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 an 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 *thread*. Multiprocessing is gerenrally 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.

Probably the easiest way to do multiprocessing is to use the so-called 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).

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 we shall only consider a simple example of running two chunks of code in parallel:

```
#include <omp.h>                    // to be built with -fopenmp -lgomp
#include <math.h>                   // -lm
#include <stdio.h>
int main()
{
double x=0,y=0;
#pragma omp parallel sections
// the following sections of code wil be run parallelly in separate threads
   {
   #pragma omp section          // first thread will run this block of code
      {
      // do something useful here, for example:
      for(int i=0;i<1000000;i++) x=cos(x);
      }
   #pragma omp section           // second thread will run this block of code
      {
      // do something useful here in parallel, for example:
      for(int i=0;i<1000000;i++) y=cos(y);
      }
   }
printf("x=%g y=%g\n",x,y);
}
```