Project Report

On

ADDING A SYSTEM CALL TO KERNEL

Submitted in partial fulfilment of the requirements for the award of

# BACHELOR OF TECHNOLOGY

**in**

# COMPUTER SCIENCE & ENGINEERING

(Artificial Intelligence & Machine Learning) by

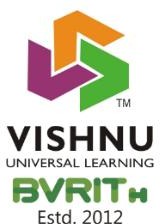
**Ms. ANKITHA.E - 23WH1A6605**

**Ms. AKSHITHA.N - 23WH1A6615**

**Ms. AMRUTHA VARSHINI.P - 23WH1A6632 Ms. S.RUTHVIKA - 23WH1A6663**

**Under the esteemed guidance of Ms. S Annapoorna**

**Assistant Professor, CSE(AI&ML)**

****

**BVRIT HYDERABAD College of Engineering for Women**

**(UGC Autonomous Institution | Approved by AICTE | Affiliated to JNTUH) (NAAC Accredited - A Grade | NBA Accredited B.Tech. (EEE, ECE, CSE and IT) Bachupally, Hyderabad – 500090**

2024-25

**Department of Computer Science & Engineering (Artificial Intelligence & Machine Learning)**

# BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad) Accredited by NBA and NAAC with A Grade**

**Bachupally, Hyderabad – 500090 2024-25**

****

# CERTIFICATE

This is to certify that the project entitled “ADDING A SYSTEM CALL TO THE LINUX KERNEL” is a Bonafide work carried out by the students listed below under the guidance of Ms.

ANNAPOORNA.S in partial fulfillment for the award of B.Tech degree in Computer Science & Engineering (AI & ML).

• 23WH1A6605 E. ANKITHA

• 23WH1A6615 N. AKSHITHA

• 23WH1A6632 P. AMRUTHA VARSHINI

• 23WH1A6663 B. RUTHVIKA

Supervisor Head of the Department

Ms.S.Annapoorna Dr. B. Lakshmi Praveena

Assistant Professor HOD & Professor

Dept of CSE(AI&ML) Dept of CSE(AI&ML)

# ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal, BVRIT HYDERABAD College of Engineering for Women**, for her support by providing the working facilities in the college

Our sincere thanks and gratitude **to Dr. B. Lakshmi Praveena, Head of the Department, Department of CSE(AI&ML), BVRIT HYDERABAD College of Engineering for Women,** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Ms. S.Annapoorna, Assistant Professor CSE(AI&ML), BVRIT HYDERABAD College of Engineering for Women,** for her constant guidance and encouragement throughout the project.

• 23WH1A6605 E. ANKITHA

• 23WH1A6615 N. AKSHITHA

• 23WH1A6632 P. AMRUTHA VARSHINI

• 23WH1A6663 B. RUTHVIKA

# DECLARATION

We hereby declare that the work presented in this project entitled “**ADDING A SYSTEM CALL TO KERNEL**” submitted towards completion of Project work in II Year of B.Tech of CSE(AI&ML) at **BVRIT HYDERABAD College of Engineering for Women, Hyderabad** is an authentic record of our original work carried out under the guidance of **Ms. S.Annapoorna, Assistant Professor, Department of CSE(AI&ML).**

**Sign with Date:**

**Ankitha.e**

**(23WH1A6605)**

**Sign with Date:**

**N.Akshitha**

**(23WH1A6615)**

**Sign with Date:**

**Amrutha Varshini.p**

**(23WH1A6632)**

**Sign with Date:**

**Ruthvika.S (23WH5A6663)**

# AIM

The primary objectives of this project are:

* To understand the Linux kernel architecture and system call mechanism
* To implement a custom system call that prints a message to the kernel log
* To learn the complete process of kernel modification and compilation
* To create a user-space test program to verify the system call
* To gain practical experience in Linux kernel development
* To analyze the performance impact of adding new system calls
* To understand the relationship between user space and kernel space
* To study system call numbering and registration process

# DESCRIPTION

## System Call Overview

System calls provide an essential interface between user applications and the Linux kernel. Our custom system call sys mycall performs a simple operation (printing a message to the kernel log) and returns a status value, demonstrating the complete lifecycle of a system call from implementation to invocation.

## System Architecture

* **Kernel Space**: Modified kernel source files to implement the new functionality
* **User Space**: Test program to invoke the system call
* **Interface**: System call table and header files that connect both spaces
* **Verification**: Kernel logs and return value validation

## Development Environment

* Ubuntu 20.04 LTS with Linux kernel 5.4.0
* GCC 9.3.0 compiler suite
* GNU Make 4.2.1 for kernel compilation
* VirtualBox 6.1 for safe kernel testing
* 8GB RAM, 4-core CPU allocation

## Methodology

1. Identified the appropriate files to modify in the kernel source
2. Implemented the system call function in kernel/sys.c
3. Updated system call tables and headers
4. Recompiled and installed the modified kernel
5. Created test programs to verify functionality
6. Analyzed system call performance using time command
7. Verified through kernel logs and return values

# IMPLEMENTATION DETAILS

## Step 1: Kernel Source Modification

* + Added custom function to kernel/sys.c:

// Simple system call implementation asmlinkage long sys\_mycall( void ) {

printk( KERN\_INFO " Custom system call executed from process % d\ n ",

current - >pid ); return 0;

}

1

2

3

4

5

6

* + Updated arch/x86/entry/syscalls/syscall 64.tbl:

# System call number

333

ABI Name

common mycall

Entry point

sys\_mycall

1

2

* + Modified include/linux/syscalls.h:

// System call prototype declaration asmlinkage long sys\_mycall( void );

1

2

## Step 2: Kernel Compilation

Listing 1: Kernel Compilation Commands

# Navigate to kernel source directory cd / usr/ src/ linux -5.4.0

# Copy existing configuration

cp / boot/ config - $( uname -r) . config

# Configure ( use defaults for existing options) make olddefconfig

# Compile kernel with 4 parallel jobs make - j4

# Compile modules make modules

# Install modules

sudo make modules\_install

# Install kernel sudo make install

# Update bootloader ( for GRUB ) sudo update - grub

# Reboot into new kernel sudo reboot

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

## Step 3: Testing

Created test program test mycall.c:

Listing 2: User-space Test Program

# include <stdio .h> # include <linux/ kernel.h> # include <sys/ syscall.h> # include <unistd .h>

// System call number from syscall\_ 64 . tbl # define MY\_CALL 333

int main () {

printf(" Invoking custom system call % d\ n", MY\_CALL);

// Invoke system call using syscall () long ret = syscall( MY\_CALL);

printf(" System call returned % ld\ n", ret);

// Display last kernel messages system (" dmesg | tail -3 ");

return 0;

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

# EXPLANATION

## System Call Implementation

The custom system call sys mycall demonstrates several key concepts:

* + asmlinkage macro ensures proper parameter passing convention
  + printk logs messages to the kernel ring buffer
  + current->pid accesses the calling process’s PID
  + Return value convention (0 for success in this case)
  + System call numbering and registration process

## Kernel Compilation Process

The kernel compilation involves several important steps:

* + make olddefconfig preserves existing configuration
  + Parallel compilation (-j4) significantly reduces build time
  + Separate steps for modules and kernel installation
  + Bootloader update ensures the new kernel is bootable
  + Safe testing in a virtual machine prevents host system instability

## User Space Interface

The test program demonstrates:

* + Using syscall() wrapper function
  + Retrieving system call return value
  + Verification through standard output
  + Accessing kernel logs via dmesg
  + Error handling through return value checking

# SOURCE CODE

## Complete Kernel Modifications

Listing 3: Kernel Modifications

// File : kernel/ sys. c

// Add near other system call implementations asmlinkage long sys\_mycall( void ) {

printk( KERN\_INFO " Custom system call executed from process % d\ n", current - >pid );

return 0;

}

// File : arch / x86 / entry/ syscalls/ syscall\_ 64 . tbl

// Add to x86\_64 system call table

333 common mycall sys\_mycall

// File : include / linux/ syscalls. h

// Add prototype near similar declarations asmlinkage long sys\_mycall( void );

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

## Complete Test Program

Listing 4: Complete Test Program

# include <stdio .h> # include <stdlib .h> # include <linux/ kernel.h> # include <sys/ syscall.h> # include <unistd .h> # include <errno .h>

# define MY\_CALL 333 int main () {

printf(" Testing custom system call % d\ n", MY\_CALL); long ret = syscall( MY\_CALL);

if ( ret < 0) {

perror(" System call failed "); return EXIT\_FAILURE ;

}

printf(" System call successful. Returned : % ld\ n", ret);

// Display kernel log messages printf("\ n Kernel log messages :\ n");

system (" dmesg | grep ’ Custom system call ’ | tail -2 ");

return EXIT\_SUCCESS ;

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

# OUTPUT

## Test Program Execution

Listing 5: Terminal Output

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | $ gcc test\_mycall. c -o test\_mycall | |  |
| 2 | $ ./ test\_mycall |  |
| 3 | Testing custom system call 333 |  |
| 4 | System call successful. Returned : | 0 |
| 5 |  |  |
| 6 | Kernel log messages: |  |
| 7 | [ 1234 .567890 ] Custom system call | executed from process | 4567 |
| 8 | [ 1234 .567891 ] Custom system call | executed from process | 4567 |

## Verification Commands

Listing 6: Verification Steps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1  2  3 | # Verify system call number  $ grep mycall / proc/ kallsyms ffffffffabcdef00 T sys\_mycall | | | |
| 4 |  |  |  |  |
| 5 | # | Check system | call in kernel logs |  |
| 6 | $ | dmesg | grep | " Custom system call" |  |
| 7 | [ | 1234 .567890 ] | Custom system call executed from process | 4567 |
| 8 | [ | 1235 .678901 ] | Custom system call executed from process | 4568 |
| 9 |  |  |  |  |
| 10 | # | Check kernel | version |  |
| 11  12 | $ uname -r  5.4.0 - custom | |  |  |

Figure 1: Complete output showing system call execution, return value, and kernel log verification

# CONCLUSION

This project successfully demonstrated the complete process of adding a custom system call to the Linux kernel, providing valuable insights into:

* + The system call mechanism and its role in OS architecture
  + Kernel source code organization and modification procedures
  + The kernel compilation and installation process
  + User space to kernel space interaction
  + System call verification and debugging techniques
  + Performance considerations in kernel development

The implementation was verified through multiple methods:

* + Successful compilation and booting of modified kernel
  + Correct system call invocation from user space
  + Proper message logging in kernel ring buffer
  + Expected return value validation
  + System call table entry verification Future enhancements could include:
  + Adding parameters to the system call
  + Implementing more complex functionality
  + Performance benchmarking
  + Security analysis of custom system calls
  + Integration with kernel modules

This project serves as a foundation for more advanced operating system development and kernel programming tasks, providing practical experience with low-level system pro- gramming concepts.

[**https://github.com/akshitharajn/os-project/tree/main**](https://github.com/akshitharajn/os-project/tree/main)