标准I/O（一）

主讲：大海老师

课程目标

文件的打开（熟练）

文件的关闭（熟练）

小结

标准I/O – 打开文件

下列函数可用于打开一个标准I/O流：

FILE *fopen (const char *path, const char *mode);

成功时返回流指针；出错时返回NULL

path mode 接判

助记：不同：
函名最直相联参数，再相关？接？判？

标准I/O – fopen – mode参数

mode参数：

“r” 或 “rb”	以只读方式打开文件，文件必须存在。
“r+” 或 ” r+b”	以读写方式打开文件，文件必须存在。
“w” 或 “wb”	以只写方式打开文件，若文件存在则文件长度清为0。若文件不存在则创建。
“w+” 或 “w+b”	以读写方式打开文件，其他同” w” 。
“a” 或 “ab”	以只写方式打开文件，若文件不存在则创建；向文件写入的数据被追加到文件末尾。
“a+” 或 “a+b”	以读写方式打开文件。其他同” a”

标准I/O – fopen – mode参数

打开一个标准I/O流的六种不同的方式

限　制	r	w	a	r+	w+	a+
文件必须已存在	•			•		
擦除文件以前的内容		•			•	
流可以读	•			•	•	•
流可以写		•	•	•	•	•
流只可在尾端处写			•			•

标准I/O – fopen – 示例

```
#include <stdio.h>
int main(int argc,  char *argv[])
{
   FILE *fp;
   if ((fp = fopen( “test.txt” , “r+” )) ==
    NULL) {
       printf( “fopen  error\n” );
       return -1;
   }
   ……
   return  0;
 }
```

标准I/O – fopen – 新建文件权限

fopen() 创建的文件访问权限是0666(rw-rw-rw-)

Linux系统中umask设定会影响文件的访问权限，其规则为(0666 & ~umask)

Root用户是 022 普通用户是002

用户可以通过umask函数或者命令修改相关设定

如果希望umask不影响文件访问权限，该如何设定?

标准I/O – 处理错误信息

```
extern int  errno;
void perror(const char *s);
char *strerror(int errno);
```

errno 存放错误号，由系统生成

perror先输出字符串s，再输出错误号对应的错误信息

strerror根据错误号返回对应的错误信息

标准I/O – 错误信息处理 – 示例1

```
#include <stdio.h>

int main(int argc,  char *argv[])
{
   FILE *fp;

   if ((fp = fopen( “test.txt” ,  “r+” )) == NULL) {
       perror( “fopen” );
       return -1;
   }
   ……
```

fopen: No such file or directory

标准I/O – 错误信息处理 – 示例2

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
int main(int argc,  char *argv[])
{
   FILE *fp;
   if ((fp = fopen( “test.txt” ,  “r+” )) == NULL) {
       printf( “fopen: %s\n” , strerror(errno));
       return -1;
   }
   ……
```

fopen: No such file or directory

标准I/O – 关闭文件

```
int fclose(FILE *stream);
```

- fclose()调用成功返回0，失败返回EOF，并设置errno
- 流关闭时自动刷新缓冲中的数据并释放缓冲区
- 当一个程序正常终止时，所有打开的流都会被关闭。
- 流一旦关闭后就不能执行任何操作

标准I/O – 小结

fopen

fclose

标准I/O – 思考和练习

程序中能够打开的文件或流的个数有限制，如何测试?

思路：循环打开流，成功则计数器累加，直到出错为止

答案：1021 + stdin + stdout + stderr = 1024