-1.fw for module r8169
W: Possible missing firmware /lib/firmware/rtl_nic/rtl8126a-3.fw for module r8169
W: Possible missing firmware /lib/firmware/rtl_nic/rtl8126a-2.fw for module r8169
W: Possible missing firmware /lib/firmware/rtl_nic/rtl8125bp-2.fw for module r8169
W: Possible missing firmware /lib/firmware/rtl_nic/rtl8125d-2.fw for module r8169
W: Possible missing firmware /lib/firmware/rtl_nic/rtl8125d-1.fw for module r8169
```

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.

```
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: /usr/src/linux-6.17.3
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.17.3
Found initrd image: /boot/initrd.img-6.17.3
Found linux image: /boot/vmlinuz-6.8.0-85-generic
Found initrd image: /boot/initrd.img-6.8.0-85-generic
Found linux image: /boot/vmlinuz-6.8.0-79-generic
Found initrd image: /boot/initrd.img-6.8.0-79-generic
Found linux image: /boot/vmlinuz-6.8.0-52-generic
Found initrd image: /boot/initrd.img-6.8.0-52-generic
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/nvme0n1p1/EFI/Microsoft/Boot/bootmgfw.efi
Found linux image: /boot/vmlinuz-6.17.3
Found initrd image: /boot/initrd.img-6.17.3
Found linux image: /boot/vmlinuz-6.8.0-85-generic
Found initrd image: /boot/initrd.img-6.8.0-85-generic
Found linux image: /boot/vmlinuz-6.8.0-79-generic
Found initrd image: /boot/initrd.img-6.8.0-79-generic
Found linux image: /boot/vmlinuz-6.8.0-52-generic
Found initrd image: /boot/initrd.img-6.8.0-52-generic
Mentest86+ needs a 16-bit boot, that is not available on EFI, exiting
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/nvme0n1p1/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: /usr/src/linux-6.17.3$
```

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.

```
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: /usr/src/linux-6.17.3
Found linux image: /boot/vmlinuz-6.8.0-79-generic
Found initrd image: /boot/initrd.img-6.8.0-79-generic
Found linux image: /boot/vmlinuz-6.8.0-52-generic
Found initrd image: /boot/initrd.img-6.8.0-52-generic
Mentest86+ needs a 16-bit boot, that is not available on EFI, exiting
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/nvme0n1p1/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
done
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: /usr/src/linux-6.17.3$ cat /boot/grub/grub.cfg | grep 6.17
.3
        linux /boot/vmlinuz-6.17.3 root=UUID=74d73e04-6fd1-4a68-a48e-2edca764981c ro quiet spla
sh $vt_handoff
        initrd /boot/initrd.img-6.17.3
menuentry "Ubuntu, with Linux 6.17.3" --class ubuntu --class gnu-linux --class gnu --class os $men
uentry_id_option 'gnulinux-6.17.3-advanced-74d73e04-6fd1-4a68-a48e-2edca764981c' {
    echo 'Loading Linux 6.17.3 ...'
    linux /boot/vmlinuz-6.17.3 root=UUID=74d73e04-6fd1-4a68-a48e-2edca764981c ro qu
let splash $vt_handoff
        initrd /boot/initrd.img-6.17.3
menuentry "Ubuntu, with Linux 6.17.3 (recovery mode)" --class ubuntu --class gnu-linux --class gnu
--class os $menuentry_id_option 'gnulinux-6.17.3-recovery-74d73e04-6fd1-4a68-a48e-2edca764981c' {
    echo 'Loading Linux 6.17.3 ...'
    linux /boot/vmlinuz-6.17.3 root=UUID=74d73e04-6fd1-4a68-a48e-2edca764981c ro rec
overy nomodeset dis_ucode_ldr
        initrd /boot/initrd.img-6.17.3
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: /usr/src/linux-6.17.3$
```

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.

A terminal window with a dark purple background. The title bar shows the user 'aguiarpedrof' on a machine named 'aguiarpedrof-B450M-DS3H-V2'. The terminal output shows a login message for user 'joão', the date '17/10/2025', and the time '20:20:22'. The user then enters the command 'uname -r', and the output is '6.17.3'.

```
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: ~  
Ola usuário joão  
Seja bem-vindo à máquina server1  
Data: 17/10/2025  
Horário: 20:20:22  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$ uname -r  
6.17.3  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$
```

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`

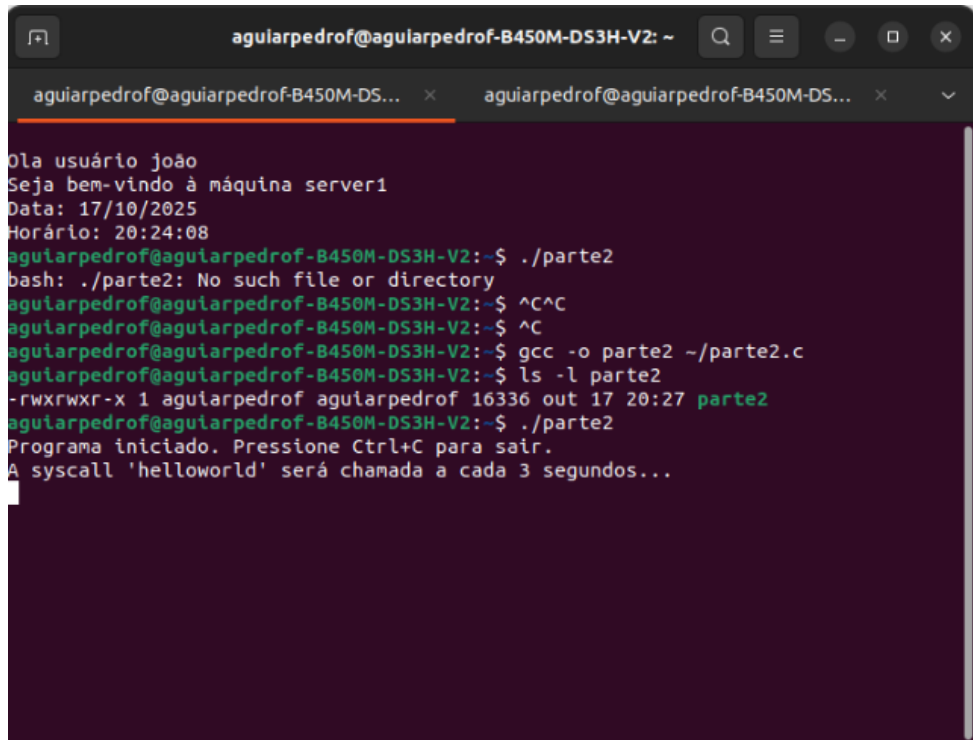
```
1 #include <stdio.h>  
2 #include <stdlib.h>  
3 #include <signal.h>  
4 #include <unistd.h>
```

```

5  #include <sys/syscall.h>
6
7  #define __NR_helloworld 548
8
9  volatile sig_atomic_t helloworld_count = 0;
10
11 void alarm_handler(int signum) {
12     syscall(__NR_helloworld);
13     helloworld_count++;
14     alarm(3); // re- agenda o alarme
15 }
16
17 void ctrl_c_handler(int signum) {
18     printf("\nPrograma finalizado. A chamada 'helloworld' "
19           "foi invocada %d vezes.\n", helloworld_count);
20     exit(0);
21 }
22
23 int main() {
24     signal(SIGALRM, alarm_handler);
25     signal(SIGINT, ctrl_c_handler);
26
27     printf("Programa acabou de ser iniciado. Pressione Ctrl+C para sair.\n");
28     printf("A syscall 'helloworld' serah chamada a cada 3 "
29           "segundos...\n");
30
31     alarm(3);
32
33     while(1) {
34         pause(); // Espera por um sinal
35     }
36     return 0;
37 }

```

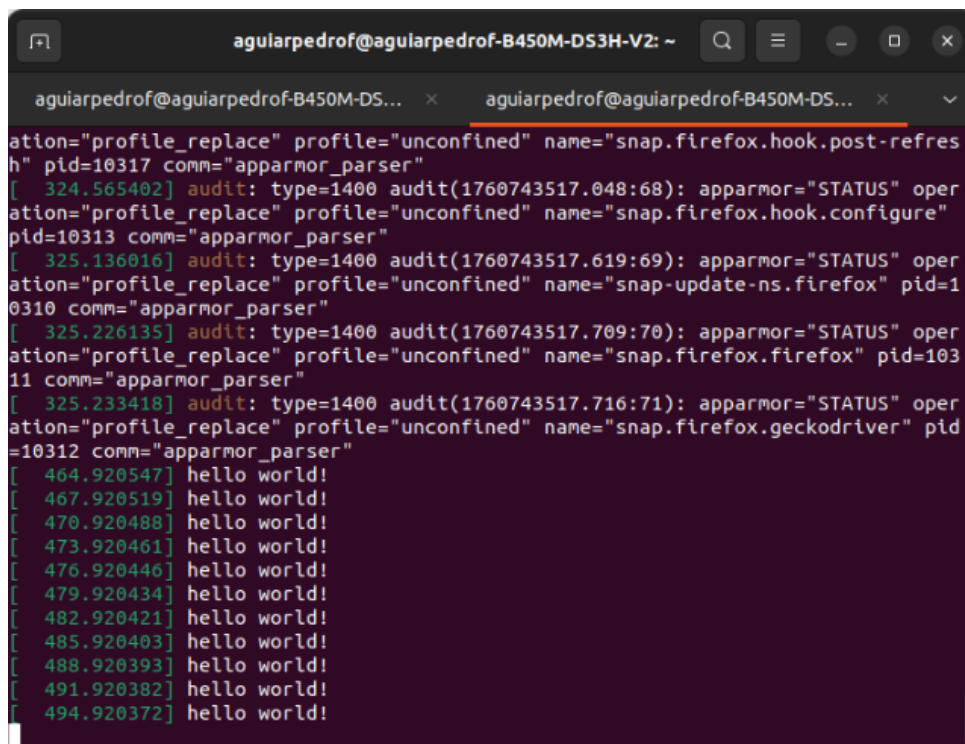
A Figura 5 (baseada em [image_c75a36.png](#)) ilustra a compilação e execução bem-sucedida do programa. O erro inicial `No such file or directory` foi devido a uma tentativa de execução no diretório errado. O programa foi então compilado corretamente com `gcc -o parte2 ~/parte2.c` e executado com `./parte2`,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



```
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: ~  
Ola usuário joão  
Seja bem-vindo à máquina server1  
Data: 17/10/2025  
Horário: 20:24:08  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$ ./parte2  
bash: ./parte2: No such file or directory  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$ ^C^C  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$ ^C  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$ gcc -o parte2 ~/parte2.c  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$ ls -l parte2  
-rwxrwxr-x 1 aguiarpedrof aguiarpedrof 16336 out 17 20:27 parte2  
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2:~$ ./parte2  
Programa iniciado. Pressione Ctrl+C para sair.  
A syscall 'helloworld' será chamada a cada 3 segundos...
```

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*.



```
aguiarpedrof@aguiarpedrof-B450M-DS3H-V2: ~  
aguiarpedrof@aguiarpedrof-B450M-DS... x aguiarpedrof@aguiarpedrof-B450M-DS... x  
ation="profile_replace" profile="unconfined" name="snap.firefox.hook.post-refres  
h" pid=10317 comm="apparmor_parser"  
[ 324.565402] audit: type=1400 audit(1760743517.048:68): apparmor="STATUS" oper  
ation="profile_replace" profile="unconfined" name="snap.firefox.hook.configure"  
pid=10313 comm="apparmor_parser"  
[ 325.136016] audit: type=1400 audit(1760743517.619:69): apparmor="STATUS" oper  
ation="profile_replace" profile="unconfined" name="snap-update-ns.firefox" pid=1  
0310 comm="apparmor_parser"  
[ 325.226135] audit: type=1400 audit(1760743517.709:70): apparmor="STATUS" oper  
ation="profile_replace" profile="unconfined" name="snap.firefox.firefox" pid=103  
11 comm="apparmor_parser"  
[ 325.233418] audit: type=1400 audit(1760743517.716:71): apparmor="STATUS" oper  
ation="profile_replace" profile="unconfined" name="snap.firefox.geckodriver" pid  
=10312 comm="apparmor_parser"  
[ 464.920547] hello world!  
[ 467.920519] hello world!  
[ 470.920488] hello world!  
[ 473.920461] hello world!  
[ 476.920446] hello world!  
[ 479.920434] hello world!  
[ 482.920421] hello world!  
[ 485.920403] hello world!  
[ 488.920393] hello world!  
[ 491.920382] hello world!  
[ 494.920372] hello world!
```

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.