FINAL PROJECT REPORT

Project 2

Group Members

Nicolas Cardenas

John Gregulak

James Owens

Aasin Jamil

Operating Systems

Fall 2015

Contents

Project Problem Statement

This project introduces you to the nuts and bolts of system calls, kernel programming, concurrency, and synchronization in the kernel. It is divided into three parts.

The first of the 3 parts was to utilize system-call tracing. We were to write a program that uses exactly 10 system calls. Next, the command line tool, strace was to be used to intercept and record the system calls used by a process.

Part 2 was to create a kernel module called my_xtime that can be used to store the current kernel time by calling current_kerkel_time() and also record the elapsed. The purpose of this part was to introduce the basics of creating a kernel module, inserting the kernel module and also removing the kernel module. The above listed steps would prove to be useful when working on part 3.

The Final task, Part 3, was to implement a scheduling algorithm for a hotel elevator. The elevator was to track the number of passengers and the total weight of the passengers. There were four types of people to account for when creating this elevator.

- An adult counts as 1 passenger and 1 weight unit
- A child counts as 1 passenger and 0.5 weight unit
- A bellhop counts as 2 passengers and 2 weight units
- Room service counts as 1 passenger and 2 weight units

Passengers will randomly appear on a floor of their choosing and they always know where they wish to go. Most of the time, when a passenger is on a floor other than the first, they will choose to go to the first floor. A passenger must always board the elevator if it can accept them in FIFO order, unless it is moving in the opposite direction. Once they board the elevator, they may only get off when the elevator arrives at the destination. Passengers wait indefinitely.

Part 3 was to be divided into 4 steps. Step 1 was to create the kernel module with an elevator. Step 2 was to create a /proc entry called /proc/elevator that would print data about the elevator. Step 3 was to add system calls and finally step 4 was to test the elevator.

Once the project was submitted, the correctness was to be demonstrated for the teacher to grade the project upon.

Assumptions made

The first assumption for this project was that only version 4.2 of the Linux Kernel will be used. Also, the assumption that our code will be demonstrated on the computer that it was written on was made.

Problem Solving Steps

The project was solved in 3 major parts. First, the individual parts were completed in numerical order. System-call tracing was completed first, then the kernel module was completed next. The most amount of time was spent on Part 3, the elevator scheduler. A lot had to be learned about kernel modules and kernel programming for this part of the project. First, the knowledge from completing part 2 was used in creating the kernel module for the elevator. Part 3 was worked on based upon the specification provided and also the rough outline provided by the TA. This rough outline provided a good general idea of how far along the program should be at which dates which really helped to keep us on track.

For the submitted project, the elevator scheduling algorithm was a FIFO mixed with SCAN implementation. The elevator always scans floors 1-10 up, then 10-1 down. On each floor, the elevator unloads all passengers that desire to go to that floor, then loads all passengers moving in the same SCAN direction in FIFO ordering (based on when they requested to use the elevator through the issue_request syscall).

Why Solution uses system calls/libraries

For this project, a few system calls were used. These system calls were

1. Int start_elevator(void)
2. Int issue_request(void)
3. Int stop_elevator(void)
4. Kthread
5. List
6. Mutex

1. Start elevator
    a. This system call would start the elevator. Initially, the state of the elevator would be stopped. Once start elevator was called, the elevator would start on floor 1 and change its status to idle.
2. Issue Request
    a. This system call took care of queueing the passengers. There was a check implemented so that if the passenger's current floor was their destination floor that they did not actually board the elevator but the passengers serviced by that floor was still increased.
3. Stop Elevator
    a. Checks for a flag condition should_stop. This will stop the elevator and return it to floor 1 without servicing any other passengers. Once it returns to floor 1, the status of the elevator will be changed to stopped.
4. Kthread
    a. A thread was used to run the elevator.
5. List

    a. Lists were used to queue the passengers at the floors where they appeared. We used 10 queues, 1 for each floor.

6. Mutex
    a. Mutex is used to eliminate concurrent access to the lists that were implemented. Concurrent access could cause data corruptions in the elevator.

Problems Encountered

The major problems encountered were in part 3 of this project. The first 2 parts were fairly straight forward and simple. For part 3, the elevator algorithm and scheduling were the major problems encountered. Implementing the queues for the floors was another problem that required a lot of thinking to get it implemented successfully.

For the scheduler, we spent a lot of time drawing state diagrams so that we could thoroughly understand what each state should do currently and also what its next move should be.

Known Bugs (same as README)

In some instances when the stop request is issued, the elevator performs a stress test of the cables by going to the top floor then down to the bottom floor where it then stops. During this time it is not accepting passengers.

Division of Labor

Project wasn’t divided into parts. It was worked on as a group. See log entries.

Slack Days Used

For Project 2, no slack days will be used.

Cumulative Log Entries

Group Member Name	Date of Work	Changed Made (Brief Description)
James, John, Nick	10/5	Finished system call tracing. Began working on Part 2.
Aasin, James, John, Nick	10/21	Finished part 2, my_xtime
Nick, John	10/23	Created /proc/elevator entries
James, John	10/26	Implemented global states and variables across the .c files. Worked on printing data and initializing the elevator
James, John, Nick	10/27	Implemented several lists. More working on the queue structure for the elevator
Aasin, James, John, Nick	10/28	More working on elevator and implementing FIFO scheduler
Aasin, James, John, Nick	10/29	Optimization of elevator, fixing little problems in the code
Aasin, James, John, Nick	10/30	Testing of all 3 parts of the project 2. Final changes to elevator

Question Responses

For Project 2, there are no questions to be answered.

Additional Features

At this time, there are no additional features.