前言
成为高手的两种方式：

1. 站在巨人的肩膀上，直接领悟代码或者维护他们的代码
2. 自省(暗时间里面说 自省是很最重要的品质)

==和=：

作者提到像 if(i == 3)容易出现 if(i = 3)情况。其实如果能写出 if(3 == i)代码的人都是有意识的结果，但是在比较两个变量的时候不起作用，所以人的意识很重要，时刻清楚自己要的是比较还是赋值。

随后作者提出了一个 Bug。= 写成了==，则赋值变成了比较，然后结果就被丢弃，导致赋值不成功。

tunfs 命令：

在 tunfs 在线手册中有提示 Bugs 说明该程序只能运行在未安装好的文件系统中。但是在一些公司移植时删除了 Bugs，让人很容易陷入陷阱。当然现在不可能存在这种情况，但是我们在使用程序的时候还是先看说明文档比较好。不然，它会坑你的。
经典语句： You can tune a file system but you can't tune a fish.

编程挑战--计算机日期
关于 time_t 定义和时间问题
1. time_t 的定义
2. time_t 的最大值
3. time_t 是否溢出，什么时候溢出
4. 解决溢出的方案

1 和 2 问题：
在不同的系统上 time_t 定义有差异。我们假设是未修改过的 long，那么 time_t 的最大值就很好算了。下面代码计算最大值：

```
#include <stdio.h>
#include <time.h>

int main()
{
    time_t biggest = 0x7FFFFFFF;
    printf("biggest = %s\n",ctime(&biggest));

    return 0;

}
```

结果(结果和书上不相同，后面会解释)

```
biggest = Tue Jan 19 11:14:07 2038
```

问题 3：

明显我们观察到 2038 年离我们不远了。而且前面的程序输出不正确。为什么呢，因为 ctime 函数只能输出当地时间，我和作者的时区不一样，所以输出不一样。gmtime 函数可以转换成 GMT 时间，而 asctime 可以转换成字符串，这样程序该这么写：

```
#include <stdio.h>
#include <time.h>
int main()
{
    time_t biggest = 0x7FFFFFFF;
    printf("biggest = %s\n",asctime(gmtime(&biggest)));

    return 0;
}
```

结果：

```
biggest = Tue Jan 19 03:14:07 2038
```

在 APUE 第一章的习题 1-5 也是这个问题。可以想象时间日期的处理很有难度。

问题 4：

Linux 系统起始时间为 1970.1.1 00：00：00

解决溢出问题可以使用更长的位数来存储时间。

系统一般定义 time_t 为 time64_t

顺便提一下，time()函数还有一个用途，就是当随机函数种子

```
srand(unsigned)time(NULL);
```