1 Multiprocessing

Symmetric multiprocessing (SMP) referes to computer hardware where two or more identical processors are connected to a single shared main memory and are controlled by a single instance of an operating system. Most ordinary computers today use the SMP architecture.

When a program uses two or more processors—to share the workload and speedup execution—on an SMP computer it is broadly refered to as *multiprocessing* or *parallel computing*.

A part of the program that runs on a single processor is refered to in this context as a *thread*. Multiprocessing is generally achieved when the master thread of a program forks off a number of extra threads which execute blocks of code in parallel on the available processors.

1.1 Pthreads

The POSIX standard defines an application programming interface—usually referred to as Posix threads or Pthreads—for creating and manipulating threads. Here is an example,

```
#include<pthread.h> // gcc -pthread
#include<math.h>    // -lm
#include<stdio.h>
void* bar(void* arg){
  double *x=(double*)arg;
  for(int i=0;i<1e8;i++) *x=cos(*x); // Do your stuff here.
  return NULL;}
int main() {
pthread_t thread; double x=0,y=100;
int flag=pthread_create(            // Create another thread
   &thread,NULL,bar,(void*)&x);     // and run "bar" in it.
bar((void*)&y);                     // Meanwhile in the master thread...
flag = pthread_join(thread,NULL);   // Join threads.
printf("x=%g\ny=%g\n",x,y);
return 0; }
```

1.2 OpenMP

One relatively easy way to do multiprocessing is to use "OpenMP" – an industry standard programming interface that supports shared-memory multiprocessing programming in C, C++, and Fortran. The GNU compilers gcc, g++, and gfortran support the latest OpenMP specifictaion (as do several others compilers).

With OpenMP the user simply marks the sections of code that are meant to run in parallel with the corresponding preprocessor directives and the compiler does all the low-level programming for creating the thread-tasks and running the threads.

In C/C++ OpenMP markings are done with “#pragma omp” preprocessor directive, in Fortran77 with “C$OMP” and in Fortran90 with “!$omp”. The directives must be on their own lines.

The full OpenMP specification is available from “openmp.org”.

Here is an simple example of running two chunks of code in parallel,

```
#include <omp.h>                      // -fopenmp -lgomp
#include <math.h>                     // -lm
#include <stdio.h>
int main()
{
double x=0,y=100;
#pragma omp parallel sections
// the following sections wil be run parallelly in separate threads
    {
    #pragma omp section  // first thread will run this block of code
        {
        for(int i=0;i<1e8;i++) x=cos(x);
        }
    #pragma omp section  // second thread will run this block of code
        {
        for(int i=0;i<1e8;i++) y=cos(y);
        }
    }
printf("x=%g y=%g\n",x,y);
}
```