

```

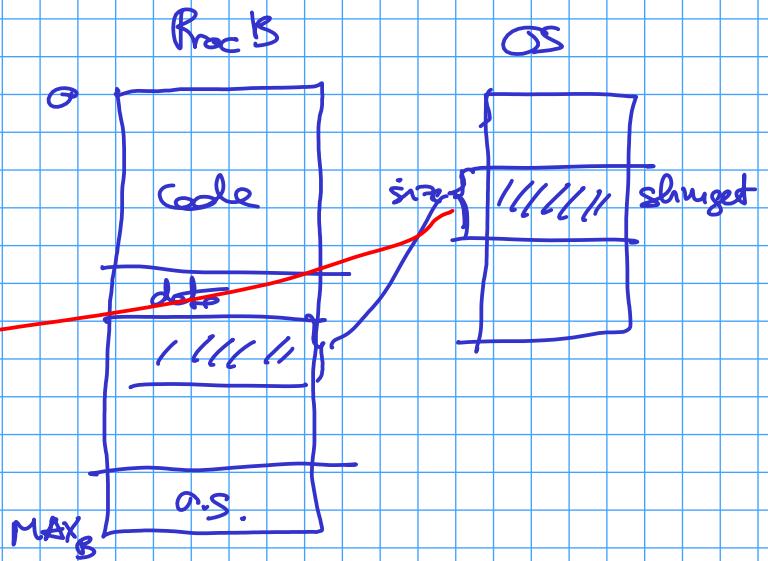
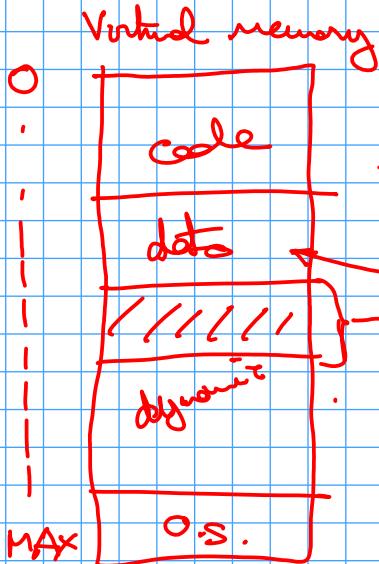
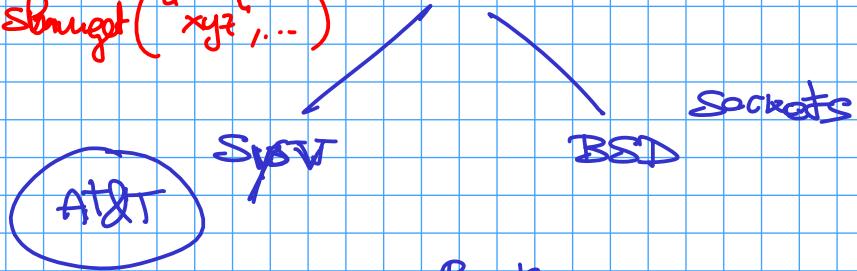
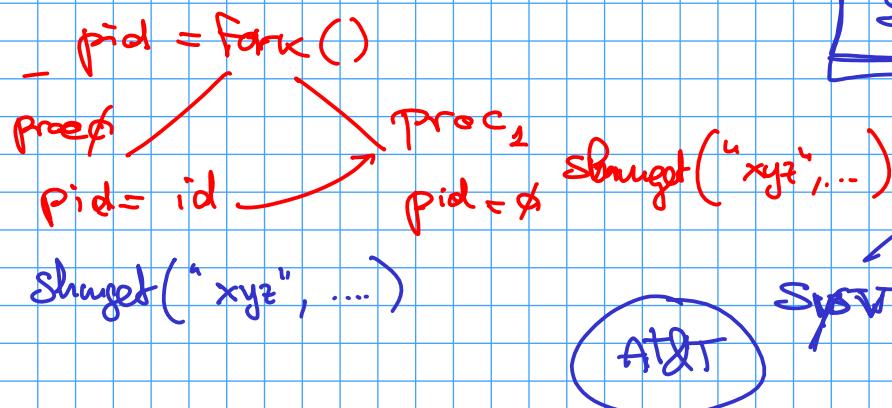
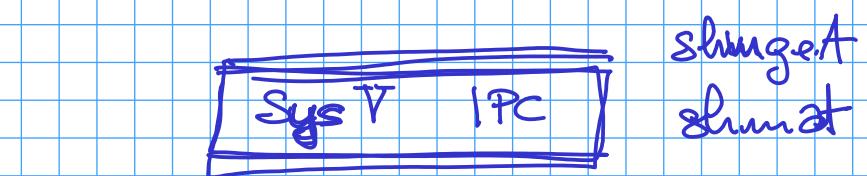
#include <iostream>
#include <thread>
#include <vector>

using namespace std;

void body(int n) {
    for(int i=0; i<n; i++)
        cout << "This is thread " << n << endl;
    return;
}

int main(int argc, char * argv[]) {
    int n = atoi(argv[1]); // segv if not present ...
    vector<thread> t;
    for(int i=0; i<n; i++)
        t.push_back(thread(body,i));
    return(0);
}

```



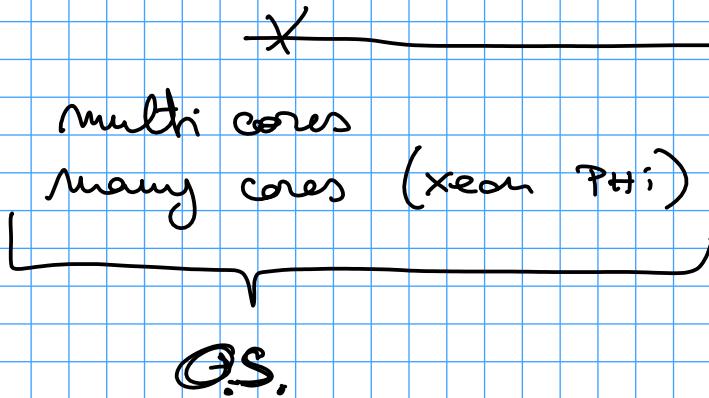
Sys V IPC

shmat
shmrd
shmrdt

semget
 $P()$
 $V()$

msgget

send
receive



libraries / frameworks

CUDA OpenCL
ACC

Sponsoring a computation
on a CPU

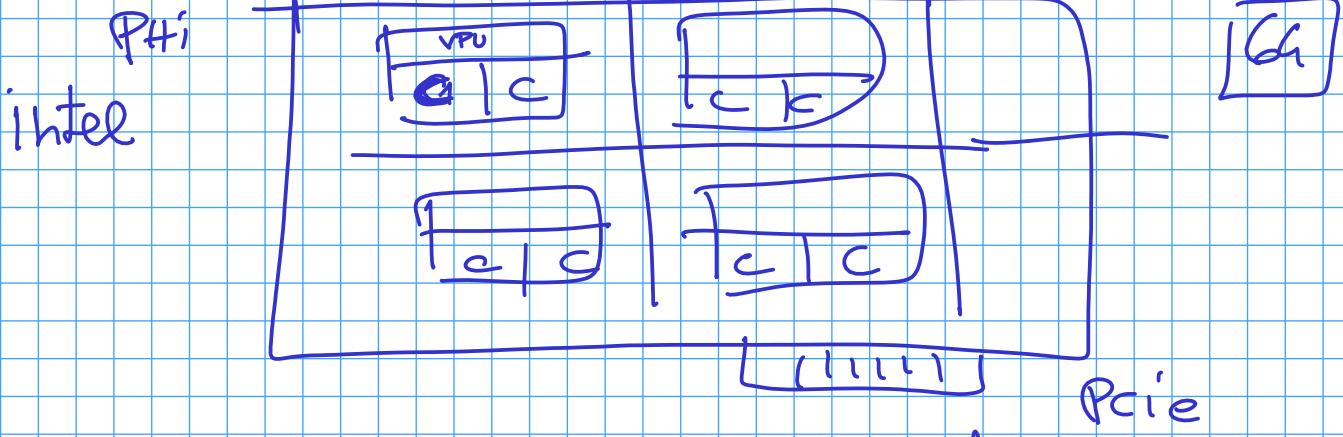
cudaMalloc
memcpy
 $\ll x, y, z \gg$ fun

- allocates memory on device
- move input data to device
- start a comp. on the device
- move back results from device to CPU memory

float sq (float x) {
 return x*x; }

~~(float*~~ v) {

int i = my index
v[i] = ~~v[i]~~ * v[i]; }



```

int main( ) {
    for(int i=0 ; i<n ; i++) {
        #pragma offload
        // code
    }
}

```

(mic)

A red bracket groups the loop body, with a label "#pragma offload" pointing to it. Another red bracket groups the entire loop, with a label "for loop offloaded" pointing to it. A red curly brace at the bottom right indicates the loop's scope.

