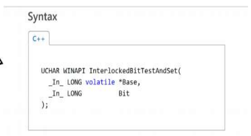
Sincronizzazione 5 Pthread spinlock WINDOWS

martedì 25 novembre 2025 11:47

Andiamo a vedere la gestione in maniera atomica in WINDOWS , sempre usando RMW ,le quali prendono il nome di **interlocked**:

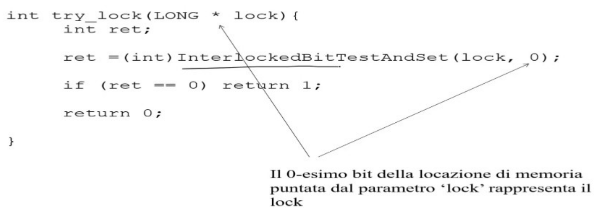
- ✓ `InterlockedCompareExchange`
- ✓ `InterlockedBitTestAndSet`



Syntax

```
UCHAR WINAPI InterlockedBitTestAndSet(
  _In_ LONG volatile *Base,
  _In_ LONG          Bit
);
```

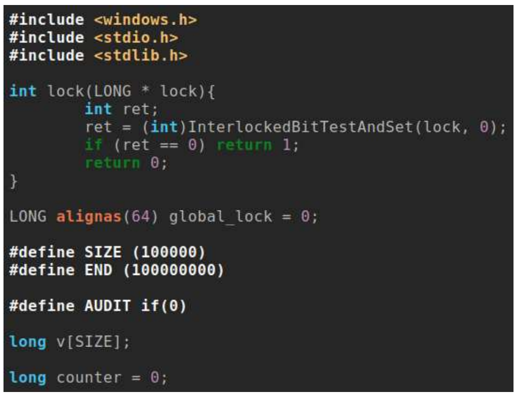
In dettaglio :



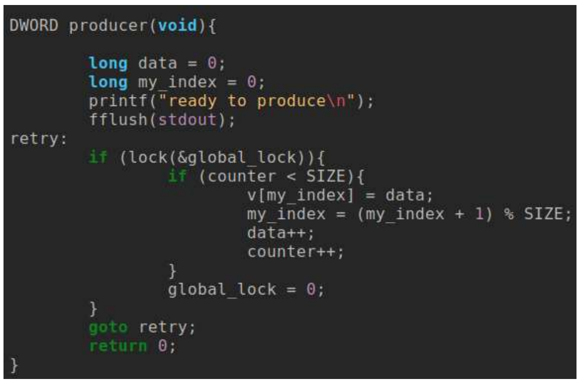
```
int try_lock(LONG * lock){
     int ret;

     ret =(int)InterlockedBitTestAndSet(lock, 0);

     if (ret == 0) return 1;

     return 0;

}
```

Il 0-esimo bit della locazione di memoria puntata dal parametro ‘lock’ rappresenta il lock

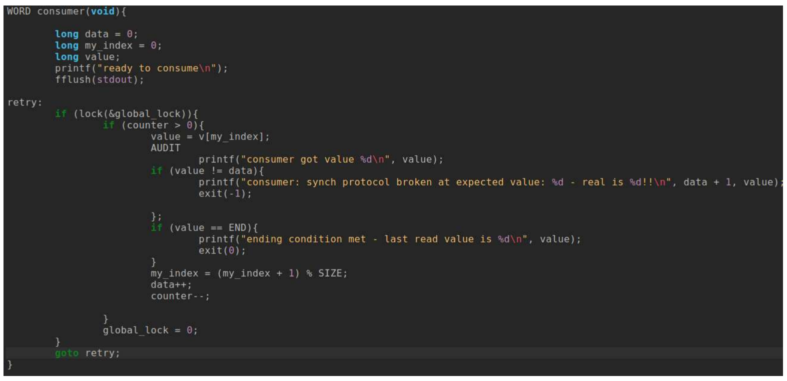
Vediamo ora degli esempi:



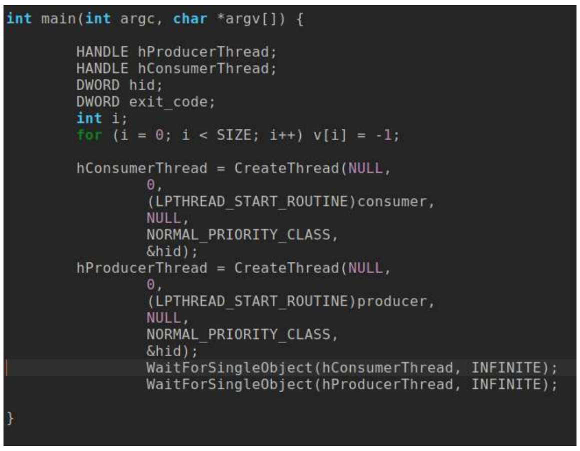
```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

int lock(LONG * lock){
        int ret;
        ret = (int)InterlockedBitTestAndSet(lock, 0);
        if (ret == 0) return 1;
        return 0;
}

LONG alignas(64) global_lock = 0;

#define SIZE (100000)
#define END (100000000)

#define AUDIT if(0)

long v[SIZE];

long counter = 0;
```



```
DWORD producer(void){

        long data = 0;
        long my_index = 0;
        printf("ready to produce\n");
        fflush(stdout);
retry:
        if (lock(&global_lock)){
                if (counter < SIZE){
                        v[my_index] = data;
                        my_index = (my_index + 1) % SIZE;
                        data++;
                        counter++;
                }
                global_lock = 0;
        }
        goto retry;
        return 0;
}
```



```
WORD consumer(void){

        long data = 0;
        long my_index = 0;
        long value;
        printf("ready to consume\n");
        fflush(stdout);

retry:
        if (lock(&global_lock)){
                if (counter > 0){
                        value = v[my_index];
                        AUDIT
                                printf("consumer got value %d\n", value);
                        if (value != data){
                                printf("consumer: synch protocol broken at expected value: %d - real is %d!!\n", data + 1, value);
                                exit(-1);

                        };
                        if (value == END){
                                printf("ending condition met - last read value is %d\n", value);
                                exit(0);
                        }
                        my_index = (my_index + 1) % SIZE;
                        data++;
                        counter--;

                }
                global_lock = 0;
        }
        goto retry;
}
```



```
int main(int argc, char *argv[]) {

        HANDLE hProducerThread;
        HANDLE hConsumerThread;
        DWORD hid;
        DWORD exit_code;
        int i;
        for (i = 0; i < SIZE; i++) v[i] = -1;

        hConsumerThread = CreateThread(NULL,
                0,
                (LPTHREAD_START_ROUTINE)consumer,
                NULL,
                NORMAL_PRIORITY_CLASS,
                &hid);
        hProducerThread = CreateThread(NULL,
                0,
                (LPTHREAD_START_ROUTINE)producer,
                NULL,
                NORMAL_PRIORITY_CLASS,
                &hid);
                WaitForSingleObject(hConsumerThread, INFINITE);
                WaitForSingleObject(hProducerThread, INFINITE);

}
```