**Lab 13**

**Tayyaba Rehman**

**49690**

**Q1.**

|  |  |  |
| --- | --- | --- |
| **Features** | **Process** | **Thread** |
| **Basic Unit** | A process is an independent program in execution. | A thread is the smallest unit of execution within a process. |
| **Execution** | Each process has its own memory space. | Threads share the same memory within a process. |
| **Memory** | Separate memory allocation for each process. | Threads share code, data, and files of the process. |
| **Overhead** | Higher(due to memory isolation) | Lower (due to shared memory) |
| **Intercommunication** | Requires inter-process communication (IPC) like pipes, sockets, etc. | Easy through shared memory. |
| **Creation** | Time consuming and resources intensive. | Faster and requires fewer resources. |
| **Switching** | Slow due to memory and context switch. | Fast due to shared context. |
| **Example** | Running two applications like Google Chrome and Microsoft Word. Each runs in a separate process, has its own memory space, and doesn’t interfere with the other. | In Google Chrome, each tab or extension may run as a separate thread within the same process. They share the browser’s data like bookmarks, cache, etc. If one thread (tab) craches, it might freeze the entire browser (process). |

**Q2. Thread Management:**

Thread management functions are operations provided by threading libraries (like **POSIX pthreads**, **Java threads**, or **Windows threads**) to control the lifecycle and behavior of threads.

These functions include:

1. **Thread Creation**

Creates a new thread within a process.

**POSIX (C/C++):**

Pthread\_create(&thread\_id, NULL, function\_name, args);

1. **Thread Termination:**

Ends a thread when it’s task is complete.

**POSIX:**

Pthread\_exit(NULL);

1. **Thread Join:**

Waits for a thread to complete before continuing.

**POSIX:**

Pthread\_join(thread\_id, NULL);

1. **Thread Yield / Sleep:**

Temporarily pauses a thread to allow others to execute.

**POSIX:**

Sched\_yield();

1. **Thread Detach**:

Separates a thread so its resources are automatically released upon termination.

**POSIX:**

Pthread\_detach(thread\_id);

**Thread Attributes:**

Thread attributes define the behavior and configuration of a thread when it is created. In POSIX, these are specified using the pthread\_attr\_t structure.

1. **Stack Size:**

Specifies how much stack memory is allocated to the thread.

Default is usually sufficient, but can be increased for memory-intensive tasks.

1. **Scheduling Policy:**

Defines how the thread is scheduled by the CPU.

Common policies:

SCHED\_FIFO (First In First Out)

SCHED\_RR (Round Robin)

SCHED\_OTHER (Default)

1. **Priority:**

Sets the execution priority of the thread (works with scheduling policy).

Higher priority threads can preempt lower-priority ones.

1. **Detach State:**

Defines whether the thread is:

**Joinable** (can be waited on)

**Detached** (cannot be waited on, cleans itself)

**Code:**

Pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_DETACHED);

1. **Inheritance of Scheduling Attributes:**

Specifies whether a thread inherits its scheduling from the parent thread or uses explicitly set values.

**Code:**

Pthread\_attr\_setinheritsched(&attr,PTHREAD\_INHERIT\_SCHED);

1. **Guard Size:**

Defines extra memory at the end of the thread’s stack to protect against overflows.

**Example in POSIX Threads (C):**

Pthread\_t thread;

Pthread\_attr\_t attr;

Pthread\_attr\_init(&attr);

Pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_JOINABLE);

Pthread\_create(&thread, &attr, function\_name, NULL);

Pthread\_join(thread, NULL);