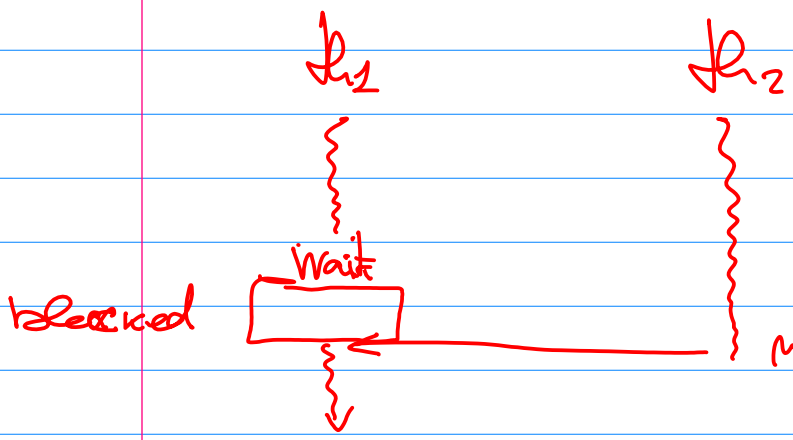
mutex < lock() / unlock() / try_lock()

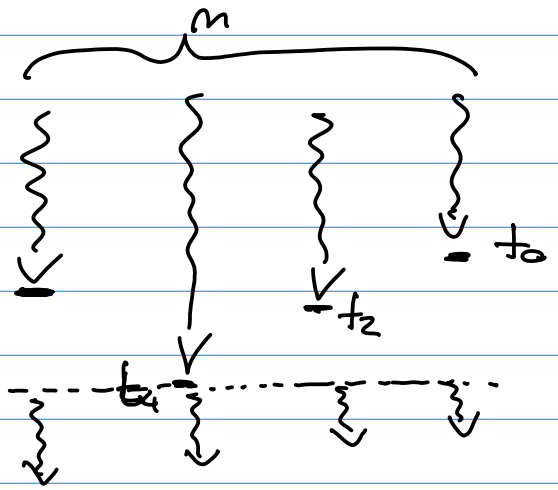
condition_variable < wait / notify_all



barrier(n)

experimental::barrier

#include <exp.../barrier>



lock

counter = n

cond_variable

th(n-1)

counter--

wait()

th(n)

n == 1

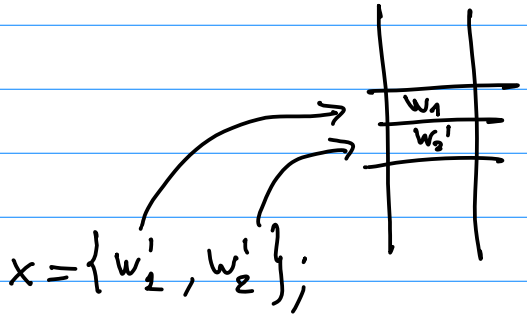
notify_all()

atomic

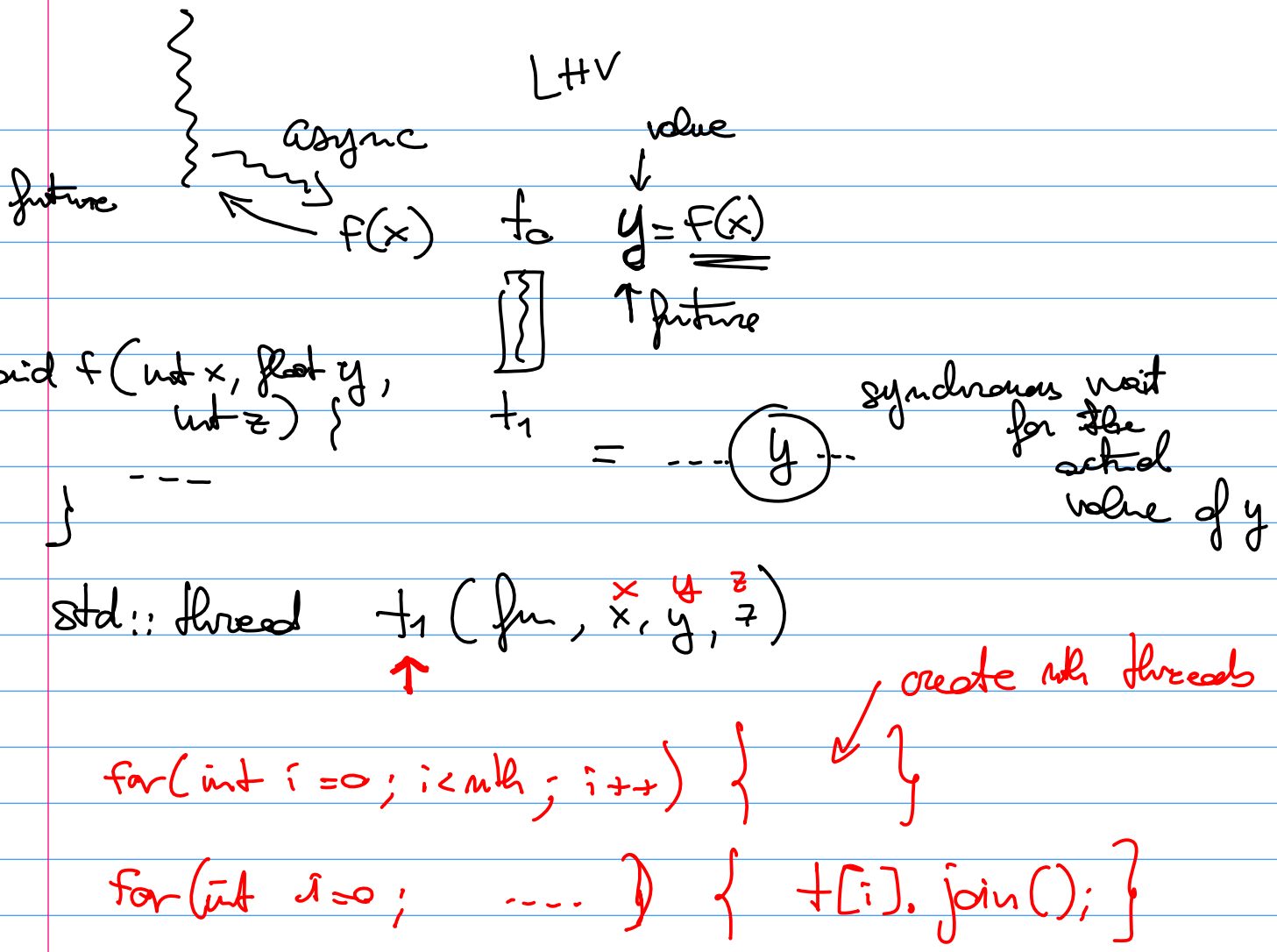
typedef ... dw;

dw x;

x = {w1, w2};

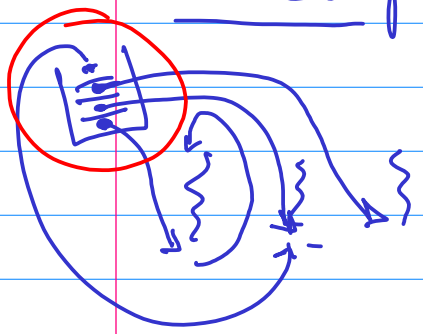


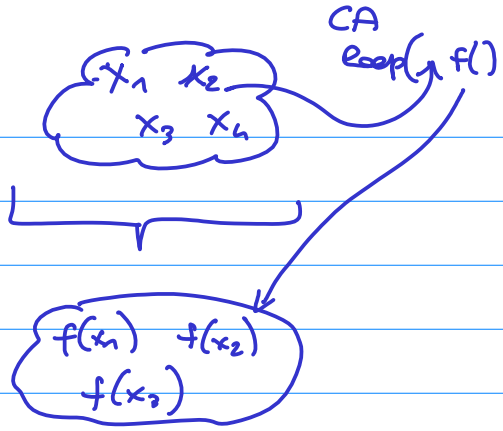
x = {w2', w2'};


LHV
async
future
f(x)
to
value
y = F(x)
future
t1
= --- y ---
synchronous wait for the actual value of y
void f(int x, float y, int z) {

}
std::thread t1 (fn, x, y, z)
create nth threads
for(int i=0; i<nth; i++) { }
for(int i=0; ....) { t[i].join(); }

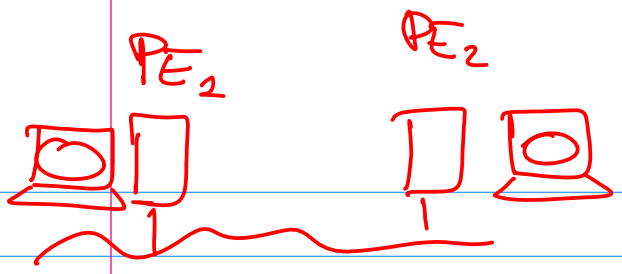
thread pool concept





$x : \alpha$ $\quad$ $f : \alpha \to \beta$

$$\text{body}\left(\underset{\alpha \to \beta}{\text{fun}}, \underset{\alpha}{x}\right)$$



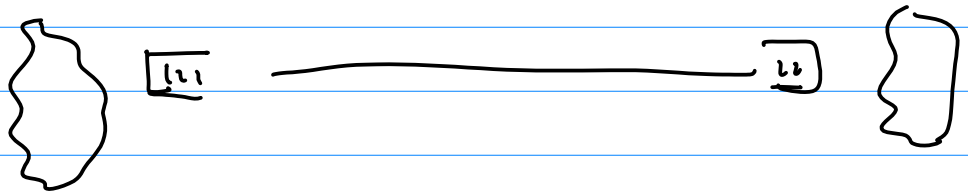
> ssh <host> cmd param*

≡

open a server socket > open a client socket

PE1 > myprogA portno

PE1 > ssh PE2 myprogB IP PE1 portno



RPC (remote procedure call)

PE1

res = myproc(x, y, z)

x, y, z

PE2

z y x

myproc( )

res

res

x = 123;

int x[3];

&x[0], sizeof(int) * 3;

client

connect

write (req)

read (answ)

&x, sizeof(x)

server socket

bind

accept

socket

read ( ) → rqst

write ( )

&(y), sizeof(y)

CORBA