Mutex

```
C++ Code
#include <iostream>
#include <windows.h>

#define MUTEX_NAME "Global\\IPC_MUTEX"

using namespace std;

bool run(){
    HANDLE hMutex = CreateMutexA(NULL, FALSE, MUTEX_NAME);
    MessageBoxA(NULL, "Payload", "Payload1", MB_OK);
    if (hMutex == NULL) {
        cerr << "Error creating Mutex" << endl;
        return false;
    }
    return true;
}

bool itsAlreadyRunning(){
    HANDLE hMutex = OpenMutexA(MUTEX_ALL_ACCESS, FALSE, MUTEX_NAME);
    if (hMutex == NULL) {
        cerr << "Mutex dont exist" << endl;
        return false;
    }
    return true;
}

int main(){
    if (itsAlreadyRunning()){
        cerr << "Already running" << endl;
        getchar();
        return 1;
    }
    else{
        run();
    }
    getchar();
    return 0;
}
```

This C++ program demonstrates the use of a Mutex, a synchronization mechanism, in a Windows environment. The program has two main functions:

1. **`run()` Function**: This function creates a Mutex named "Global\IPC_MUTEX" using the `CreateMutexA` function. It then displays a message box with the title "Payload1" to simulate the payload execution. If creating the Mutex fails, it prints an error message and returns false.
2. **`itsAlreadyRunning()` Function**: This function checks if the Mutex named "Global\IPC_MUTEX" already exists using the `OpenMutexA` function. If the Mutex exists, it means another instance of the program is already running, and it returns true. Otherwise, it returns false.
3. **`main()` Function**: In the `main()` function:
    - It first checks if there is an instance of the program already running using the `itsAlreadyRunning()` function. If another instance is running, it prints "Already running" and waits for user input before exiting with a return code of 1.
    - If there is no other instance running, it calls the `run()` function, which creates the Mutex and simulates payload execution with a message box.
    - Finally, it waits for user input and returns 0, indicating successful execution.

The primary purpose of using the Mutex is to ensure that only one instance of the program (or payload) runs at a time. If another instance is detected, it will not proceed and display a message indicating that it's already running. This is a common technique used to prevent multiple instances of a program from interfering with each other.