CS60038: Assignment 2

Creating a Custom System Call

Submission Deadline: September 14, 2024, EOD

Submission Instructions

1. Submit your assignment through the CSE Moodle platform: `https://moodlecse.iitkgp.ac.in/`. The course joining key is: **AOSD@Stud24**.
2. Only one member from each group should submit the assignment as a single zip file.
3. Ensure that the names and roll numbers of all group members are included in the submission.
4. Provide comprehensive documentation of your implementation. Include a README file explaining how to compile and run your code, along with the test cases you have used.

Objective

The objective of this assignment is to gain hands-on experience in defining and creating a custom system call to facilitate interaction between user-space programs and the Linux kernel.

Please download the 64-bit Ubuntu 22.04 LTS Desktop image and use it in a Virtual Machine (VM). Note that all assignments will be evaluated on this platform and kernel version only.

For tasks involving kernel configuration and building, use kernel version 5.10.223.

Task Description

In this assignment, you are required to:

1. Download the Linux kernel source code (Reuse the same as Assignment-1).
2. Define and declare the custom system call (**gettaskinfo**) in the appropriate files.
3. Compile the kernel.
4. Implement a C library wrapper (**lib_gettaskinfo**) around this system call.
5. Write a user-space C program to test the system call.

System Call: gettaskinfo

This system call should retrieve the following information about a specific process from its task structure:

- *State*: The current state of the process.
- *Start_time*: The time when the process started.
- *Normal_priority*: The normal priority of the process, used for scheduling.

Arguments

The system call should accept two arguments:

1. **PID**: The PID of the process whose information is required.
2. **Buffer**: A char buffer that will store the extracted fields from the task_struct. You have to store all the fields that are extracted in this buffer, and then parse the buffer in the wrapper function. The buffer should contain the *state* first, followed by the *start_time* of the process and then the *normal_priority*.

Return Values

The system call should return the following error codes:

- -ESRCH: If the PID is invalid.
- -EFAULT: If the user-space buffer location is invalid.

To test the system call without the wrapper function, you can use the **syscall()** function provided by glibc. Reference- syscall manpage.

C Library Wrapper: lib_gettaskinfo

Implement a wrapper around the custom system call to make it usable by user-space C programs. The wrapper function does not take any arguments itself, so any fields that need to be passed to the syscall need to be created/populated in the wrapper function.

Return Values

The wrapper function should return a pointer to a custom C struct with all the fields populated from the syscall. In case of any error, it should set the appropriate *errno* and return a **NULL** pointer instead.

Submissions

Students are required to submit the following:

1. Files modified in the kernel source code to implement the system call, along with any modified Makefiles.
2. The C library wrapper file for the system call.
3. The user-space test program.
4. A README file detailing all modified files and instructions on how to use the library. It should also contain any design decisions you made, with justification.