Process and Thread

* An executing instance of a program is called a process

• Process has its independent memory space

• A thread is a subset of the process – also called lightweight process allowing faster context switching

• Threads share memory space within process’s memory

• Threads may have some (usually small) private data

• A thread is an independent instruction stream, thus allowing concurrent operation

• In OpenMP one usually wants no more than one thread per core

OpenMP Introduction

* Open Specification for Multi Processing
* Provides multi-threaded parallelism
* It is an specification for

Directives

Runtime Library Routines

Environment Variables

* OpenMP is an Application Program Interface (API) for writing multithreaded, shared memory parallelism.
* Easy to create multi-threaded programs in C,C++ .

pragma vs language

* pragma is not language, should not express logics.
* To provide compiler/preprocessor additional information on how to processing directive annotated code
* Similar to #include, #define

**Lab Assignment 1**

**Problem statement 1:**

BMI and find Thread number and master thread in c using OMP function.

**Code:**

#include <omp.h>

#include <stdio.h>

int main(){

int tid, h, w, bmi;

printf("Enter height and weight: ");

scanf("%d %d",&h, &w);

#pragma omp parallel

tid = omp\_get\_thread\_num();

bmi = (w\*703)/(h\*h);

printf("My thread is = %d\n",tid);

/\*only master thread does this\*/

if(tid == 0){

printf("Number of threads = %d\n",tid);

}

printf("Threads is %d and BMI is %d\n",tid,bmi);

printf("Number of processors available = %d\n",omp\_get\_num\_procs());

}

**Problem statement 2:**

Find sum of 2 arrays and print the result in third array and find Thread number and master thread in c using OMP function.

**Code:**

#include <omp.h>

#include <stdio.h>

int main(){

int i, n;

printf("Enter the number of elements of array: ");

scanf("%d", &n);

int a[n], b[n], c[n];

printf("Enter the elements of the 1st array:\n");

for(i = 1; i <= n; i++){

scanf("%d", &a[i]);

}

printf("Enter the elements of the 2nd array:\n");

for(i = 1; i <= n; i++){

scanf("%d", &b[i]);

}

#pragma omp parallel for

for(i = 1; i <= n; i++){

c[i] = a[i]+b[i];

}

int tid = omp\_get\_thread\_num();

printf("My thread is = %d\n", tid);

if(tid == 0){

printf("Number of threads = %d\n", tid);

}

for(i = 1; i <= n; i++)

printf("Thread is %d and Array element is %d\n", tid, c[i]);

printf("Number of processors available = %d\n", omp\_get\_num\_procs());

}

**Problem statement 3:**

Print sum of odd numbers and even numbers in an array and find Thread number and master thread in c using OMP function.

**Code:**

#include<stdio.h>

#include<omp.h>

int main(){

int n, i, sum1 = 0, sum2 = 0;

printf("Enter the size of the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter the elements of the array:\n");

for(i = 0; i < n; i++){

printf("Element %d: ", i+1);

scanf("%d", &arr[i]);

}

#pragma omp parallel for

for(i = 0; i < n; i++){

int thread\_id = omp\_get\_thread\_num();

if(arr[i]%2 == 1){

printf("Thread is %d.\n", thread\_id);

sum1 += arr[i];

}

}

for(i = 0; i < n; i++){

int thread\_id = omp\_get\_thread\_num();

if(arr[i]%2 == 0){

printf("Thread is %d.\n", thread\_id);

sum2 += arr[i];

}

}

printf("Number of processors available is: %d\n", omp\_get\_num\_procs());

printf("Sum of the odd entries is: %d\n", sum1);

printf("Sum of the even entries is: %d\n", sum2);

printf("\n\nAssignment 1\nTapabrata Roy\n23MAI1032\n");

return 0;

}