# 实验五 基于OpenMP的并行计算实验

## 一、实验目的

在Linux系统中，基于C++编写OpenMP用例对并行计算进行实验：

1. 掌握OpenMP的配置方式；
2. 通过编写测试用例，观察引入OpenMP的加速效果。

## 二、实验内容

### 2.1 配置并测试OpenMP

1. 配置环境变量

配置环境变量，设置线程数：

```
export OMP_NUM_THREADS={{.num.of.threads}}
```

> 一般根据机器的核心数量（使用`htop`命令）进行设置。

使用`source`命令生效该环境变量。

2. 编写测试用例

编写一个简单的测试用例：

```
#include <iostream>
#include <omp.h>

using namespace std;

int main(){
#pragma omp parallel
  {
    cout << "Test" << endl;
  }
  return 0;
}
```

3. 编译并运行

编译上述测试用例：

```
g++ -fopenmp filename.cpp -o filename
```

执行生成的可执行文件，其结果如下所示：

```
Test
Test
Test
Test
```

> 由于设置的线程数为4，因此共输出4次。

### 2.2 编写测试用例

通过编写一个测试用例来说明并行计算的加速效果。该用例有一个简单的函数`test()`，在`main()`中用一个`for`循环把这个函数重复运行`8`次。

首先编写单核运行的程序，并加入计时功能：

```
#include <iostream>
#include <sys/time.h>

using namespace std;

void test(){
  int res = 0;
  for (int i=0; i<100000000; i++)
    res++;
}

int main(){
  struct timeval t1, t2;
  double timeuse;
  gettimeofday(&t1, NULL);

  for (int i=0; i<8; i++)
    test();

  gettimeofday(&t2, NULL);
  timeuse = (t2.tv_sec - t1.tv_sec) + (double)(t2.tv_usec -
t1.tv_usec)/1000000.0;
  cout << "time elapse: " << timeuse << " sec"  << endl;

  return 0;
}
```

编译运行后，打印出消耗时间。

引入OpenMP，将上述代码转换为多核运行：

```
#include <iostream>
#include <sys/time.h>
#include <omp.h>

using namespace std;

void test(){
  int res = 0;
  for (int i=0; i<100000000; i++)
    res++;
}

int main(){
  struct timeval t1, t2;
  double timeuse;
  gettimeofday(&t1, NULL);
```

```
  omp_set_num_threads(4);
#pragma omp parallel for
    for (int i=0; i<8; i++)
      test();

  gettimeofday(&t2, NULL);
  timeuse = (t2.tv_sec - t1.tv_sec) + (double)(t2.tv_usec -
t1.tv_usec)/1000000.0;
  cout << "time elapse: " << timeuse << " sec"  << endl;

  return 0;
}
```

编译运行后，打印出消耗时间。比较两次运行的耗时。