# **Yrysbaev Maksatbek**

Professor’s Name: Glory Kurian

Course Name: COMP 2415 Systems Programming

Date: 04.30.2024

# **Assignment 05**

1. State the difference between a library function call and a system call in C programming.
   1. Library function call: If you want to print a message or add numbers, you might use a library function like printf() to show the message or add() to add numbers.
   2. System call: If you want to save a file, your program will use a system call to ask the computer's system to save the file for you.
   3. Answer: The Library functions as regular tools that your program uses all the time, and system calls as special requests for help from the computer's system when your program needs to do something special!
2. Explain the role of the exec() family of system calls in C programming.
   1. When you call an exec() function, it tells the operating system to stop the current program and start a different one. The new program starts running from the beginning, and it replaces everything of the old program except for its process ID and some other process details.
3. How is process termination typically handled using system calls in C programming? Which are the system calls involved?
   1. In C programming, process termination is typically managed using the **exit** and **\_exit** system calls. The **exit** function handles high-level termination, performing cleanup by flushing buffers and running functions registered with **atexit**. The **\_exit** system call, called internally by **exit**, performs the actual process termination at the operating system level, exiting without any cleanup. Other related functions include **abort**, which terminates the program abruptly without cleanup, and **atexit** for registering functions to execute on termination.
4. What is a signal handler? What does it do?
   1. A signal handler is a function defined in a program to manage signals, which are notifications sent by the operating system indicating events like errors or external interrupts. The role of a signal handler is to execute specific actions in response to these signals, such as cleaning up resources, ignoring the signal to continue running, or terminating the process. This allows a program to handle unexpected events gracefully and maintain control over its execution.
5. Discuss the significance of header files in C programming with respect to libraries.
   1. Header files in C programming are crucial for managing and using libraries effectively. They serve several key purposes:
      1. Function Declaration
      2. Type Definition
      3. Macro Definition
      4. Template Code
6. What is the Curses library used for and which are the I/O functions controlled by the Curses library?
   1. The Curses library is used in C programming for creating text-based user interfaces within a terminal.
   2. Key I/O functions controlled by the Curses library include:
      1. Window creation and management
      2. Screen refresh functions
      3. Input handling
      4. Cursor movement
      5. Text output
      6. Attribute control
7. State two points differentiating the X library and the Win 32 library?
   1. Operating system support:
      1. X library: Primarily used in Unix-like systems, including Linux and various Unix flavors.It provides a low-level client interface to the X Window System, which is the standard toolkit for managing graphical user interfaces in these environments.
      2. Win32 API: Designed specifically for Windows operating systems. It provides a comprehensive set of functions for creating and managing windows, drawing graphics, and handling user input, among other system tasks.
   2. Level of abstraction:
      1. X Library: Offers a more low-level approach to GUI programming. It directly interacts with the X Window System for basic windowing and graphics tasks, requiring more code to perform what might be simpler tasks in higher-level toolkits. It is typically used for applications needing fine control over windowing and event handling.
      2. Win32 API: Provides a relatively higher-level approach compared to Xlib, with more direct support for complex GUI elements, such as dialogs, menus, and standard controls. The API includes functions that abstract many of the details needed to manage windows and devices, making it easier to develop fully-featured applications.
8. How do you create and use your own custom libraries in C programming?
   1. Write the Library Code:
      1. Create a header file (mylib.h) for function prototypes and definitions.
      2. Implement these functions in a source file (mylib.c).
   2. Compile the Library:
      1. For a static library, compile with gcc -c mylib.c and create the library with ar rcs libmylib.a mylib.o.
      2. For a dynamic library, compile with gcc -fPIC -c mylib.c and create the shared library with gcc -shared -o libmylib.so mylib.o.
   3. Use the Library in an Application:
      1. Include the library’s header file in your application code.
      2. Compile the application by linking it to the library using gcc -o myapp myapp.c -L. -lmylib and set the library path if using dynamic libraries.