Homework 1: Syscall Basics

Spring 2026 CS 3502: Operating Systems

Environment Setup

This assignment relies on Linux-specific system calls and will not work directly on Windows or macOS. You must set up a Linux environment.

Windows users should avoid VirtualBox. Install WSL2 (Windows Subsystem for Linux) by running **wsl --install** in PowerShell as Administrator, then reboot and open "Ubuntu". Mac users with Apple Silicon (M1/M2/M3) must use UTM or Docker, as VirtualBox is unstable. Intel Mac users can use VirtualBox or run natively.

If you cannot install software, use GitHub Codespaces or Replit. Select a C/C++ or Bash template to access a Linux terminal in your browser.

Once that's set up, run the following command to ensure you have all the packages you need:

sudo apt update && sudo apt install build-essential strace ltrace zip unzip

Getting Started

Download **hw1.zip** from D2L and unzip it in your Linux environment. You will see a program called **hw1_grader**. Run it at any time using **./hw1_grader** to check your progress. If it returns "Permission denied," you will fix that in Task 2.

Task 1: Hidden Archive

The **data** folder contains a maze of nested subfolders. Hidden inside is a file named **target.txt**. Find this file and copy it to the main **hw1** directory.

Manual exploration will not work because there are hundreds of directories, including hidden ones (starting with a dot) and decoy files. Use the **find** command (reference **man find**). The grader checks if **target.txt** exists in the **hw1** folder.

Task 2: Permission Denied

The script **run_me.sh** fails with "Permission denied" when you try to execute it. Inspect the file's current permissions using **ls -l run_me.sh**, then use the **chmod** command to fix the issue so the script can run.

When successful, the script creates a file called **key_file.txt**. The grader checks for this file.

Task 3: Hello World

Create a file named **syscall_hello.c**. Write a C program that prints "Hello, OS!" (followed by a newline) to the terminal using the **write()** system call. You cannot use **<stdio.h>** or **printf**. You must use **<unistd.h>**.

Use the following template:

#include <unistd.h>

int main() {

// write(FileDescriptor, Buffer, Length);

// FileDescriptor 1 = Standard Output (the screen)

write(1, "Your Text Here\n", ???);

return 0;

}

The third argument to **write()** is the number of bytes. You must count your string length accurately, including the newline character **\n**. Compile using **gcc syscall_hello.c -o syscall_hello**. The grader checks for the correct system call usage and output.

Task 4: Missing File

Run the binary **program_A**. It will crash due to missing configuration. Use **strace ./program_A** to discover what file the program is trying to open. Look for a line containing **openat** returning **-1 ENOENT** (No such file or directory).

Once you identify the missing filename, create it using **touch** , then run **program_A** again to verify. The grader checks that the required file exists.

Task 5: Runaway Process

Run **program_B**. It will spawn a background process and exit, but the background process will keep running. You must identify its PID and terminate it.

Use **ps aux | grep program_B** to find the running process. Create a file named **killed_pid.txt** containing only the PID number. Then, terminate the process using the **kill** command. Verify it is gone using **ps**. The grader checks that **killed_pid.txt** exists and contains the correct number.

Task 6: Noisy Program

Run **program_C**. It outputs a flood of text containing two secret keys. One is in the standard output (stdout, file descriptor 1) and the other is in the standard error (stderr, file descriptor 2).

You must capture both streams to separate files: **stdout.log** and **stderr.log**. Use redirection to separate the streams (e.g., **> stdout.log 2> stderr.log**). The grader checks that both files exist and contain the keys.

Task 7: Secret Environment

Run **program_D**. It prints "ACCESS DENIED" because it checks for a specific environment variable. **strace** will not help here.

Use **ltrace ./program_D** to view library calls. If **ltrace** is not installed, install it via **sudo apt install ltrace**. Alternatively, run **strings program_D | grep -i env** to look for strings inside the binary.

Identify the variable name, set it using **export VARIABLE_NAME=value**, and verify "ACCESS GRANTED" appears. Save the successful output by running **./program_D > env_out.txt**.

Task 8: The Gauntlet

Run **program_E**. It fails with a generic error. You must fix multiple issues to make it run successfully. No step-by-step instructions are provided. Use the tools learned in previous tasks (**strace**, **ltrace**, **chmod**, **touch**, **export**).

When you succeed, save the proof by running **./program_E > gauntlet.txt**. The grader checks this file for the success message.

Note: if you see an error similar to **access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)**, you can safely ignore it. This is not causing the program to fail.

Submission

Run **./hw1_grader**. If all tasks are complete, it will output a Submission Hash. Copy this string and submit it to D2L in a file called **submission.txt**.

Summary of Required Files

- **target.txt** (Task 1)
- **key_file.txt** (Task 2)
- **syscall_hello.c** (Task 3)
- The missing config file (Task 4)
- **killed_pid.txt** (Task 5)
- **stdout.log**, **stderr.log** (Task 6)
- **env_out.txt** (Task 7)
- **gauntlet.txt** (Task 8)

Academic Integrity

You may discuss concepts with classmates, but all commands must be typed by you. Copying another student's hash string is detectable and will result in a zero.