Conceptual Questions

1. Explain what OpenMP is, what are it's benefits?

OpenMP (Open Multi-processing) is an API for writing multithreaded applications. It contains compiler directives and libraries that help write parallel applications. OpenMP simplifes multithreaded programming for Fortran, C and C++. OpenMP standardizes the last 20 years of symmetric multiprocessing.

2. What are the #Pragma definitions, what do they do?

#pragma definitions are compiler directives. They are used to define parllel regions of the code. Aby code that is inside the pragma definition will be run by all of the threads that are specified in the program.

3. Write the OpenMP #pragma definition to execute a loop in parallel

```
#pragma omp parallel num_threads(4) {
   #pragma omp for
   for (i=0; i<N; i++)
      // do something
}
```

4. What does the reduction do in the #pragma definition to execute a loop in parallel?

Reduction is used when we have variables that multiple threads are changing throughout the parallel loop. To handle this, we use `reductuon(operation:var)`, where *operation* is how we want to deal with the variables and *var* is the variable name.

5. Explain the critical and private() declarations used in OpenMP.

the critical declaration defines a critical section in a parllel section of code. This section will be mutally exclusive and only allow 1 thread to execute it at a time. The private declaration specifies variables that a parallel section should allocate for each individual thread, rather than share among threads. Each thread will have it's own instances of the variables defined as private.

Application Questions

All code and output can be found here.

Link to clone repository: git@github.com:AshwinK97/Operating-Systems.git