UNIVERSITÉ Concordia UNIVERSITY

GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

COEN 346 OPERATING SYSTEMS – WINTER 2023

LABORATORY # 1: SYSTEM CALLS AND JAVA PROGRAMMING REFRESHER

Introduction

"A system call is a public function that the kernel provides to the application and executes on the application's behalf. Common examples include reading and writing to a file, allocating memory, or terminating the program. System calls are generally implemented using an interrupt handler, and pass arguments using specific registers. This is called a calling convention.

At this point, you may be asking yourself, why not just issue a function call directly into the kernel? The answer is primarily so that the kernel can protect itself from a buggy or malicious program. For instance, the system call function may validate that the input arguments make sense, and then call an internal function that does the work. If an application were able to call the internal function directly, the application could circumvent these input parameter checks, and potentially damage the kernel."[1]

Lab description

In this laboratory you will explore and compare the system calls used by two programs, one written in C and one written in JAVA. You will familiarize yourself with the linux **strace** command for monitoring system calls (or a similar tool). Finally, you will refresh your JAVA programming skills by writing a simple process id allocation API and a client for it.

Preparation

1. I recommend that you use a Linux shell for this laboratory. You can access linux from the laboratory computers or install a linux shell in your computer. See how in windows or mac. In windows you will need to turn WSL on, read how on the description of the application. Alternatively, you can use virtual box and install a linux virtual machine with a GUI.
2. [OPTIONAL] To monitor system calls in windows you can use process monitor and in mac you can use Instruments.
3. If you haven't already, install IntelliJ in your computer for the programming assignment part.
4. Create a directory for storing the files of this laboratory in your computer. Place this guide in the directory. This will help you stay organized.
5. Download the source code files from the moodle assignment for this laboratory and place them in the directory for this laboratory in your computer.

[1] Extract from Donald Porter's Laboratory available at https://www.cs.unc.edu/~porter/courses/cse306/s13/lab1.html

In Lab activity

This part of the laboratory should be done **individually**, although you can discuss with your peers and ask them or the TA for help.
Note: the instructions for this part assume a ubuntu shell. You might need to adjust if you are using a different tool.

1. Run the shell and change the current directory to the directory you created for this laboratory using the **cd** command. If you are running one of the shells in mac or windows you might need to give it permission to access the directory files. (see how to do it in multipass and in WSL ubuntu console)
2. Compile the c program (copyfile.c) using gcc

```
gcc copyfile.c -o copyfile
```

3. Run the program and make sure the file is copied correctly

```
./copyfile samplefile.txt copyofsamplefile.txt
```

4. Run the strace command to inspect system calls made by the program

```
strace -c ./copyfile samplefile.txt copyofsamplefile.txt
```

   The -c option of the command provides a summary of all the system calls that the process is using.

5. What are the system calls to open, read and close a file?
6. Compile the java program using javac

```
Javac CopyFile.java
```

7. Run the program and make sure the file is copied correctly

```
Java CopyFile samplefile.txt copyofsamplefile2.txt
```

8. Run the strace command to inspect the system calls made by the program

```
strace -c java CopyFile samplefile.txt copyofsamplefile2.txt
```

   When you have finished this part of the laboratory during the laboratory session, show your TA the results of the **strace** program (in person) for a participation point.

9. Compare the system calls made by the C and JAVA programs. Why are the system calls so different if the two programs have the same functionality? What are some system calls made by the JAVA program that are not made by the C program and why might they be called?
   You may search online for these answers but provide an explanation in your own words.

   Read more about strace

Programming assignment description

This activity is intended to refresh your java programming skills. If you need a refresher on OOP concepts, use this guide.

UNIVERSITÉ Concordia UNIVERSITY

GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

COEN 346 OPERATING SYSTEMS – WINTER 2023

This assignment should be done in groups of one, two or three students. Your group members may be in a different section, but you should all be present for the demonstration. You may not change group during the term. Please register your group by filling this form https://forms.office.com/r/Aa2P6ZWZZE

An operating system's pid manager is responsible for managing process identifiers. When a process is first created, it is assigned a unique pid by the pid manager. The pid is returned to the pid manager when the process completes execution, and the manager may later reassign this pid. What is most important here is to recognize that process identifiers must be unique; no two active processes can have the same pid.
When a process is created, the operating systems asks the PIDManager for a pid for that process.

Programming assignment tasks

1. Create a new project in IntelliJ.
2. In the src folder create a new package called ca.concordia.processmanagement
3. In the new package, create the class PIDManager that will be responsible for allocating process id’s. The class should declare two constants to identify the range of possible pid values:

    ```
    static final int MIN_PID = 300
    static final int MAX_PID = 500
    ```

    The class should implement the following methods:

    `void allocateMap()throws Exception`—Creates and initializes a data structure for representing pids; throws Exception if unsuccessful
    `int allocatePid() throws Exception`–Allocates and returns a pid; throws Exception if unable to allocate a pid (all pids are in use)
    `void releasePid(int pid)`–Releases a pid

    You may use any data structure of your choice to represent the availability of process identifiers. One strategy is to adopt what Linux has done and use an array of bits in which a value of 0 at position indicates that a process id of value is available and a value of 1 indicates that the process id is currently in use. (check the BitSet class in JAVA)
4. Create a class to represent a Process Control Block. For now, the control block should only contain the PID and the process status.
5. Create a JAVA interface to allow operations on processes. (Right click, new Java Class and then select interface in IntelliJ)
    The interface should declare the following methods of the API:

    ```
    int createProcess() throws Exception
    ```

    Creates a process (by creating its process control block) and adds it to the ready queue. Returns the PID of the process created. Throws exception if unable to create the process (for example if no PID are available)

```
void terminateProcess(int pid) throws Exception
```

Terminates the process with the specified pid. Throws exception if the process doesn't exist.

6. Create the class that implements the interface from step 5. When a process is created, it should ask for a PID to the PIDManager, creates a process control block and add it to a queue (the ready queue). When a process is terminated, it should release the PID and remove the process control block from the queue.
7. In the main class (outside of the processcontrol package), create three scenarios to test your implementation. For example, create a process and print its PID. Terminate that process. Create as many processes as ID's available and then create one one more process, you should get an error. Create as many processes as ID's available and then terminate one process. Create one new process, there should be no error. Create one more process. There should be an error. Terminate all processes.
8. You will demonstrate your assignment to the TA after the due date.

   [5points] BONUS: Create your own exceptions to handle errors.

Deliverable

You should submit a .zip folder with the following files

1. A two-page pdf file including your answers to ALL questions in the laboratory.
2. A pdf file including a two-page report of the programming assignment with the following sections:
   a. Group name and Names of group members
   b. High level description of the code (description of the methods/functions/threads/data structures and the flow of the program). You can use UML class and sequence diagrams or any other tool or natural language description.
   c. A detailed conclusion, discussing you experience. Please include the answer to What problems did you face?
   d. A table detailing the contributions to this assignment of every team member. Note that every member of the team must contribute to the design and coding aspects of the assignment.
3. A .zip file of the project code. It should contain only source code not compiled files)