C++ 11
- GNU
- clang
- MS
- Intel (2016/17)

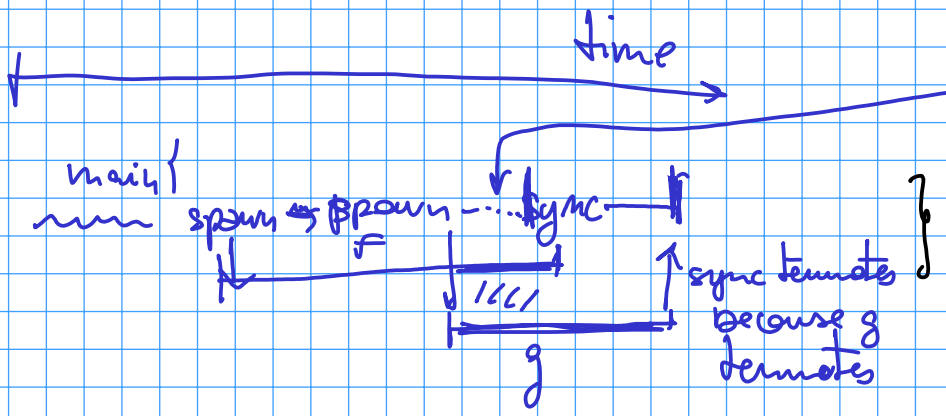
CILK g++ -fcilkplus

cilk_spawn

```
void f(...) { ... }
void g(...) { ... }
```

cilk_sync

```
main( ) {
  {
  cilk_spawn f(1, "a");
  cilk_spawn g(1.234);
  :
  cilk_sync;
}
```

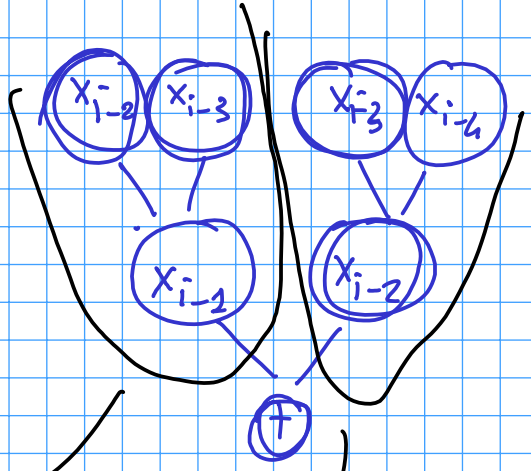


1 1

x_1 x_2

$x_i = x_{i-1} + x_{i-2}$

```
int fib(int i) {
  if(i<3) return (1)
  else return (fib(i-1) + f(i-2))
}
```



return ((spawn (fib (i-1))) + (spawn (fib (i-2))))

```
int x1 = spawn (fib (i-1))
int x2 = spawn (fib (i-2))
sync;
return (x1 + x2);
```

```
int x1 = spawn (fib (i-1))
int x2 = fib (i-2);
Sync;
return (x1 + x2);
```

C++11 threads (C++14 C++17)

```
void body ( ... ) { ⊗ }
```

```
int x = 321;          // x is shared among thread t1 & t2
main ( ... ) {
```

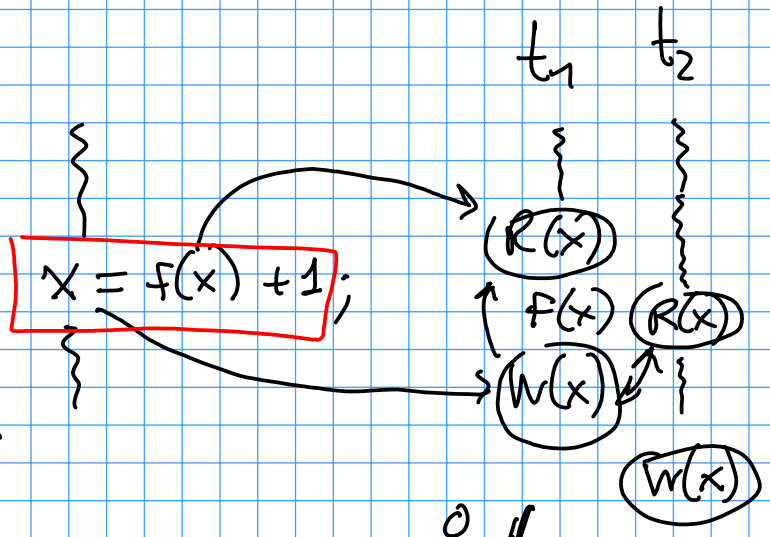
```
thread t1 = thread ( body ( 1, "a", 123.4 ) )
thread t2 = thread ( body ( 2, "b", 7.a ) )
```

starts the thread

```
t1.join();
```

waits the termination of exactly t1

```
void body ( ) {
    x = f(x) + 1;
}
```



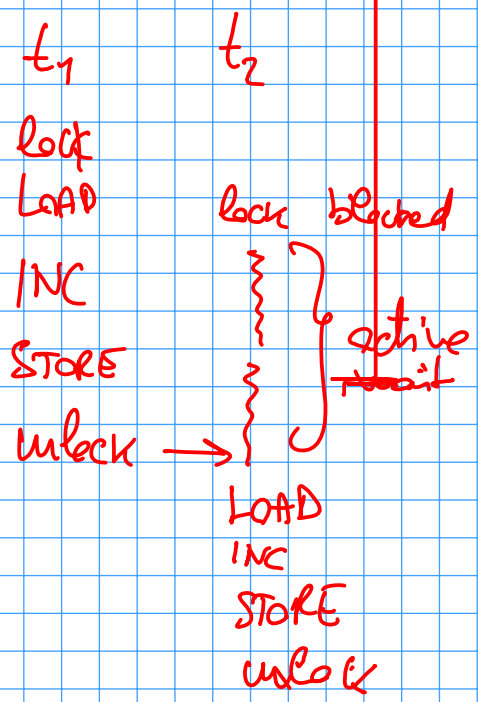
$X = X + 1$

```
LOAD  X → R1
INC   R1
STORE X ← R1
```

```
LOAD  0
INC   1
STORE 1
```

mutexes
- .lock()
- .unlock()

```
mutex m;
body( ) {
    m.lock();
    x = f(x) + 1;
    m.unlock()
}
```



try_lock()

Condition variables

- wait()
- notify()
- notifyAll()

---

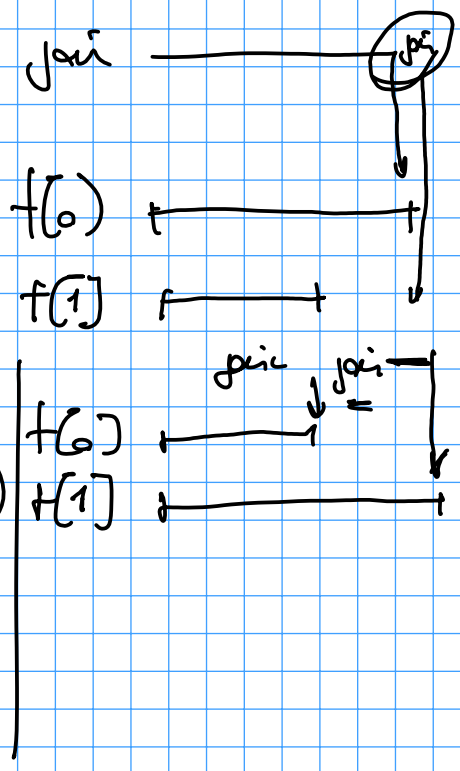
if( ) then
else

spawn
spawn

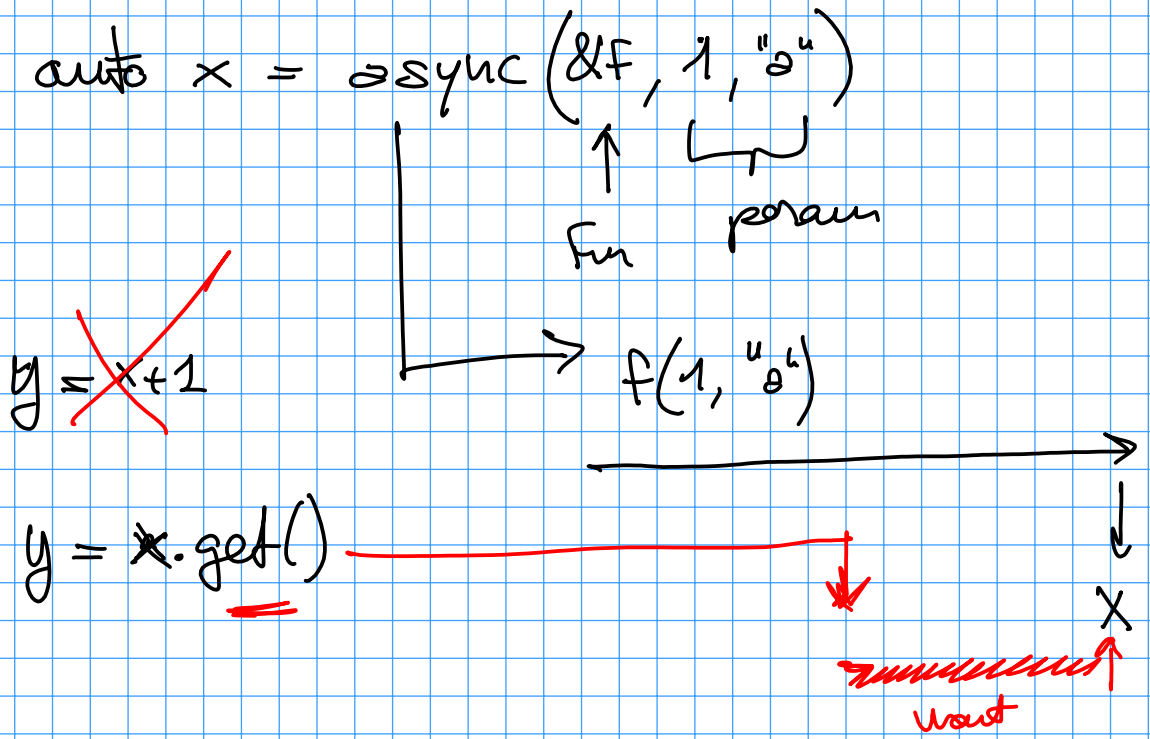
sync

```
t[0]
    thread( )
t[1]
    thread( )

for(i=0; i<2; i++)
    t[i].join();
```



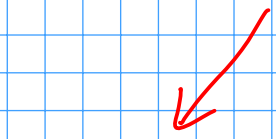
async



Book = char[ ]

Translation = Capital letters → non capital

This is my Book → this is my book



CLR

C++ libraries