

**CL 2006 OPERATING SYSTEM**



**BS Software Engineering**

**Fall-2024**



**Course Instructors: Syed Daniyal Hussain Shah, Hifza Umar**

**Course Coordinator: Syed Daniyal Hussain Shah**



**Lab Policy and Rules:**

1. **100% attendance is mandatory.** In case of an emergency, you can avail yourself of up to 3 absences. So, don't waste your absences , save them for emergencies. If you are debarred from the course due to low attendance, do not come to me to correct it.

2**. Disturbing the class environment during lab sessions.** such as by talking, using mobile phones, eating, etc. will result in penalties (e.g., deduction of marks).

3. **Lab tasks will not be accepted if you are absent.** If you miss a lab, you cannot submit that lab task.

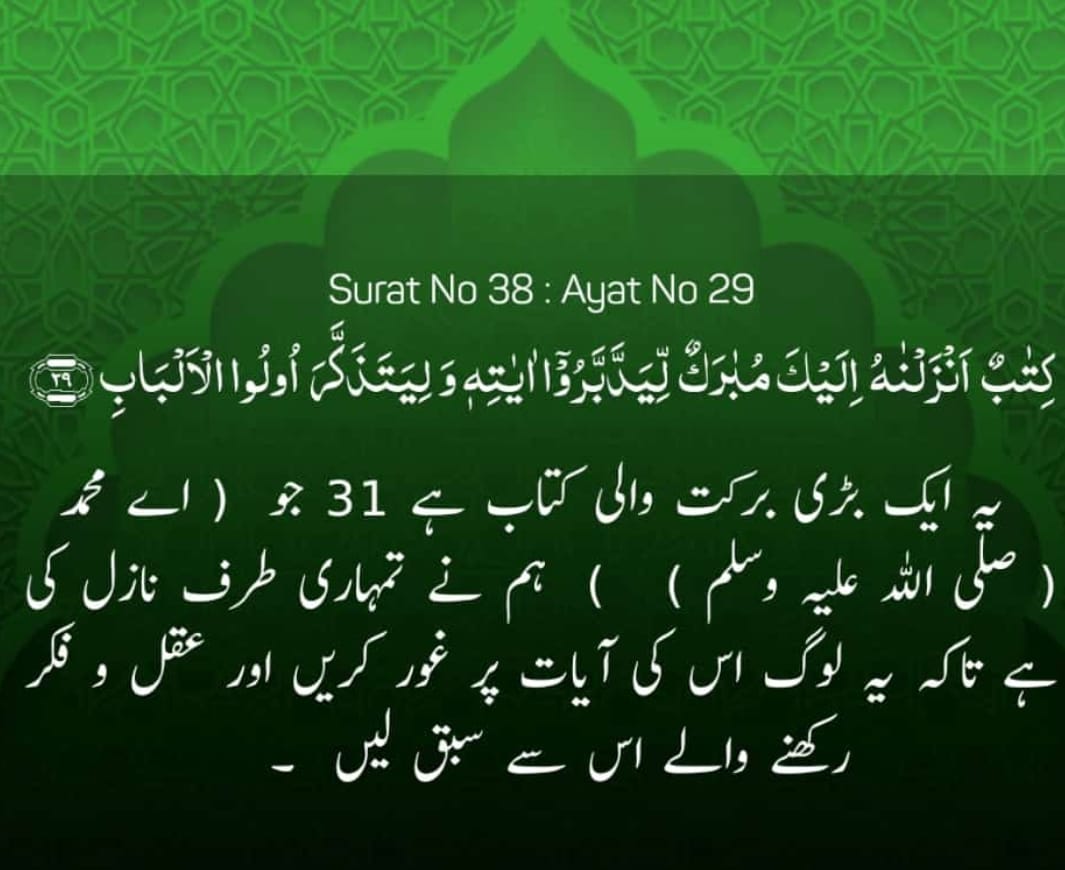
4**. Lab demos for each lab task will be conducted at the end of each session.** If you miss the demo for any reason, no retake will be allowed, and that lab will be marked as 0.

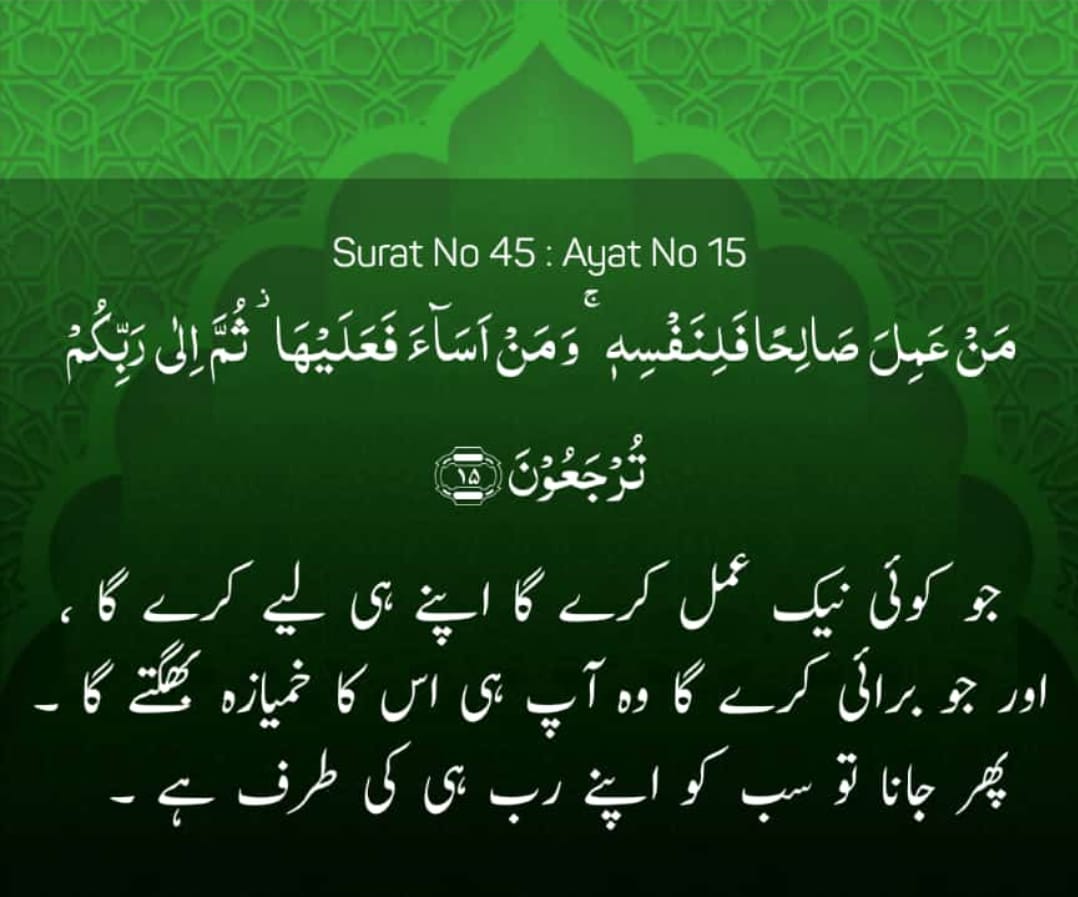
5. **All quizzes will be unannounced**. Be prepared for surprise quizzes. Quizzes will cover content from previous labs as well as theory lectures from the corresponding theory course.

6. **You can take help from the internet for lab tasks,** but simply copying and pasting without understanding will be marked as 0. You should be able to explain the syntax or material used in your tasks.

7**. Do not ask for personal favors.** If you have concerns, such as short attendance, please speak with the relevant authority (e.g., academics).

8**. Students on warning:** Now is the time to study and earn a good grade. Do not ask for extra marks at the end if you are unable to clear your warning.





# 

# **Lab 11: Threads**

# Threads:

A **thread** is a smaller unit of a process that can be executed independently. A process can have multiple threads, all sharing the same memory space but executing different parts of the program simultaneously.

**Imagine a cooking task:** if you have multiple cooks (threads), they can each focus on a specific task (one chops vegetables, another boils water). They all work together towards a common goal (preparing the meal).

In programming, threads allow us to split tasks so they can run in parallel, which can make programs faster and more efficient.

# pthread\_create:

This function is used to create a new thread.

## Syntax:

int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg);

**thread:** The identifier for the new thread.

**attr:** Specifies attributes (can be NULL for default settings).

**start\_routine**: The function the thread will execute.

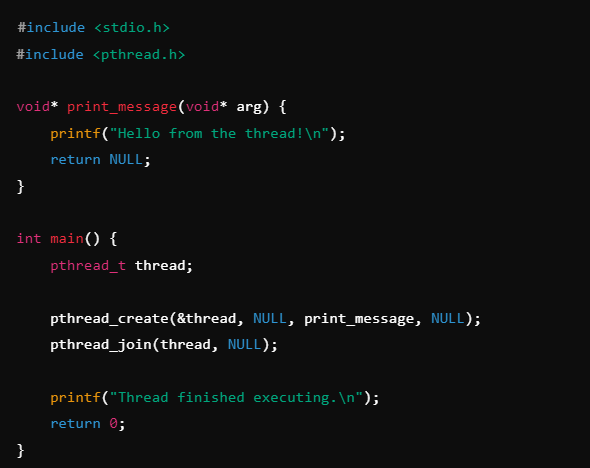
**arg:** A pointer to arguments passed to the function (can be NULL).

## Return Value:

**Returns 0:** If the thread was successfully created.

**Returns a non-zero value:** If an error occurred (e.g., insufficient resources).

## Example:



In this example, pthread\_create creates a new thread that runs the print\_message function.

## Output:



# pthread\_self:

pthread\_self() returns the ID of the calling thread.

## Example:



In this example, the thread prints its own ID using pthread\_self().

## Output:



# pthread\_join:

This function makes the calling thread (usually the main thread) wait until another thread terminates.

## Return Value:

**Returns 0:** If the call completed successfully.

**Returns a non-zero value:** If an error occurred (e.g., invalid thread ID).

## Syntax:

int pthread\_join(pthread\_t thread, void \*\*retval);

**thread:** The thread to wait for.

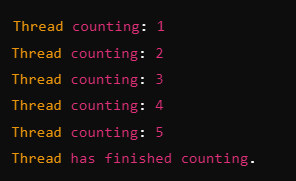
**retval:** Pointer to the return value of the terminated thread (can be NULL).

# Example:



Here, pthread\_join ensures the main thread waits for count\_numbers to finish before printing the final message.

## Output:



# pthread\_exit:

pthread\_exit() is used to terminate the calling thread. It allows a thread to exit without affecting the process or other threads.

## Syntax:

void pthread\_exit(void \*retval);

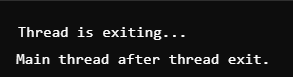
**retval:** The return value of the thread (can be NULL).

## Example:

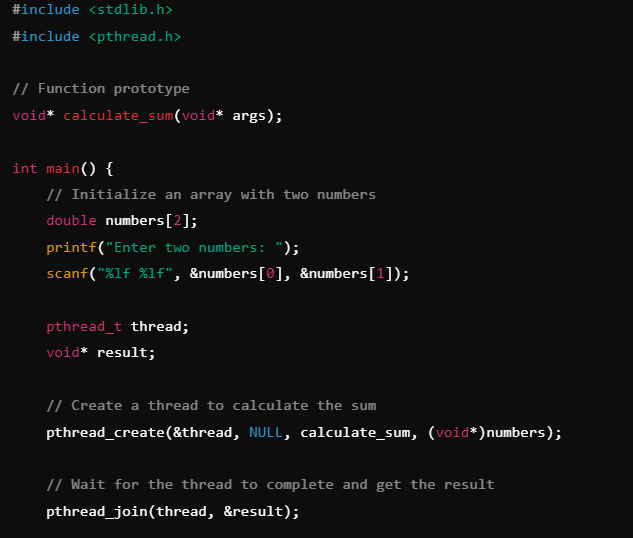


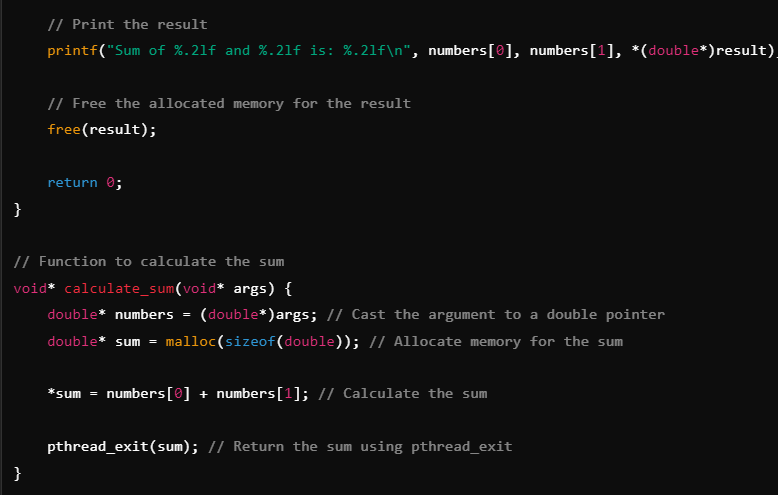
Here, the pthread\_exit function terminates the thread, and the main thread waits for its completion using pthread\_join.

## Output:

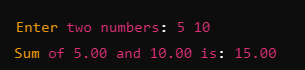


# pthread\_join() and pthread\_exit with non null arguments:



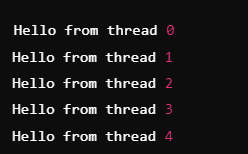


## Output:



# Tasks:

1. Create a program where multiple threads print "Hello from thread X", where X is the thread number. **( 20 mins )**



1. Create a program where array is pass to a thread and thread calculates the sum of elements in array and displays it. **( 25 mins )**
2. Modify task 2, where threads calculates the sum and return the result to main thread (main function) and displays the sum in main function. (don’t use global variables). Use pthread\_exit. **( 15 mins )**
3. Write a program that creates a thread to find the maximum value in an array. The array should be passed to the thread, and the maximum value should be returned from the thread. No global variables should be used. Don’t use pthread\_exit. **( 20 mins )**

# Task Submission Guidelines:

1. **Make a word document and paste the code with output’s screenshot there and save it as pdf or odt.**
2. **Include your name and roll no. at the front page.**
3. **Files other than pdf or odt will not be accepted and will be marked as 0.**