Bahir DarUnversty

Bahir Dar Institute Of Technology FaculityOf Computing

Department: Software Engineering

Project Name: system call implementation

Name : Tiruneh Getachew **Submitted To: Lec Wendimu B**

ID : BDU1602613 **Sobmission Date: 16/08/17**

❖ Implementing a Simple System Call

This project includes implementing a basic system call to demonstrate how applications interact with the operating system. System calls are special functions that user-level programs use to request services from the operating system's core (kernel).

This system call is part of the POSIX standard and allows a process (with appropriate privileges) to change the time for a specific clock — most commonly, CLOCK_REALTIME, which affects the entire system clock.

Step 1: Prepare Termux in VMware

1. **Install Termux** from **F-Droid** (avoid Play Store, as it's outdated).
2. Update Termux packages:
   bash
   ```
   pkg update &&  pkg upgrade
   ```
3. Install essential tools (`gcc`, `clang`, `make`):
   bash
   ```
   pkg install clang make
   ```

Step 2: Write the Program

1. Create a file (e.g., `set_time.c`) using `nano` or `vim`:

   bash
   ```
   nano set_time.c
   ```
2. Paste the **modified code** (from the previous answer). Example:

bash

```
#include <time.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/syscall.h>
```

```
int main() {
    struct timespects;
ts.tv_sec = 1713960000;  // Example UNIX timestamp
ts.tv_nsec = 0;

    // Direct syscall to clock_settime
    if (syscall(SYS_clock_settime, CLOCK_REALTIME, &ts) == -1) {
perror("Failed to set time");
    } else {
printf("Time set successfully.\n");
    return 0;}
```

save it and.

Step 3: Compile the Program

1. Compile with `gcc` or `clang`:

   bash

   ```
   gcc set_time.c -o set_time
   ```

Step 4: Run the Program

1. **Attempt to run without root** (will fail):

   Bash

   ```
   ./set_time
   ```