文件类型

- 普通文件：存放在磁盘里面的文件
- 目录文件：存放在磁盘里面，其他文件的位置信息
- 链接：存放另一个文件的路径
- 设备文件：字符设备 块设备
- 管道
- socket

文件指针

ISO C 标准.

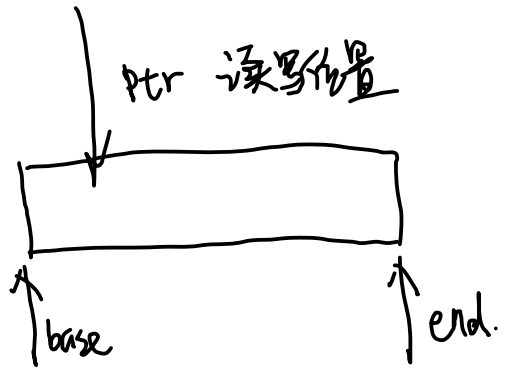
可移植性

FILE *

如何查看 FILE

1. 预处理

2. grep



打开关闭文件

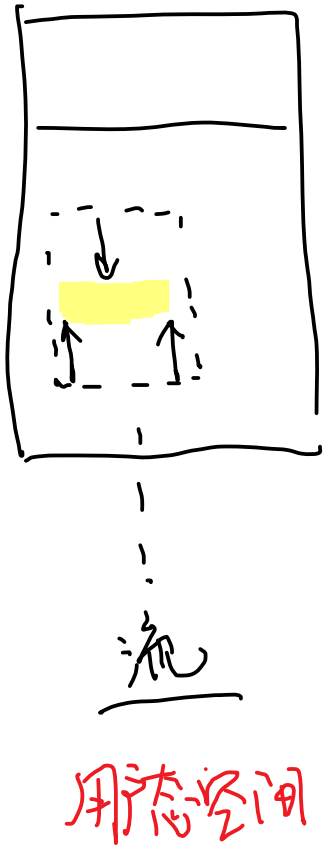
fopen `FILE *fopen(const char *pathname, const char *mode);`

路径　　模式

fclose

mode 的可选模式列表

模式	读	写	位置	截断原内容	创建
rb	Y	N	文件头	N	N
rb+	Y	Y	文件头	N	N
wb	N	Y	文件头	Y	Y
wb+	Y	Y	文件头	Y	Y
ab	N	Y	文件尾	N	Y
ab+	Y	Y	文件尾	N	Y



Linux内，文本模式与二进制模式 没有区别的

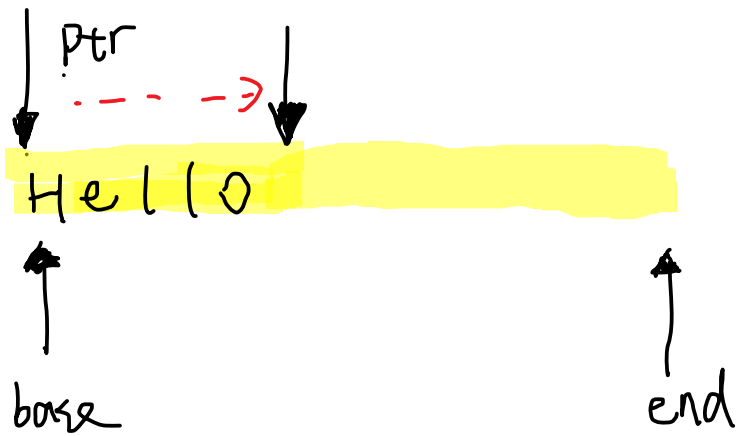
func.h

```c
#include <stdio.h>
#include <string.h>
#define ARGS_CHECK(argc,val) {if(argc != val) {printf("args error!\n"); return -1;}}
#define ERROR_CHECK(ret,val,info) {if(ret == val) {perror(info);return -1;}}
```

/usr/include/

追加模式

"a"　"a+"



第一次写入时，自动偏移到文件末尾

```c
#include <func.h>

int main(int argc, char *argv[])
{
    ARGS_CHECK(argc, 2);
    FILE *fp;
    fp = fopen(argv[1],"ab+");
    ERROR_CHECK(fp, NULL, "fopen");
    char buf[128] = {0};
    fread(buf,sizeof(char),sizeof(buf),fp);
    puts(buf);
    const char *p = "how are you";
    fwrite(p,sizeof(char),strlen(p),fp);
    fclose(fp);
    return 0;
}

```

```
long ftell(FILE *stream);
```

```c
#include <func.h>

int main(int argc, char *argv[])
{
    ARGS_CHECK(argc, 2);
    FILE *fp;
    fp = fopen(argv[1],"ab+");
    ERROR_CHECK(fp, NULL, "fopen");
    long ret;
    char buf[128] = {0};
    ret = ftell(fp);
    printf("ptr pos = %ld\n", ret);
    fread(buf,sizeof(char),sizeof(buf),fp);
    puts(buf);
    ret = ftell(fp);
    printf("ptr pos = %ld\n", ret);
    const char *p = "how are you";
    fwrite(p,sizeof(char),strlen(p),fp);
    ret = ftell(fp);
    printf("ptr pos = %ld\n", ret);
    fclose(fp);
    return 0;
}
```

```
int chmod(const char *pathname, mode_t mode);
```

```
#include <func.h>
```

```
int main(int argc, char *argv[])
{
    ARGS_CHECK(argc, 3);
    //chmod 0777 file1
    int mode;
    sscanf(argv[1], "%o", &mode);//将命令行参数的字符串，转换成八进制整型
    int ret = chmod(argv[2],mode);
    ERROR_CHECK(ret, -1, "chmod");
    return 0;
}
```

宁度优先

传入参数 vs 传入传出参数

```
int chmod(const char *pathname, mode_t mode);
```

传入参数 不修改

```
char *getcwd(char *buf, size_t size);
```

传入传出参数 修改

```
char *getcwd(char *buf, size_t size);
```

```c
#include <func.h>

int main()
{
    //char buf[128] = {0};
    //char *ret = getcwd(buf,sizeof(buf));
    //ERROR_CHECK(ret, NULL, "getcwd");
    //puts(buf);
    printf("%s\n",getcwd(NULL,0));
}
```

推荐使用

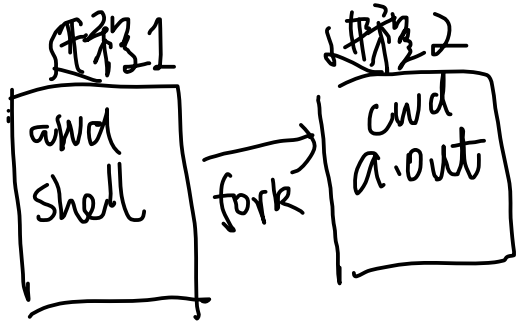
```
int chdir(const char *path);
```

```c
#include <func.h>

int main(int argc, char *argv[])
{
    ARGS_CHECK(argc, 2);
    printf("before cd ,%s\n",getcwd(NULL,0));
    int ret = chdir(argv[1]);
    ERROR_CHECK(ret, -1, "chdir");
    printf("after cd ,%s\n",getcwd(NULL,0));
}
```

```
[liao@ubuntu ~/test/file/chmod]$ ./a.out dir1
before cd ,/home/liao/test/file/chmod
after cd ,/home/liao/test/file/chmod/dir1
[liao@ubuntu ~/test/file/chmod]$
```

Shell的 cwd 没有变.



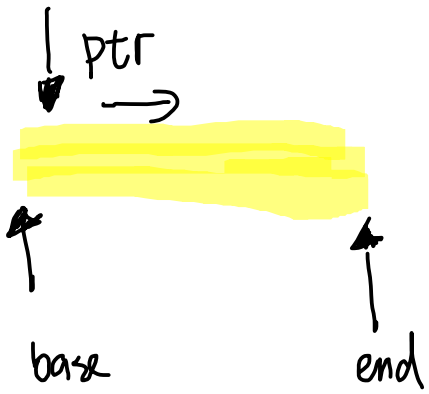
```
int mkdir(const char *pathname, mode_t mode);

int main(int argc, char *argv[])
{
    ARGS_CHECK(argc, 2);
    int ret = mkdir(argv[1],0777);//创建的时候，权限会受到掩码的影响
    ERROR_CHECK(ret, -1, "mkdir");
    return 0;
}
```

```
int rmdir(const char *pathname);

int main(int argc, char *argv[])
{
    ARGS_CHECK(argc, 2);
    int ret = rmdir(argv[1]);//待删除的目录，必须是空的
    ERROR_CHECK(ret, -1, "rmdir");
    return 0;
}
```

目录流



文件流