2024-Spring

Operating System Project1 Report

Department: Computer Science & Engineering

2022320149 호수빈

Submission date: 2023/04/10

Number of Free days: 0

1. Development Environment

Window 11 / Virtual Box 7.0.10 / Ubuntu 18.04.2 / Linux kernel 4.20.11

2. Explanation of system calls on Linux

The CPU operates in two execution modes for system protection: kernel mode and user mode. Kernel mode grants full privileges, allowing the operating system to execute tasks at the system level. On the other hand, user mode offers low-level privileges compared to kernel mode, restricting certain instructions. Consequently, a mode switch is required when executing specific instructions within a user application.

A system call serves as an interface between user mode and kernel mode. It is utilized when accessing protected services provided by the kernel. Instead of directly invoking system calls, APIs are employed to make these calls. For instance, when writing code using printf(), it invokes the write() call in the background.

During a system call sequence, the user application initiates the system call. Subsequently, the system call interface executes a mode switch from user mode to kernel mode. The operating system identifies the system call through an interrupt, accessing the relevant entry in the system call table and providing the system call handler with the necessary parameter addresses. The system call handler then executes the instruction's implementation. Finally, another mode switch occurs, transitioning from kernel mode back to user mode.

3. Implementation

1) syscall_64.tbl

   It's a system call table file that maps the numbers of the system calls to the names of the system-level functions used to invoke them. I added definitions for the system calls that will be used in sslab_my_queue.c.

2) syscalls.h

   It's a file that defines the prototypes of system call functions and declares the functions using asmlinkage. I added definitions for the system call functions that will be used in sslab_my_queue.c. For example, "asmlinkage int sys_os2024_enqueue(int);" means defining a system call function, sys_os2024_enqueue, which takes an integer parameter and returns an integer value."

3) Makefile

   It's a file that designates the objects to be included during the kernel compile process. The variable `obj-y` is used in the Makefile to compile and build all source files in the

directory it is assigned to. By adding sslab_my_queue.o to the obj-y part, an object file is created, allowing sslab_my_queue.c to be compiled within the kernel.

4) sslab_my_queue.c

The code defines two system calls, ‘os2024_enqueue’ and ‘os2024_dequeue’ which implement enqueue and dequeue operations on a queue respectively.

4-1) os2024_enqueue

I have implemented enqueue operations with a queue size set to 100. I defined the system call `os2024_enqueue`, which accepts an integer parameter `a` and returns an integer value. Within this definition, I've incorporated error handling using conditional statements to check if the queue is full or if the integer already exists in the queue. Upon verifying there are no duplications, the queue's rear is updated with the integer `a`. Finally, I utilized `printk()` to ensure that enqueued values are displayed within the kernel.

4-2) os2024_dequeue

I've set the queue size to 100 and implemented dequeue operations. I defined the system call `os2024_dequeue`, which returns an integer `temp`. To handle error cases where the queue is empty, I employed an if statement. Following the check for queue emptiness, I stored a current queue’s rear value in the temp, and shifted the queue's elements to the left using a for loop iteration. Finally, I ensured that dequeued values are displayed within the kernel by utilizing `printk()`.

5) call_my_queue.c

Invoking a system call using a #define statement.

For instance, #define my_queue_enqueue 335, indicates that 'my_queue_enqueue' serves as a macro name to be substituted with 335.

Within the main function, push values into the queue over 3 for iterations. Generate integer values to insert using the rand() function. Call the syscall to invoke the enqueue system call function and insert the value. Handle errors if the value already exists in the queue. And print the enqueued values afterward.

Pop values from the queue over 3 for iterations. Call the syscall to invoke the dequeue system call function and fetch the value. Handle errors if the queue is empty and returns -1. And print the dequeued values afterward.

4. Snapshot of project execution

1) user application

```
hosubin02@hosubin02-VirtualBox:~/project1$ ./call_my_queue
Enqueued: 30
Enqueued: 64
Enqueued: 93
Dequeued: 30
Dequeued: 64
Dequeued: 93
```

2) dmesg

```
[   53.105023] [System call] os2024_enqueue()
[   53.105025] Queue Front --------------------
[   53.105026] 30
[   53.105027] Queue Rear --------------------
[   53.105184] [System call] os2024_enqueue()
[   53.105185] Queue Front --------------------
[   53.105186] 30
[   53.105187] 64
[   53.105188] Queue Rear --------------------
[   53.105193] [System call] os2024_enqueue()
[   53.105193] Queue Front --------------------
[   53.105194] 30
[   53.105194] 64
[   53.105195] 93
[   53.105196] Queue Rear --------------------
[   53.105198] [System call] os2024_dequeue()
[   53.105198] Queue Front --------------------
[   53.105199] 64
[   53.105199] 93
[   53.105200] Queue Rear --------------------
[   53.105202] [System call] os2024_dequeue()
[   53.105202] Queue Front --------------------
[   53.105203] 93
[   53.105203] Queue Rear --------------------
[   53.105205] [System call] os2024_dequeue()
[   53.105206] Queue Front --------------------
[   53.105206] Queue Rear --------------------
```

5. Difficulties and efforts

When implementing the dequeue operation, setting the condition as if == front when the queue is empty. And I realized that using rear == front as the condition led to the incorrect determination of the queue being empty even when elements were present. Subsequently, modifying the if condition to rear == −1 resolved the issue, ensuring proper functionality of the code.