

## Implementation of clock\_settime system call on Android-x86

Android's native environment does not provide direct support for running C language code or utilizing essential tools like the GCC (GNU Compiler Collection) library out of the box. This limitation arises because Android focuses primarily on Java or Kotlin-based applications for app development, leaving system-level programming tasks less accessible to regular users. As a result, the default terminal emulator on Android lacks the necessary compilation tools for C programs.

To overcome this limitation, developers and enthusiasts turn to Termux, a powerful terminal emulator application available as an APK for Android devices. Termux bridges the gap by offering a Linux-like environment with access to essential development tools. With Termux installed, users can set up a fully functional C development environment by downloading libraries like GCC, Clang, and other utilities directly within the app. This enables the compilation, testing, and execution of C code seamlessly on Android devices.

Using Termux for system programming provides flexibility and convenience, allowing users to explore low-level programming tasks without the need for external hardware or additional software. Its versatility makes it a vital tool for developers looking to experiment with system calls like `clock_settime()` in a simplified, portable environment.

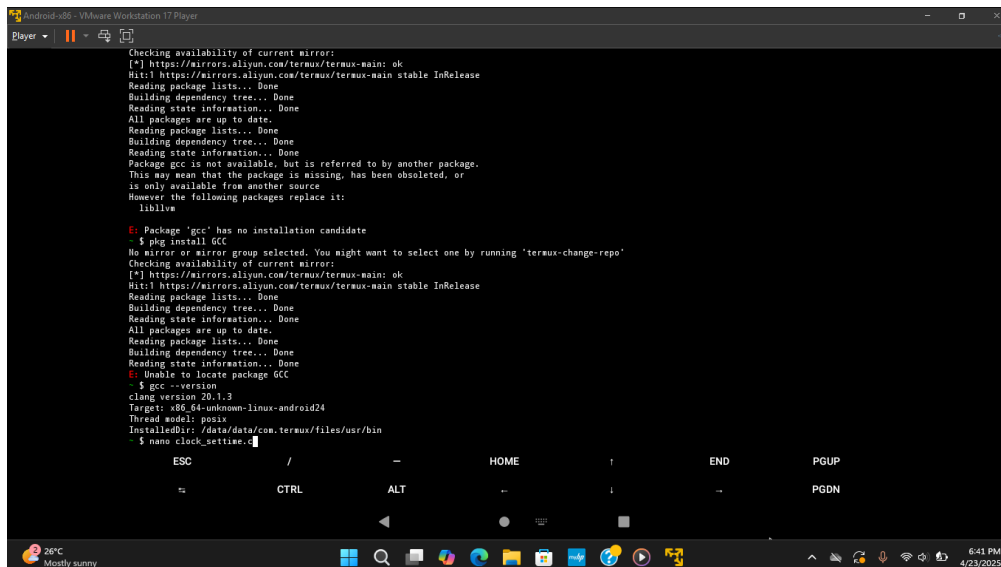
### Steps for Implementation

1. Setting Up Termux for Development:
  - a. Provide a guide on installing Termux and essential development tools like GCC, Clang, and libraries for C programming.
  - b. Include commands to update Termux and install packages like: Pkg update (checks for updates to the list of available packages from Termux's repositories.)
    - Pkg upgrade (refreshes the list of available packages, running; does the actual work of upgrading installed packages to their latest versions.)
    - Pkg install gcc
2. Writing the C Program:
  - Write a clear explanation of how you can create a new `.c` file within Termux using editors like nano or vim.
  - Include the complete C code for the `clock_settime()` system call with inline comments to explain key sections.
  - The primary purpose of the `clock_settime()` system call is to modify the system clock of the operating system to a specific time and date.

## Key Functions of clock\_settime()

1. Updates System Clocks:
  - It allows programs to change the value of a specific clock in the system, such as:
    - ✓ CLOCK\_REALTIME: The real-time clock representing the current calendar time.
    - ✓ CLOCK\_MONOTONIC: A clock that measures time relative to when the system was started, unaffected by system time changes.
2. Precise Time Control:
  - You specify the exact number of seconds (tv\_sec) and nanoseconds (tv\_nsec) since the epoch (January 1, 1970, 00:00:00 UTC).
  - This enables accurate time adjustments for system clocks.
3. System-Level Effects:
  - For example, when the CLOCK\_REALTIME clock is updated, the date and time displayed by commands like date or system logs will change accordingly.
4. Testing and Synchronization:
  - It's commonly used for testing systems that are time-dependent or synchronizing the system time with an external source (e.g., NTP servers).

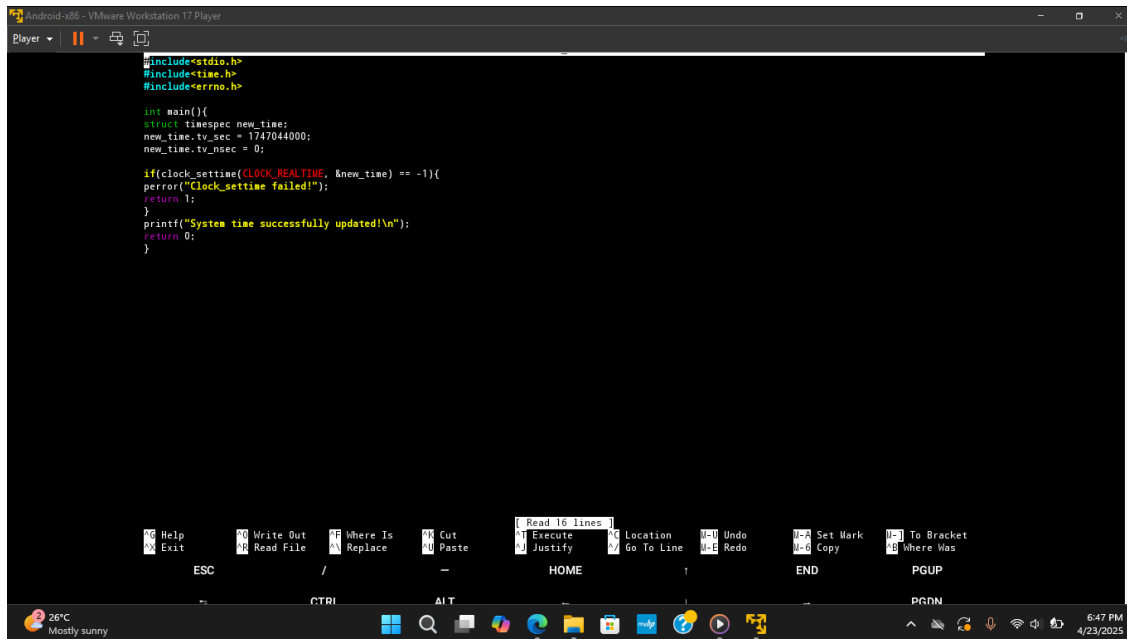
To implement this system call, first we must create a c language code as shown below.



```
Checking availability of current mirrors:
[*] https://mirrors.aliyun.com/teraux/teraux-main: ok
Hit:1 https://mirrors.aliyun.com/teraux/teraux-main stable InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package gcc is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source
However the following packages replace it:
libltdl

E: Package 'gcc' has no installation candidate
$ pkg install gcc
No mirror or mirror group selected. You might want to select one by running 'teraux-change-repo'
Checking availability of current mirrors:
[*] https://mirrors.aliyun.com/teraux/teraux-main: ok
Hit:1 https://mirrors.aliyun.com/teraux/teraux-main stable InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package GCC
$ gcc --version
clang version 20.1.3
Target: x86_64-unknown-linux-android24
Thread model: posix
InstalledDir: /data/data/com.teraux/files/usr/bin
$ nano clock_settime.c
ESC / HOME END PGUP
to CTRL ALT -- -- PGDN
```

This is a page where the updated and upgraded form is. GCC compiler is also installed and here we can create a c file by writing “nano clock\_settime.c”. Here, nano is the text editor and when we press enter, it takes us to the nano editor where we can write our code.

A screenshot of a nano text editor window. The window title is "Android-x86 - VMware Workstation 17 Player". The code inside is as follows:

```
#include<stdio.h>
#include<time.h>
#include<errno.h>

int main(){
    struct timespec new_time;
    new_time.tv_sec = 1747044000;
    new_time.tv_nsec = 0;

    if(clock_settime(CLOCK_REALTIME, &new_time) == -1){
        perror("Clock_settime failed!");
        return 1;
    }
    printf("System time successfully updated!\n");
    return 0;
}
```

The nano editor interface includes a menu bar at the bottom with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Read 10 lines, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, and Where Was. The status bar at the very bottom shows "26°C Mostly sunny" and "6:47 PM 4/23/2025".

This is our code above.

`#include` : This library provides functions and structures for working with date and time, such as `clock_settime()`.

`#include` : This library is used for input and output functions, like `printf()` to display messages.

`struct timespec new_time;`: Declares a variable `new_time` of type `struct timespec`. This structure is used to represent time with two components:

- `tv_sec`: The number of seconds since the Unix epoch (January 1, 1970, 00:00:00 UTC).
- `tv_nsec`: The additional nanoseconds beyond the seconds value.

`new_time.tv_sec = 1747044000;`: Sets the `tv_sec` field to the value 1747044000. This represents a specific point in time (Monday, May 12, 2025, at 10:00:00 UTC). We can adjust this value to set a different time.

`new_time.tv_nsec = 0;`: Sets the `tv_nsec` field to 0, meaning there are no additional nanoseconds.

`clock_settime(CLOCK_REALTIME, &new_time);`

- This system call sets the system's real-time clock (used for calendar time) to the value specified in `new_time`.
- The first argument, `CLOCK_REALTIME`, indicates that you're modifying the real-time clock.
- The second argument, `&new_time`, is a pointer to the struct `timespec` containing the new time.

If `clock_settime()` fails, it returns `-1`. The `perror()` function prints an error message (like "Permission denied"), and the program exits with a return code of `1`.

If `clock_settime()` succeeds, this line is executed. It prints a success message to the console.

After we finish, we click "ctrl + x" to exit. Then it asks us whether we want to save our file or not. We write capital Y which stands for yes save my file, then we press enter. This exits us from the editor.

Now, if we compile and run our code, it will return `-1`, which we do not want. In order to make it work, we have to switch termux into root.

We needed to use root because the `clock_settime()` system call modifies the system clock, which is a protected resource that requires administrative (superuser) privileges to change. On Android, the ability to adjust the system's real-time clock is restricted to prevent unauthorized access or accidental modifications that could affect other applications, logs, or system processes.

To switch to root, we write "su" then it will switch. After that, we locate our file using "cd". Then, we compile it using "gcc clock\_settime.c -o clock\_settime". This will compile our code. To run it, we write "./clock\_time". This gives us our desired output.

