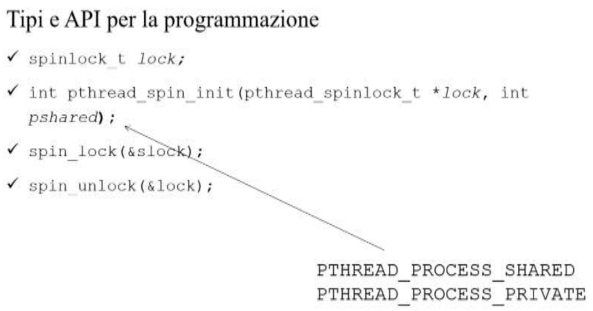
Sincronizzazione 4 Pthread spinlock UNIX

martedì 25 novembre 2025 11:38

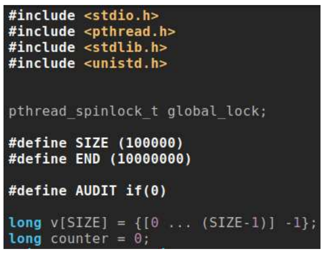
Vediamo ora in dettaglio come usare questi costrutti :



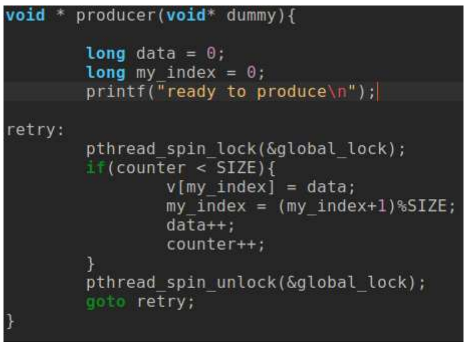
Tipi e API per la programmazione

```
✓ spinlock_t lock;
✓ int pthread_spin_init(pthread_spinlock_t *lock, int
  pshared);
✓ spin_lock(&slock);
✓ spin_unlock(&lock);

                              PTHREAD_PROCESS_SHARED
                              PTHREAD_PROCESS_PRIVATE
```

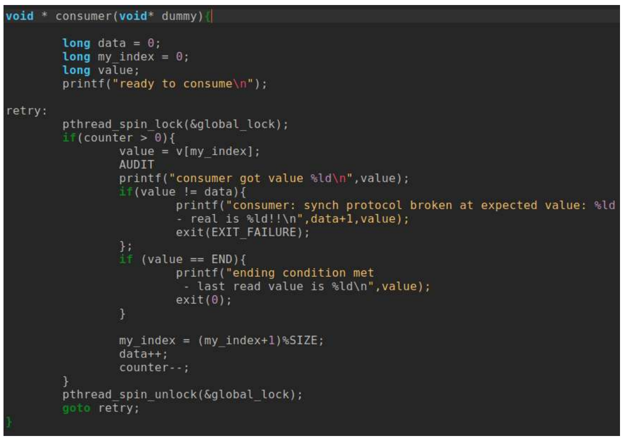
<u>Dove privato funziona tra tutti i thread della stessa applicazione.</u> Vediamo un esempio :



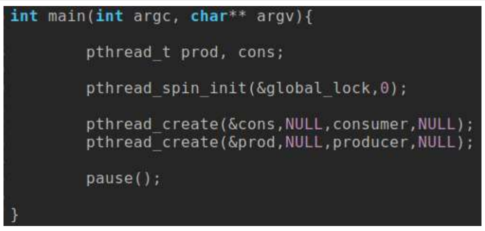
```c
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <unistd.h>


pthread_spinlock_t global_lock;

#define SIZE (100000)
#define END (10000000)

#define AUDIT if(0)

long v[SIZE] = {[0 ... (SIZE-1)] -1};
long counter = 0;
```



```c
void * producer(void* dummy){

        long data = 0;
        long my_index = 0;
        printf("ready to produce\n");

retry:
        pthread_spin_lock(&global_lock);
        if(counter < SIZE){
                v[my_index] = data;
                my_index = (my_index+1)%SIZE;
                data++;
                counter++;
        }
        pthread_spin_unlock(&global_lock);
        goto retry;
}
```



```c
void * consumer(void* dummy){

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

retry:
        pthread_spin_lock(&global_lock);
        if(counter > 0){
                value = v[my_index];
                AUDIT
                printf("consumer got value %ld\n",value);
                if(value != data){
                        printf("consumer: synch protocol broken at expected value: %ld
                        - real is %ld!!\n",data+1,value);
                        exit(EXIT_FAILURE);
                };
                if (value == END){
                        printf("ending condition met
                         - last read value is %ld\n",value);
                        exit(0);
                }

                my_index = (my_index+1)%SIZE;
                data++;
                counter--;
        }
        pthread_spin_unlock(&global_lock);
        goto retry;
}
```



```c
int main(int argc, char** argv){

        pthread_t prod, cons;

        pthread_spin_init(&global_lock,0);

        pthread_create(&cons,NULL,consumer,NULL);
        pthread_create(&prod,NULL,producer,NULL);

        pause();

}
```

Anche in questo caso corretta sincronizzazione.