Step By Step Custom System Call Creation in the Linux

Contents

Introduction

Creating a custom system call in Ubuntu involves several steps. This process requires modifying the Linux kernel, so it should be done with caution. Below is a step-by-step guide to help you through the process:

This workbook is aimed at help building a custom system call in Linux, step by step.

Process overview step by step

The following figure explains the flow of steps to be performed to create a custom system call, registering with the system, and testing it.

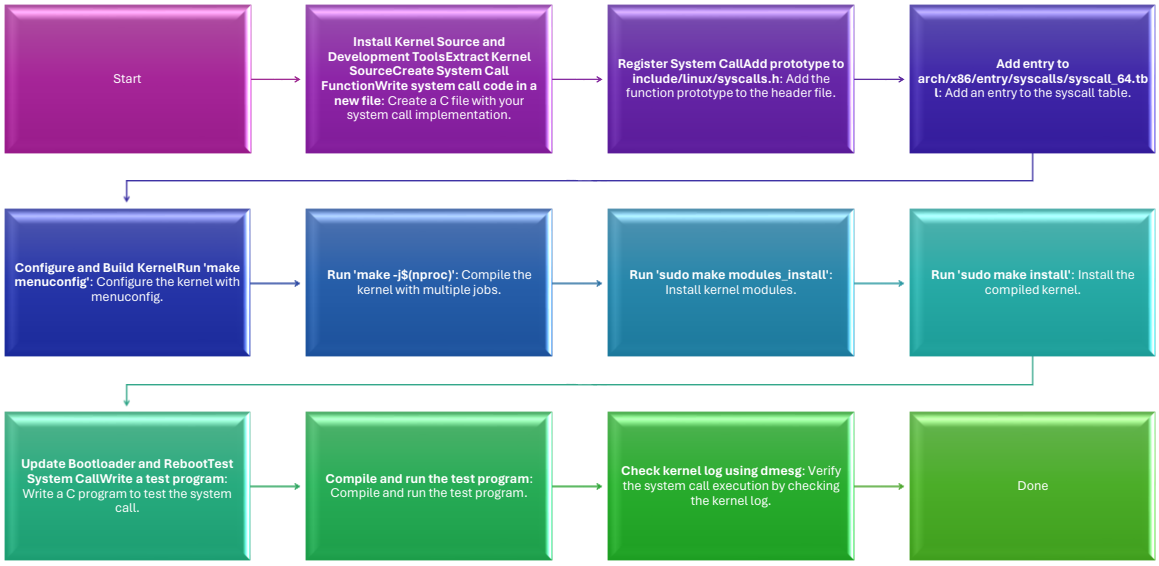

Figure 1 - Process overview of creating a custom system call

Step By Step on Ubuntu commands

- Install Kernel Source and Development Tools: Install necessary tools and kernel source code using apt.
- Extract Kernel Source: Navigate to /usr/src/ and extract the kernel source.
- Create System Call Function: Write the code for your custom system call in a new file.
- Write system call code in a new file: Create a C file with your system call implementation.
- Register System Call: Add the prototype and entry for your system call in the appropriate files.
- Add prototype to include/linux/syscalls.h: Add the function prototype to the header file.

- Add entry to arch/x86/entry/syscalls/syscall_64.tbl: Add an entry to the syscall table.
- Configure and Build Kernel: Configure the kernel, compile it, and install the new build.
- Run 'make menuconfig': Configure the kernel with menuconfig.
- Run 'make -j$(nproc)': Compile the kernel with multiple jobs.
- Run 'sudo make modules_install': Install kernel modules.
- Run 'sudo make install': Install the compiled kernel.
- Update Bootloader and Reboot: Update the bootloader configuration and reboot the system.
- Test System Call: Write and compile a test program to invoke the system call and check the result.
- Write a test program: Write a C program to test the system call.
- Compile and run the test program: Compile and run the test program.
- Check kernel log using dmesg: Verify the system call execution by checking the kernel log.

Steps to Create a Custom System Call

1. Install Kernel Source and Development Tools

First, ensure that you have the necessary tools and kernel source installed:

```
bash
Copy code
sudo apt update
sudo apt install build-essential libncurses-dev bison flex libssl-dev
libelf-dev
sudo apt install linux-source
```

2. Extract Kernel Source

The kernel source is usually located in `/usr/src/`. Extract it if it's in a compressed format:

```
cd /usr/src/
sudo tar -xvf linux-source-<version>.tar.xz
cd linux-source-<version>
```

3. Create Your System Call Function

Navigate to the appropriate directory and create your system call. You will typically add your system call to an existing file for simplicity (e.g., `sys.c` in the `kernel/` directory).

```
// kernel/my_syscall.c

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

SYSCALL_DEFINE0(my_syscall)
```

```
{
    printk(KERN_ALERT "My custom syscall has been invoked!\n");
    return 0;
}
```

4. Register Your System Call

You need to register your new system call in the appropriate header and syscall table.

- **Header File**: Add a prototype for your system call in an appropriate header file (e.g., `include/linux/syscalls.h`).

```
// include/linux/syscalls.h

asmlinkage long sys_my_syscall(void);
```

- **System Call Table**: Add an entry for your system call in the system call table. The location of this table can vary between kernel versions. For example, in x86_64, it's usually `arch/x86/entry/syscalls/syscall_64.tbl`.

```
plaintext
Copy code
// arch/x86/entry/syscalls/syscall_64.tbl

### Add your custom syscall entry ###
334    common  my_syscall    sys_my_syscall
```

5. Configure and Build the Kernel

Now you need to configure and compile the kernel with your new system call.

```
bash
Copy code
cd /usr/src/linux-source-<version>
make menuconfig
```

You can use the default configuration for your system and save it. After that, compile the kernel:

```
bash
Copy code
make -j$(nproc)
sudo make modules_install
sudo make install
```

6. Update Bootloader and Reboot

After building and installing the new kernel, update the bootloader and reboot your system.

```
bash
Copy code
sudo update-grub
sudo reboot
```

7. Test Your New System Call

Once the system is back up, you can test your new system call. Create a small test program:

```
// test_syscall.c

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

#define __NR_my_syscall 334

int main() {
    long res = syscall(__NR_my_syscall);
    if (res == 0) {
        printf("System call executed successfully.\n");
    } else {
        perror("System call failed");
    }
    return 0;
}
```

Compile and run the test program:

```
bash
Copy code
gcc -o test_syscall test_syscall.c
./test_syscall
```

If everything is set up correctly, you should see the message from the system call in the kernel log:

```
bash
Copy code
dmesg | tail
```

You should see something like:

```
[ 1234.567890] My custom syscall has been invoked!
```