lab3实验报告

姓名：张浩南
学号：181860134

命令行工具

通过main函数中传递的argc和argv参数来控制命令行，如果有数字传递就读入运行次数round和运行函数func，否则默认执行一次
（要先输入`-r`选项）

```
char *opt=argv[1];

int rounds=1;
if (argc>=3&&opt[0]=='-'&&opt[1]=='r')
{
  if (argc==4)
  {
    sscanf(argv[2],"%d",&rounds);
    void (*func)()= lookup(argv[3]);
    run(func, rounds);
  }
  else
  {
    void (*func)()= lookup(argv[2]);
    run(func, rounds);
  }
}
```

最终用时的相关信息直接输出到终端上

获取精确时间

使用`clock()`函数，通过在调用func前后记录时刻来获得运行时间（相减后单位为us）
通过设置较大rounds并取平均值的方式来减小运行时间误差

```
static int gettime() {
  // TODO: implement me!
  return clock();
}
```

测试性能

运行时间的相关数据有总和sum,最小/大值minimum/maximum，平均值average，中位数median，方差variance，标准差standard，斜度alpha与峰度beta（单位：us）

检测方式

这里选择的编译选项为makefile中的编译选项，即

```
gcc -m64 -O1 -lm -std=gnu11 -ggdb -Wall -Werror -Wno-unused-result *.c... -o perf-64
```

注意这里还多加了一个-lm选项（因为要调用math库中的数学函数）
采取随机数生成的方式，每种计算方法分别使用5000000组随机数据
其中控制a,b,m大小的随机数生成函数为`int64_t get(int64_t lim);`（在`impl.c`中）
控制a,b二进制1的个数的随机数生成函数为`int64_t bitget(int id)`（在`_rand.c`中）

控制a,b,m大小

1.当$0 \leq a,b,m < 2^{62}$时，
`multimod_p1`性能情况

```
sum:17007557us
maximum:123us minimum:1us
average:3.401511us
median:3us
variance:2.425170us
standard:1.557296us
Negative skew//表示负斜度
Peak kurtosis//表示尖锐峰度
```

`multimod_p2`性能情况

```
sum:5554345us
maximum:68us minimum:0us
average:1.110869us
median:1us
variance:0.439550us
standard:0.662986us
Negative skew
Peak kurtosis
```

`multimod_p3`性能情况

```
sum:2291724us
maximum:58us minimum:0us
average:0.458345us
median:0us
variance:0.319934us
standard:0.565627us
Negative skew
Peak kurtosis
```

注意到整个用时分布相较于正态分布而言偏向负斜度（小数据占比大）且更为尖锐（较多数据集中于较小区间），且p1,p2,p3的数据稳定性逐渐更好.
p2效率相较于p1提高了3倍，p1相较于p2提高了2倍多.
2.当$0 \leq a,b,m < 2^{31}$时，
`multimod_p1`性能情况

```
sum:9770056us
maximum:81us minimum:1us
average:1.954011us
median:2us
variance:0.525249us
standard:0.724741us
Negative skew
Peak kurtosis
```

`multimod_p2`性能情况

```
sum:4247286us
maximum:127us minimum:0us
average:0.849457us
median:1us
variance:0.309786us
standard:0.556584us
```

```
Negative skew
Peak kurtosis
```

multimod_p3 性能情况

```
sum:2459891us
maximum:72us minimum:0us
average:0.491978us
median:0us
variance:0.345007us
standard:0.587373us
Negative skew
Peak kurtosis
```

3.当$0 \leq a, b, m < 2^{16}$时，
multimod_p1 性能情况

```
sum:5718213us
maximum:197us minimum:0us
average:1.143643us
median:1us
variance:0.396368us
standard:0.629577us
Negative skew
Peak kurtosis
```

multimod_p2 性能情况

```
sum:3201027us
maximum:107us minimum:0us
average:0.640205us
median:1us
variance:0.372683us
standard:0.610478us
Negative skew
Peak kurtosis
```

multimod_p3 性能情况

```
sum:2363819us
maximum:63us minimum:0us
average:0.472764us
median:0us
variance:0.339983us
standard:0.583081us
Negative skew
Peak kurtosis
```

情况2,3相对于情况1而言，p1，p2的用时逐步降低，尤其p1降低明显（这主要是由于对应的实现方法在小数据下的较优秀表现）。而p3的用时则较为稳定（且无论如何都比p1,p2方法用时更少）
且情况2,3中p1，p2的数据稳定性更为优秀，但仍保持着负斜度和尖锐峰度的特点，可见这里的数据分布基本不符合正态分布。
情况1,2,3中p1,p2,p3顺次两者之间始终保持着2~3倍左右的效率差距，可见3种方法在效率上的优劣。

控制a,b中二进制1的个数

同样采用5000000组随机数的方式，但考虑将64位数分为8个8bit，通过控制各部分中1的个数来调整a,b中1的个数，用1的个数整体将测试数据分为[0,7],[8,15],[16,24],…,[56,63]共8组
由于组数较多，这里的数据统计只考虑总和与平均值

multimod_p1

```
//[0,7]
sum:21422168us
average:4.284434us
//[8,15]
sum:26530676us
average:5.306135us
//[16,23]
sum:27539180us
average:5.507836us
//[24,31]
sum:28210120us
average:5.642024us
//[32,39]
sum:28717889us
average:5.743578us
//[40,47]
sum:28197697us
average:5.639539us
//[48,55]
sum:31031354us
average:6.206271us
//[56,63]
sum:29558596us
average:5.911719us
```

值得注意的是本次用时普遍高于之前对a,b,m范围大小研究时的用时，但这主要是由于数据生成时更加复杂的运算导致，因此我们只关注本次用时的相互比较。
p1方法使用的是类似高精度乘法的方式，从实验数据来看随着1个数的增加，整体上用时呈上升趋势，但在后面用时逐渐趋于平稳甚至略有降低，整体值域区间大概在21s~31s左右。
multimod_p2

```
//[0,7]
sum:5633026us
average:1.126605us
//[8,15]
sum:6212598us
average:1.242520us
//[16,23]
sum:7139664us
average:1.427933us
//[24,31]
sum:7623316us
average:1.524663us
//[32,39]
sum:8051906us
average:1.610381us
//[40,47]
sum:7942985us
average:1.588597us
//[48,55]
sum:8209284us
average:1.641857us
//[56,63]
sum:8002419us
average:1.600484us
```

p2采用类似快速幂的做法，用时较p1更小，且与p1类似，随着1个数的增加，整体上用时呈上升趋势，但在后面用时逐渐趋于平稳甚至略有降低，整体值域区间大概在5.6s~8.2s左右，从平均用时上看该方法较p1提高了四倍多。
multimod_p3

```
//[0,7]
sum:3063362us
average:0.612672us
```

```
//[8,15]
sum:3196314us
average:0.639263us
//[16,23]
sum:3326717us
average:0.665343us
//[24,31]
sum:3437839us
average:0.687568us
//[32,39]
sum:3595336us
average:0.719067us
//[40,47]
sum:3416381us
average:0.683276us
//[48,55]
sum:3359402us
average:0.671880us
//[56,63]
sum:3116139us
average:0.623228us
```

p3采用自然溢出的方法，用时较p2更小，且与前面两种方法类似，，随着1个数的增加，整体上用时呈上升趋势，但在后面用时逐渐趋于平稳甚至略有降低，整体值域区间大概在3.0s~3.6s左右，从平均用时上看较p2又提升了两倍多，较符合比较a,b,m大小时关于效率的结论.
总的来说，1的个数对三种方法都有较大影响，且当1个数较少时运行速度均较快，而效率的最低值并非1的个数最多的时候，而是中间偏右的位置（1的个数较多而非最多）.