The program has the presence of the **main thread** and the help of a **further 5 threads** which are spawned at Run-Time.
The software **works with 6 threads and two semaphore structures** to manage the atomicity of operations.
Using UNIX Semaphores V. (System V);
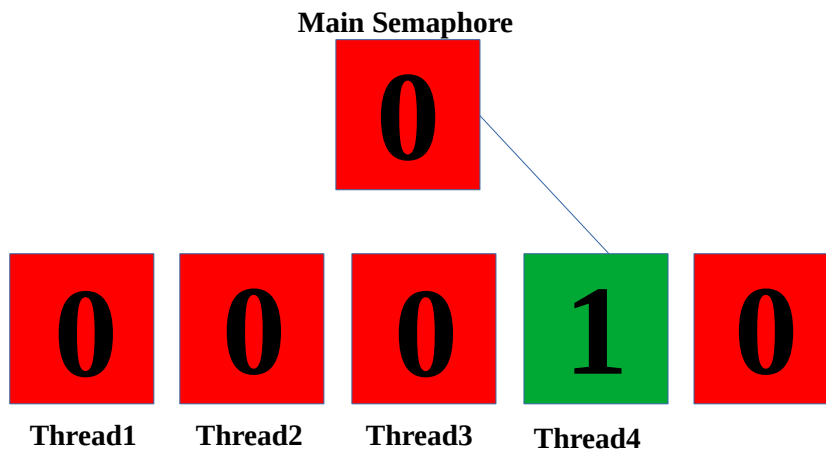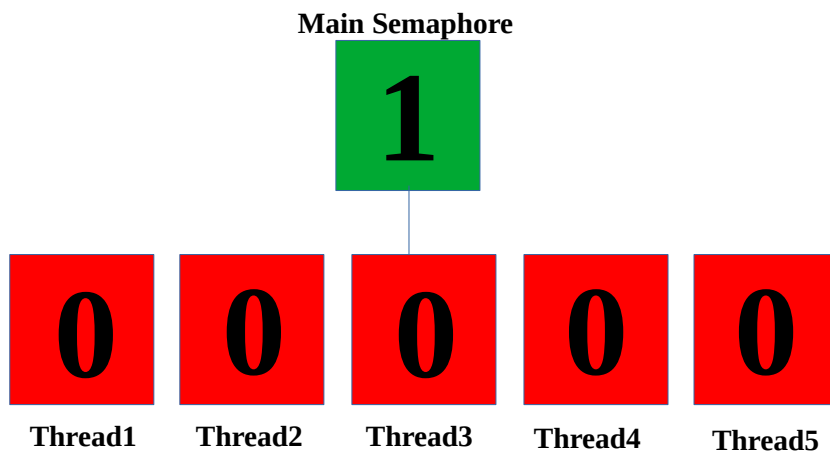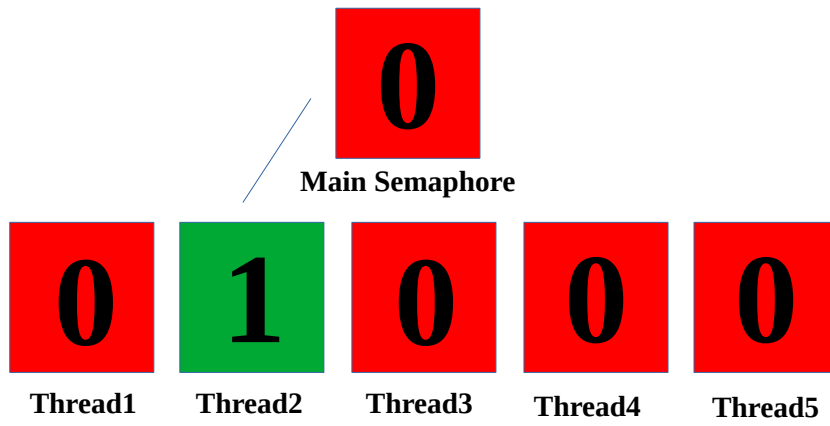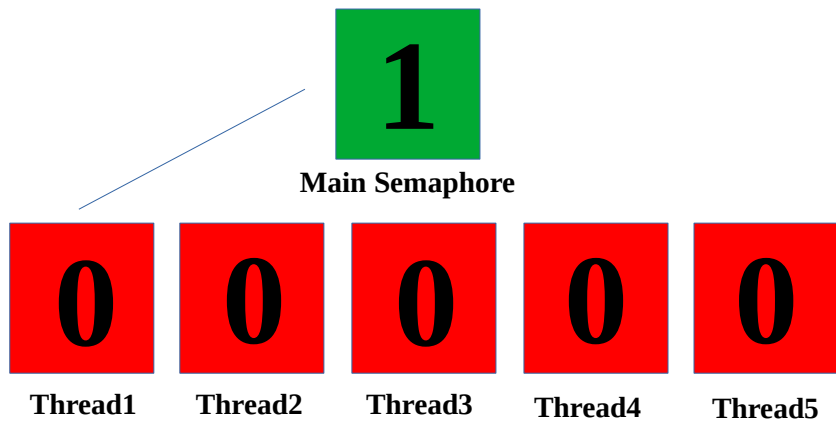The program is structured as follows:
- Thread 1 **writes** to file1;
- Thread 2 **writes** to file2;
- Thread 3 **reads** from file1;
- Thread 4 **reads** from file2;
- Thread 5 **"merges"** the contents of the two files and displays their contents.

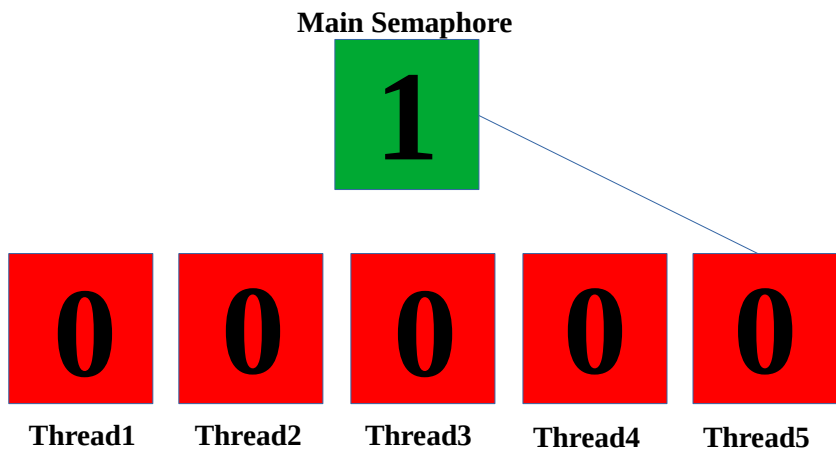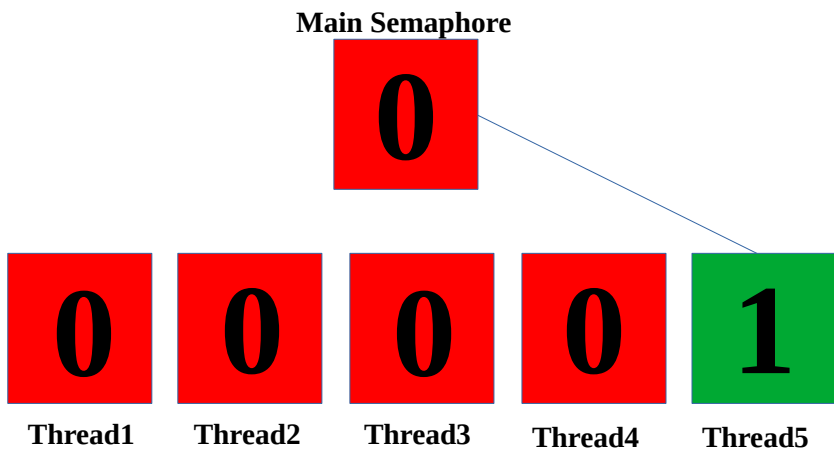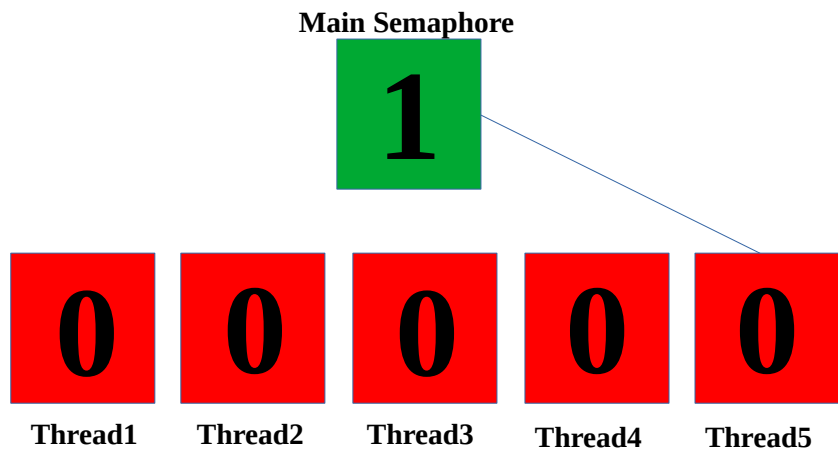Upon arrival of the SIGINT signal (CTRL + C) the contents of the "merged" file are completely transferred to STDOUT.

Semaphore structure for thread synchronization.
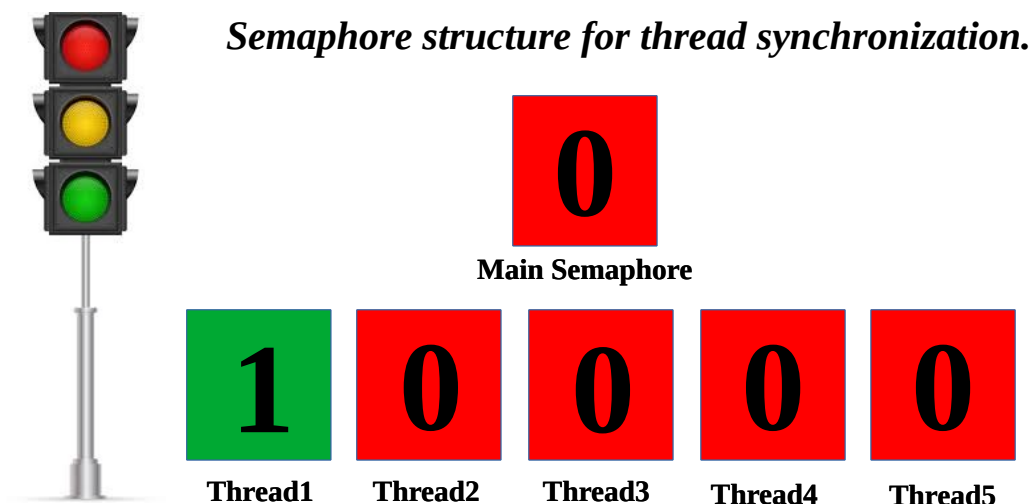
The semaphore managed by the main thread must immediately block to allow thread 1 to execute.
Thread 1 takes the available token.

1

Main Semaphore

0	**0**	**0**	**0**	**0**
Thread1	Thread2	Thread3	Thread4	Thread5

0

Main Semaphore

0	**1**	**0**	**0**	**0**
Thread1	Thread2	Thread3	Thread4	Thread5

Main Semaphore

1

0	**0**	**0**	**0**	**0**
Thread1	Thread2	Thread3	Thread4	Thread5

Main Semaphore

0

0	**0**	**0**	**1**	**0**
Thread1	Thread2	Thread3	Thread4	

Main Semaphore

1

0 **0** **0** **0** **0**

Thread1 Thread2 Thread3 Thread4 Thread5

Main Semaphore

0

0 **0** **0** **0** **1**

Thread1 Thread2 Thread3 Thread4 Thread5

Main Semaphore

1

0 **0** **0** **0** **0**

Thread1 Thread2 Thread3 Thread4 Thread5

Restart loop.