

Makefile:

CXX = g++

CXXFLAGS = -std=c++17

LDFLAG = -pthread

make t3 → Compile

Motex

#include <mutex>

mutex l;

Stupid way (C++ non ha "supporto" per)

l.lock()

l.unlock()

→ se c'è una sleep (1) → d tempo che posso e' redi, sente le lock non possono durare i secondi: corretti.

l.lock()

l.unlock()

if (...) { } → return(); → que va fatto l'unlock altrimenti rimane con le lock e non sblocca le lock.

Ci sono vari tipi di lock:

→ unique_lock

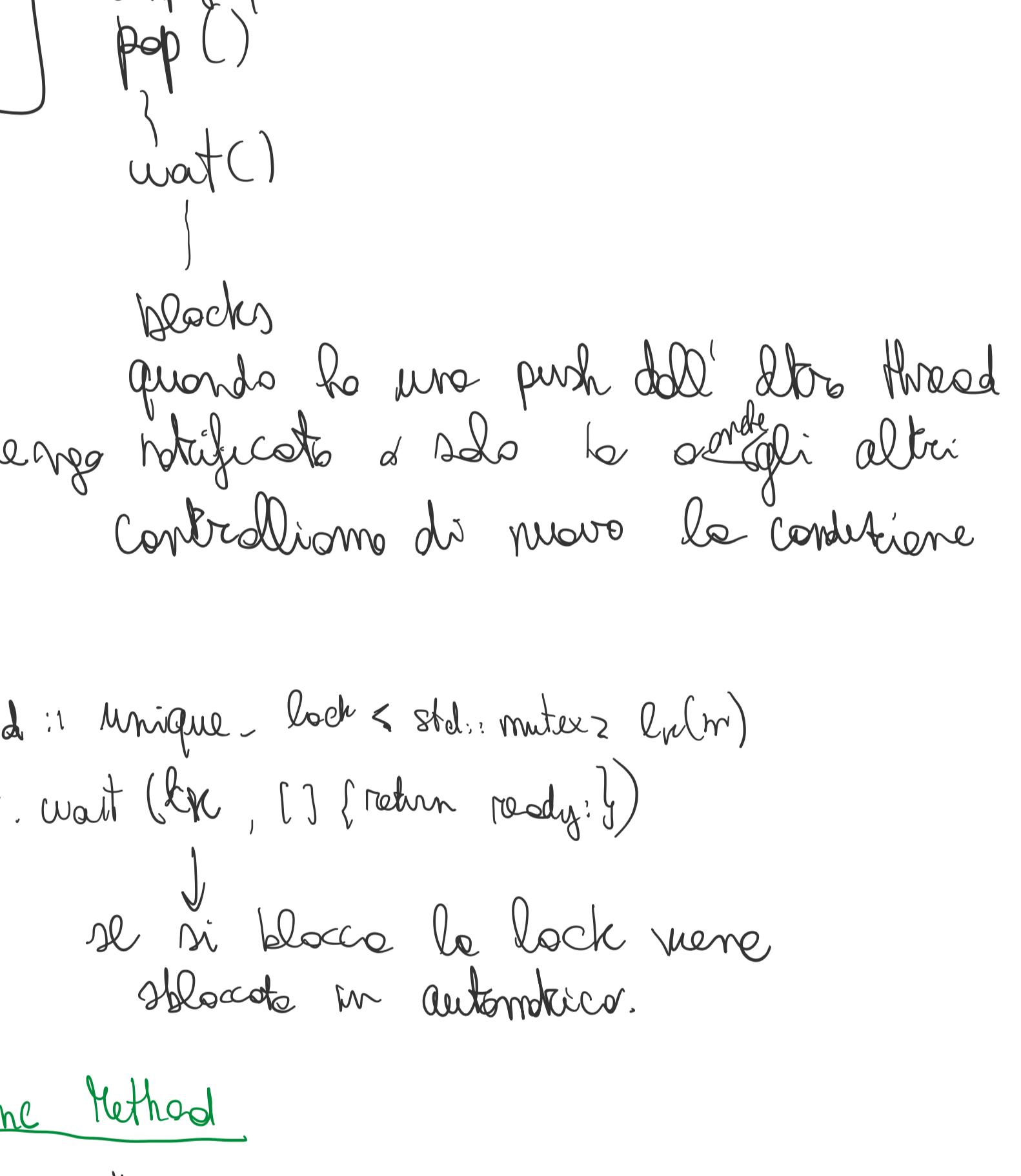
→ guard_lock

→ Scope_lock

Cppreference.com

lock_guard → Scope lock è simile → preferibile l'uso di queste lock al posto delle lock classiche quando si esce dallo scope la routine viene rilasciata automaticamente

Unique_lock → usare con le condition variable perché garantisce delle proprietà che non sentono se usare la condition variable.

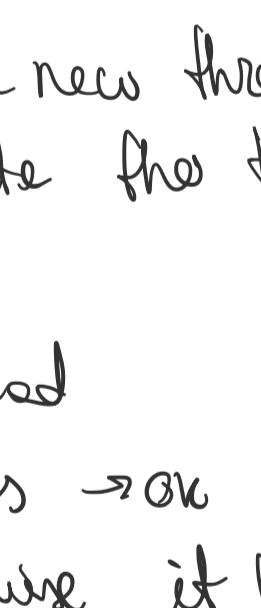
Situazioni più complesse

Queue

→ push()

→ pop()

→ empty()

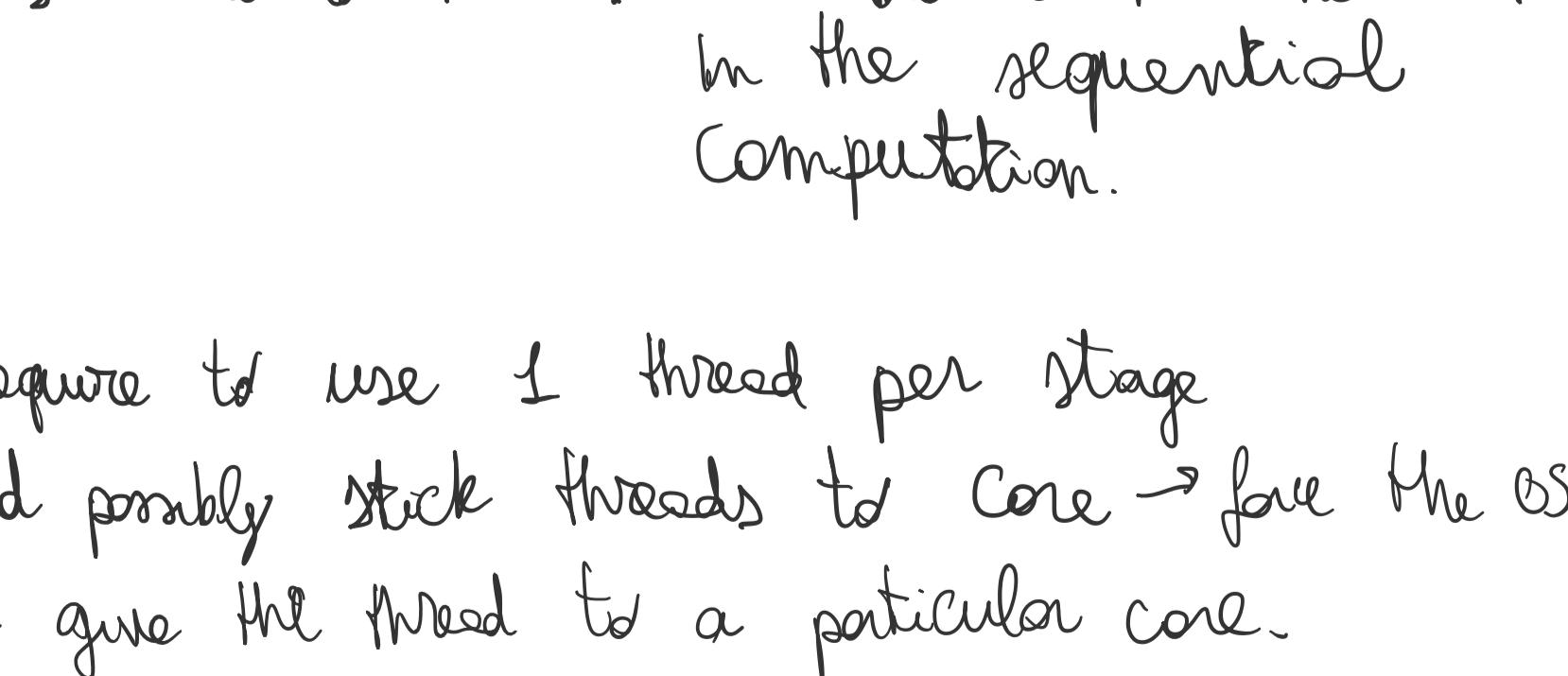


Devo gestire la sincronizzazione.

When I pop() se io de qualsiasi ci sarà ma se è vuota non voglio gestire da solo l'attesa.

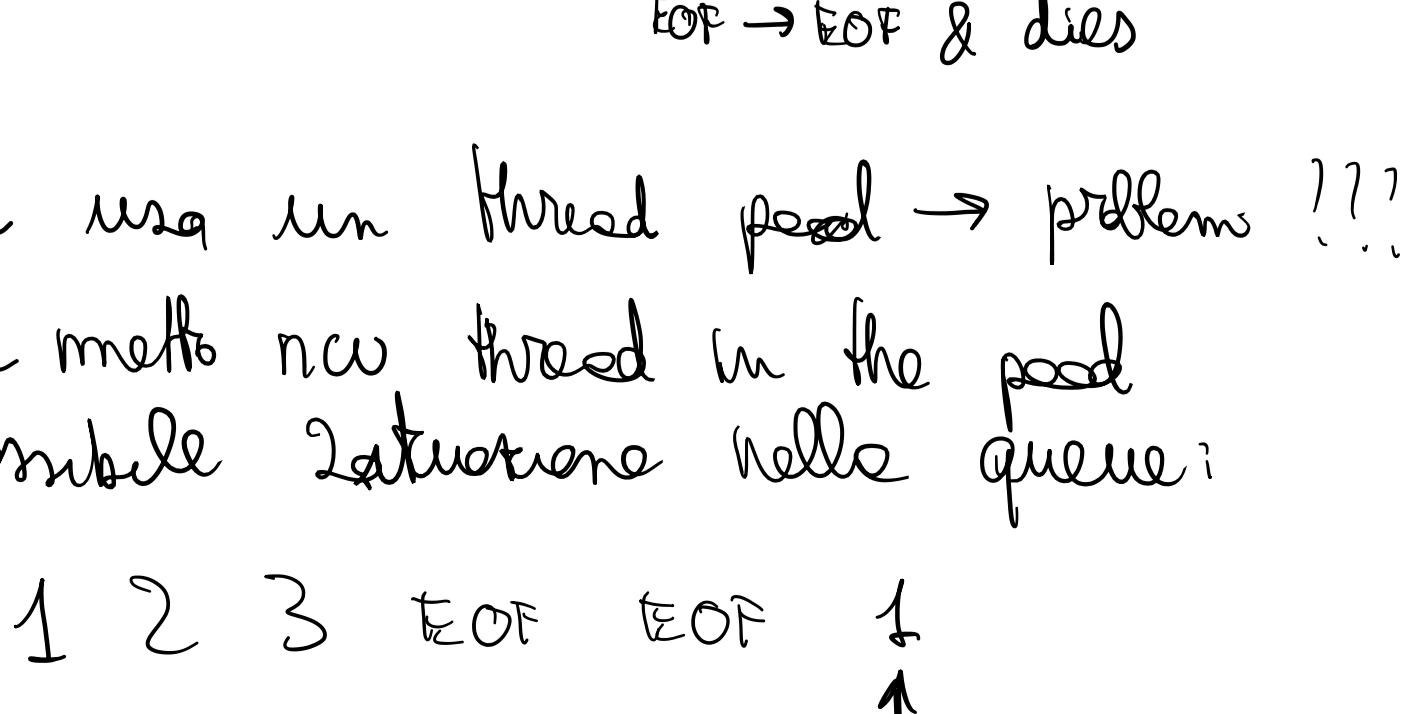
Si usa una condition variable

2 operation → wait: controlla se non c'è la condizione mi ferma → notify: notifica l'altra thread.



Se uso un thread pool → problemi ??!

Se metto n. c. thread in the pool possibile saturazione nelle queue:



Stream are represented with an "EOF"

→ si potrebbe usare lo 0 come EOF

for m = 1 ... 321

top → EOF & dies

i → i+1

top → EOF & dies