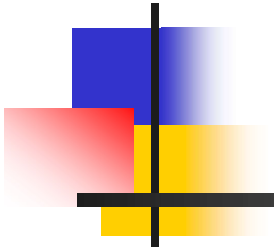


Calcolo Parallelo e Distribuito

Laboratorio OpenMP in ambiente Linux

Docente: Prof. L. Marcellino

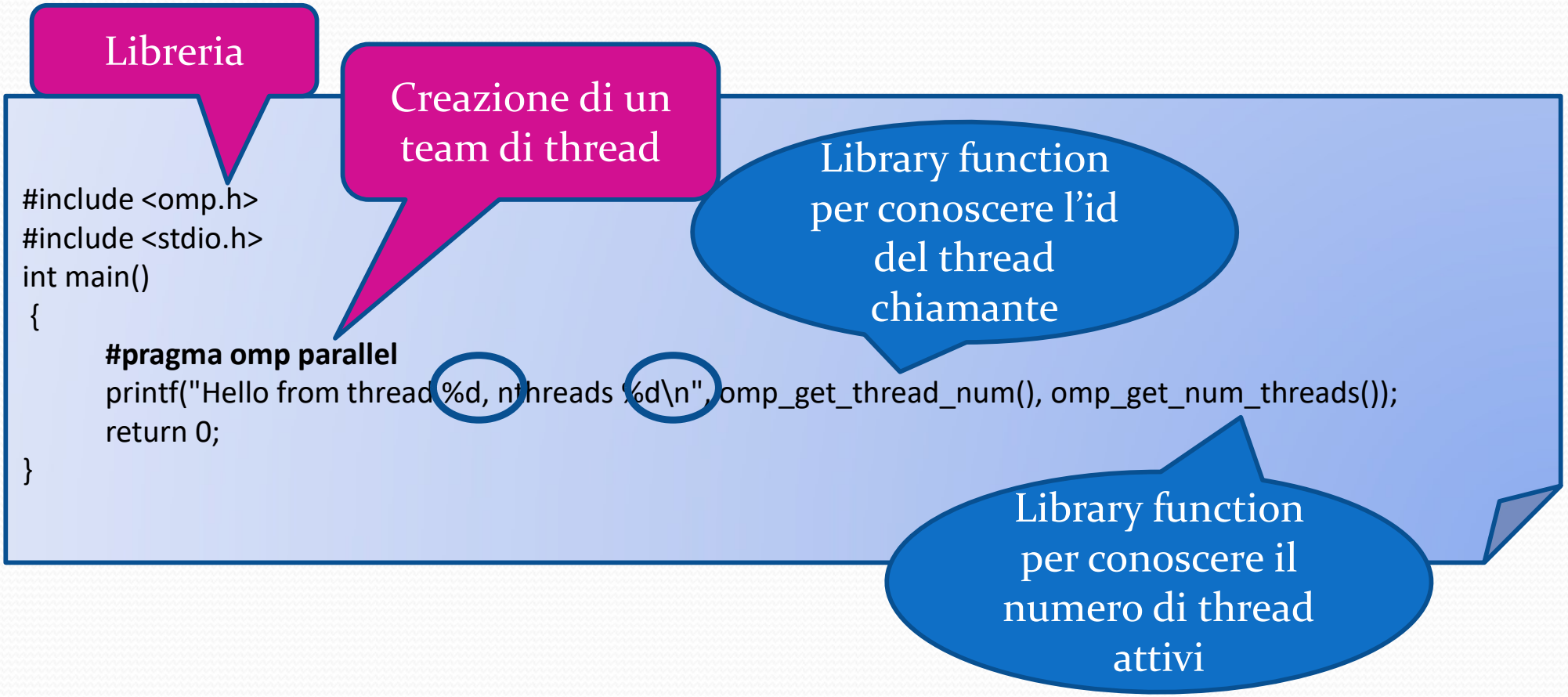
Tutor: Prof. P. De Luca

Finalmente cominciamo…

il laboratorio

OpenMp

Esempio: Hello World



Ambiente MIMD-SM

potreste anche usare i vostri PC che sicuramente sono MULTICORE (quad-octa core)…

… lavorando con il sistema operativo LINUX

cat /proc/cpuinfo **modello CPU**

Compilare ed eseguire

- OpenMp viene implementato da molti compilatori, tra questi il gcc (v. 4 e superiori)
- Per compilare basta
  - aggiungere al comando di compilazione l'opzione –fopenmp

gcc -fopenmp -o *nome-eseguibile* nome-codice.c

Compilare ed eseguire

- Una volta compilato il codice, basta lanciare l'eseguibile come di consueto

```
./nome-eseguibile
```

per default sul mio PC parte con 8 thread

- Si possono modificare prima le variabili d'ambiente proprie dello standard OpenMp, come visto.

```
export OMP_NUM_THREADS=2
```

Esempio: Hello World

Proviamo insieme a fare una leggera modifica al programma!!!

```
#include <omp.h>
#include <stdio.h>
int main()
 {        // dichiarazione variabili
          // Dichiariamo per gestire la stampa in maniera più esplicita:
          -    Una variabile per l'identificativo del thread
          -    Una variabile per il numero di thread in esecuzione

          #pragma omp parallel // definizione delle variabili shared o private
          {   // aggiungiamo le parentesi graffe per creare un blocco di istruzioni


          printf("Hello from thread %d, nthreads %d\n", <nuove variabili>);

          } // fine blocco direttiva

          return 0;
}
```

Provate a farlo da soli!

Ma attenzione a chi deve essere shared e chi invece private!

Esempio: Hello World

Esercizio da fare da soli

```
#include <omp.h>
#include <stdio.h>
int main()
 {        // dichiarazione variabili
         // Dichiariamo per gestire la stampa in maniera più esplicita:
         -    Una variabile per l'identificativo del thread
         -    Una variabile per il numero di thread in esecuzione

         #pragma omp parallel // definizione delle variabili shared o private
         {  // aggiungiamo le parentesi graffe per creare un blocco di istruzioni


         printf("Hello from thread %d, nthreads %d\n", <nuove variabili>);

         } // fine

         return 0;
}
```

Avete provato?

Esempio: Hello World

Io l'avevo pensato così…

```
#include <omp.h>
#include <stdio.h>
int main()
 {
       int id_thread, num_threads;

         #pragma omp parallel private(id_thread), shared (num_threads)
         {
                  id_thread = omp_get_thread_num();
                  num_threads = omp_get_num_threads();

         printf("Hello from thread %d, nthreads %d\n", id_thread, num_threads);

         }

         return 0;
}
```

Compilare ed eseguire

```
>> gcc –fopenmp –o hello hello.c

>> export OMP_NUM_THREADS=2

>> ./hello
```

Avete provato a far variare il numero dei core?