根据蒙特卡罗方法求 PI

方法 1：

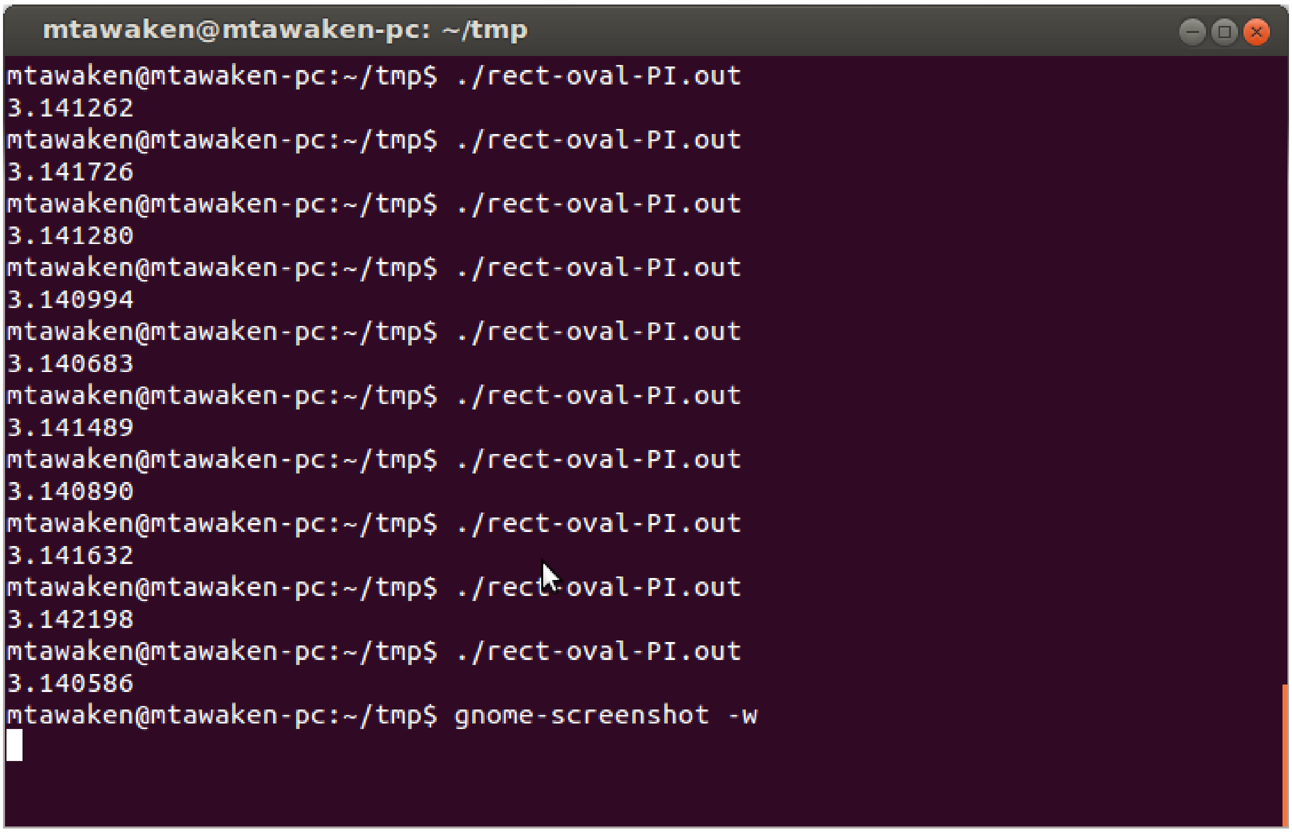
1.1 原理：

建立图形：以原点为圆心，R=1 的单位圆和它的外切正方形。在正方形范围内随机撒点，则随着次数增长，在圆内点的个数 m 与撒点的总数 n 之间存在关系 m/n = PI * R^2 / $(2R)^2$ 。从而求得 PI。

1.2 代码：

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define N 10000000
int main(){
    srand(time(NULL));
    int i,m;
    float x,y;
    m=0;
    for(i=0;i<N;i++){
        x = rand()*2.0/RAND_MAX - 1;
        y = rand()*2.0/RAND_MAX - 1;
        if((x*x+y*y)<=1){
            m++;
        }
    }
    printf("%f\n",m*4.0/N);
    return 0;
}
```

1.3 执行结果：



```
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.141262
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.141726
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.141280
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.140994
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.140683
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.141489
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.140890
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.141632
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.142198
mtawaken@mtawaken-pc:~/tmp$ ./rect-oval-PI.out
3.140586
mtawaken@mtawaken-pc:~/tmp$ gnome-screenshot -w
```

方法 2：蒲丰投针

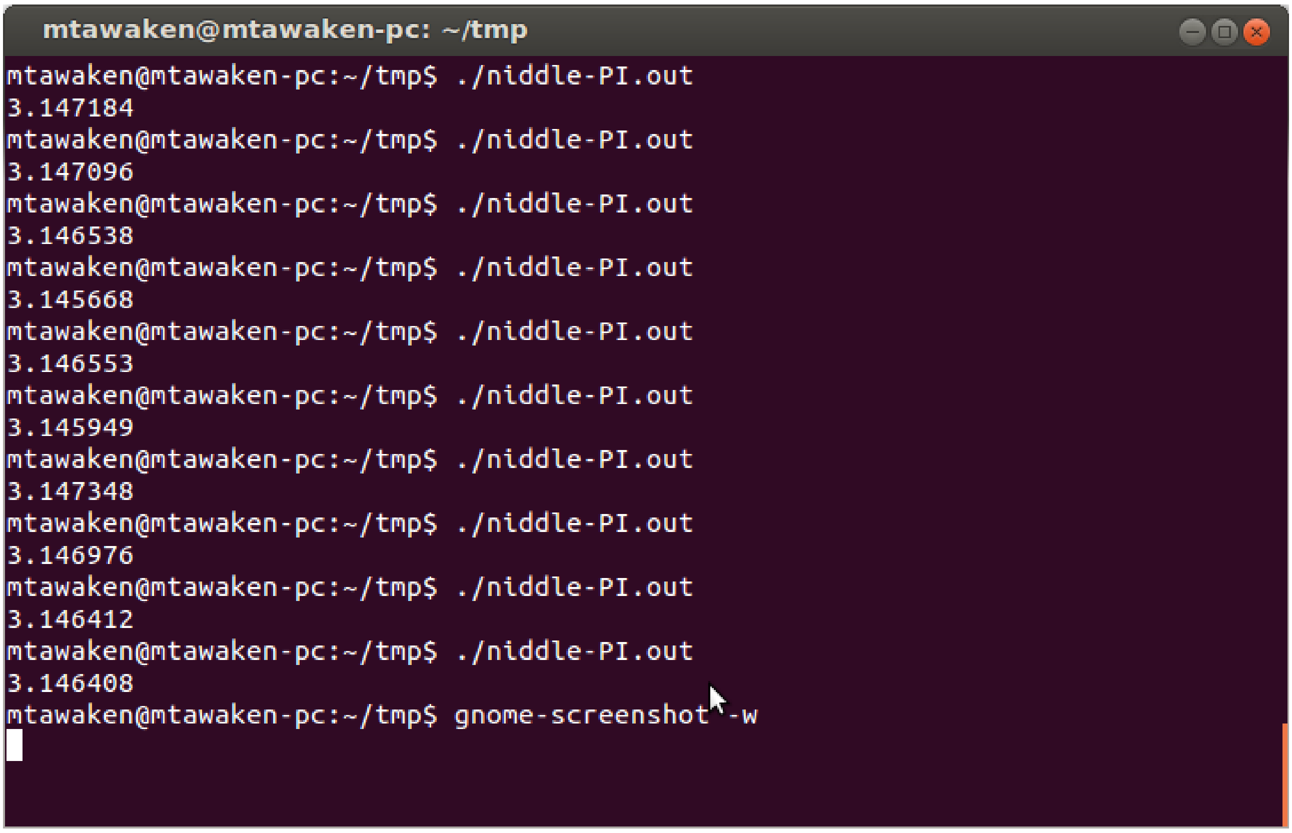
2.1 原理：

在等间距(不妨 d=1)的平行线上投针(l=0.85<d)。针在平行线上的概率 p=2*l/(PI*d)。从而求得 PI。模拟投针可以通过模拟针一头的落点和针与平行线的角度两个变量实现。

2.2 代码：

```
#define D 1
#define L 0.85
#define N 10000000
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int main(){
   srand(time(NULL));
   float by,ty,a;
   int m,i;
   m=0;
   for(i=0;i<N;i++){
      by = rand()*1.0/RAND_MAX + 1;
      a = rand()*180.0/RAND_MAX;
      ty = by - L*sin(a);
      if(((int)floor(by))!=((int)floor(ty))){
         m++;
      }
   }
   printf("%f\n",2.0*N*L/m);
   return 0;
}
```

2.3 执行结果：



```
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.147184
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.147096
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.146538
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.145668
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.146553
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.145949
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.147348
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.146976
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.146412
mtawaken@mtawaken-pc:~/tmp$ ./niddle-PI.out
3.146408
mtawaken@mtawaken-pc:~/tmp$ gnome-screenshot -w
```

方法 3：随机数的互素概率

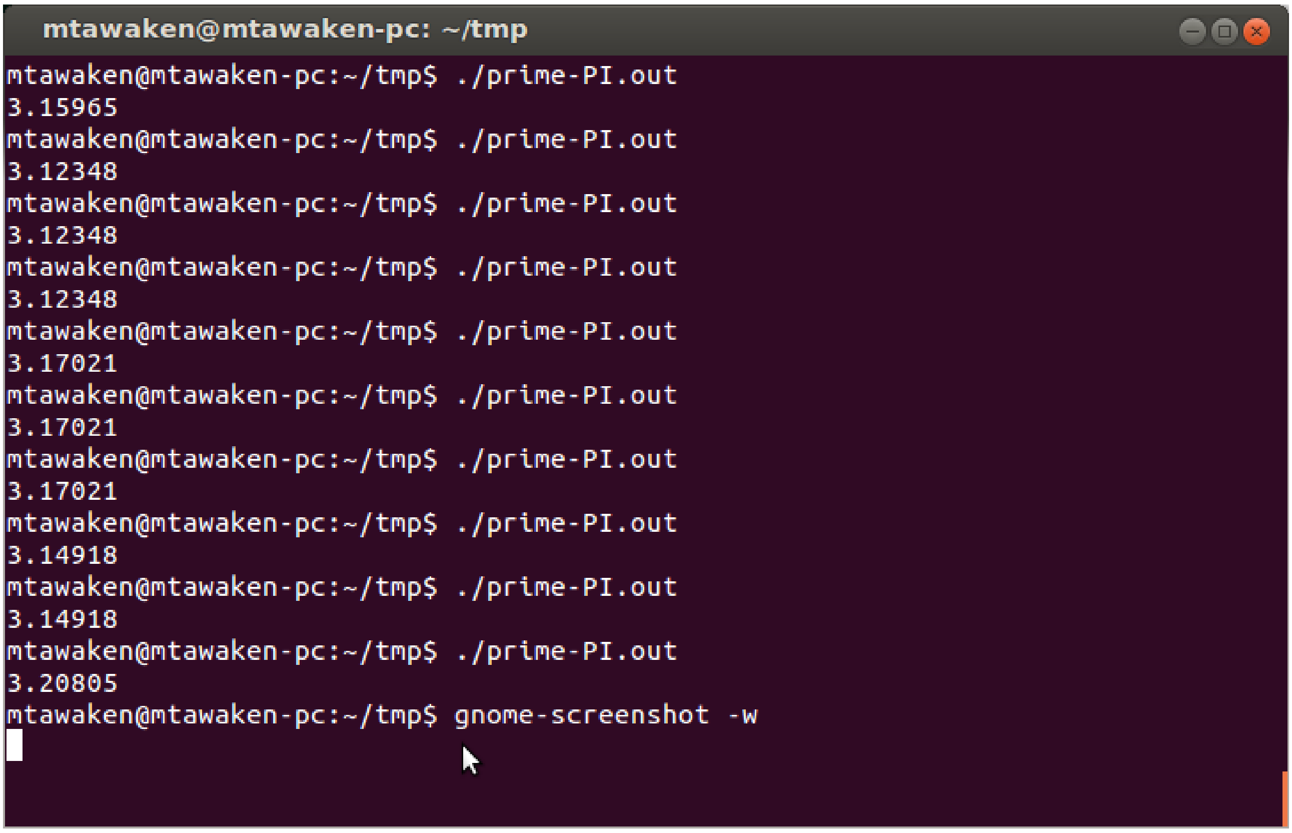
3.1 原理：

在给定范围 N 内的随机数对互素的概率接近于 p=1/(1+ 1/ 2^2 +1/ 3^2 + ...)=6/PI*PI。

3.2 代码：

```
#include<iostream>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#define N 1000
using namespace std;
int gcd(int a,int b){
   int r;
   while(b>0){
      r=a%b;
      a=b;
      b=r;
   }
   return a;
}
int main(){
   srand(time(NULL));
   int n = 1000;
   int m = 0;
   float p;
   int a,b,numofgcd;
   for(int i=0;i<n;i++){
        a = (int)((rand()*1.0*N)/RAND_MAX);
        b = (int)((rand()*1.0*N)/RAND_MAX);
        numofgcd = gcd(a,b);
        if(numofgcd==1)
            m++;
    }
    cout<<sqrt(6.0*n/m)<<endl;
    return 0;
}
```

3.3 运行结果：



```
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.15965
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.12348
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.12348
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.12348
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.17021
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.17021
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.17021
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.14918
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.14918
mtawaken@mtawaken-pc:~/tmp$ ./prime-PI.out
3.20805
mtawaken@mtawaken-pc:~/tmp$ gnome-screenshot -w
```