System Calls

Creating a custom system call in XV6 involves modifying the XV6 operating system's source code (**kernel/**).

Here are the general steps to create a custom system call in XV6:

1. Set Up Your XV6 Development Environment:

This was covered in previous labs.

2. Modify the System Call Interface:

- Open `syscall.h` located in the kernel directory.
- Define a unique system call number for your custom system call. Add a `#define` for it, incrementing the number for each additional system call:

  -
    ```
    #define SYS_mycall 22
    ```

3. Add the System Call Prototype in syscall.h:

- In `syscall.c`, declare a prototype for your system call function, like:

  -
    ```
    extern uint64 sys_mycall(void);
    ```

4. Update the System Call Table:

- In `syscall.c, add an entry for your custom system call in the `syscalls` array. This maps the interface for your user-level programs. Add an entry like:

  -
    ```
    static uint64 (*syscalls[])(void) = {
       [SYS_mycall] sys_mycall,
    }
    ```

5. Implement the System Call:

- Write the implementation of your custom system call in the `sysproc.c` file. This is where you define what your system call does when it is called by a user-level program. For example:

  -
    ```
    uint64
    sys_mycall(void) {

       // You implementation here
       return 0;  // Success
    }
    ```

6. Update the usys.pl file in the user directory:

- Add an entry in the usys.pl

    -
      ```
      entry("mycall")
      ```
- So, when you compile this perl file, an entry in `` `usys.S` `` is made.

7. Call the System Call in User Code:

- In the user-level program, you can now call your custom system call as you would with any other system call.

8. Build and Test:

- Rebuild the xv6 operating system using the `` `make` `` command and test your custom system call by creating a user-level program that calls it.

NOTE: Always test your modifications carefully and be prepared to debug any issues that may arise.