一.数据结构总结

1.掌握线性结构(重点)

线性表：顺序表 链表(重点)

栈和队列

2.熟悉非线性结构

树：二叉树(构造二叉树 遍历)

遍历方法：

递归遍历：先序 中序 后序

非递归遍历：层次遍历

最优二叉树：

扩展：红黑树 平衡二叉树

图：图的遍历(深度、广度)

拓扑排序

3.算法

查找算法：顺序 二分查找、hash查找(重点) 分块查找

排序算法：选择排序 冒泡排序(写代码 时间复杂度) 快速排序 堆排序 桶排序 shell 归并....

二.IO 进线程

1.IO(2天)

标准IO: c标准定义的一堆用于输入输出的函数，例如：printf/scanf；

文件IO:系统提供的一堆用于输入输出的函数(系统调用)；

1）Linux文件类型（bcd-lsp）

b(block): 块设备文件(例如:光盘 硬盘...)

c：字符设备文件(/dev/video0 /dev/ttyUSB0)

d:目录文件

-：普通文件

ASCII文本文件：*.c *.h *.cpp *.s *.txt .....

二进制文件：a.out（elf）

l:符号链接文件(实质是文件的地址，类似于Windows的快捷方式)

s(socket):套接字文件(主要用于进程间的通信，既可以用于同一台主机上的进程间通信，也可以用于不同主机间的进程间通信)

p(pipe):管道文件(主要用于进程间的通信，只能用于同一台主机上的进程间通信)

注意：

不同操作系统下的文件类型是不兼容的，比如windows下没有p文件

2)库函数和系统调用

库函数：c标准定义的函数,例如：fopen/fclose/printf/scanf....

系统调用：操作系统定义的函数,例如：open/close/read/write....

3）程序的运行方式

第一种：无os编程(裸机编程)

程序直接通过寄存器操作硬件，例如：c51/stm32编程

第二种：有os编程(linux编程)

程序运行在操作系统，通过操作系统操作硬件

4)库函数与系统调用的特点

1.系统调用

①用户空间进程访问内核的接口

②把用户从底层的硬件编程中解放出来

③极大的提高了系统的安全性

④使用户程序具有可移植性

2.库函数

①库函数为了实现某个功能而封装起来的API集合

②提供统一的编程接口，更加便于应用程序的移植

4)文件操作

标准IO：使用标准c库定义的一堆输入输出函数操作文件;

fopen()/fclose()

fgetc()/fputc() //一个一个读写

fgets()/fputs() //按行读写

fread()/fwrite()//按块读写

```
FILE *fp = fopen("1.c");//打开文件 fopen->open
fread(fp)
FILE *:流指针  操作文件
```

文件IO：使用系统调用函数操作文件

```
int fd = open("1.c");//打开文件
fd:文件描述符
```

注意:

当终端打开之后，会自动打开三个文件：标准输入 标准输出 标准错误输出

stdin stdout stderr

0 1 2

流：所有的I/O操作仅是简单的从程序移进或者移出，这种字节流，就称为流。

文本流：在流中处理的数据是以字符出现

二进制流：在流中处理的数据是以二进制出现

注意：

a. 不同的操作系统中对文本流和二进制流的定义是不一样的 ；

linux下对换行符 '\n' --> '\n'

windows下对换行符 '\n' --> '\r''\n'

b.linux下不区分文本流和二进制，实质就是二进制流

5)标准IO缓冲类型

全缓存：当fopen打开文件时默认是全缓存

读：当缓存区空时进行系统调用

写：当缓存区满时进行系统调用

```
int main()
{
    printf("hello world");

    fflush(stdout);

    _exit(0);
}
```

行缓存：当缓存区中遇到换行符时进行系统调用

例如：stdout stdin是行缓存

不缓存：stderr是不缓存

6)标准IO操作函数

```
1.fopen函数
    头文件：
        #include <stdio.h>
    函数原型：
        FILE *fopen(const char *pathname, const char *mode);
    函数参数：
        pathname:文件路径首地址
        mode:文件打开方式
            r:只读方式打开存在的文件
            r+:读写方式打开存在的文件
            w:只写方式打开文件，若文件不存在则创建，存在则清空
            w+:读写方式打开文件，若文件不存在则创建，存在则清空
            a:只写并追加方式打开文件，文件不存在则创建
            a+:读写并追加方式打开文件，文件不存在则创建
```

函数功能：
打开文件
返回值：
成功：流指针
失败:NULL,并设置errno的值，使用perror/strerror函数打印errno对应的错误信息
eg:

```
FILE *fp = fopen("test.c","r");
if(NULL == fp)
{
    perror("fopen");
    return -1;
}
```

说明：
errno: 错误号 定义在头文件#include <errno.h>中 int errno;
对应的错误信息通过函数perror/strerror查看

2.fgets函数
头文件：

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

函数原型：

```
char *fgets(char *s, int size, FILE *stream);
```

函数参数：
s:buffer的首地址
size:要读取的内容大小
stream：要操作的流指针
函数功能：
读取一行文件内容
返回值：
成功：s
失败:NULL

eg:

```
char buf[N] = {0};
fgets(buf,sizeof(buf),stdin);
```

练习1：使用fgetc/fputc实现文件拷贝
作业1：使用fgets统计文件的行数
作业2:编程读写一个文件test.txt，每隔1秒向文件中写入一行数据，类似这样：

```
1,  2007-7-30 15:16:42
2,  2007-7-30 15:16:43
```

该程序应该无限循环，直到按Ctrl-C中断程序。
再次启动程序写文件时可以追加到原文件之后，并且序号能够接续上次的序号，比如：

```
1,  2007-7-30 15:16:42
2,  2007-7-30 15:16:43
3,  2007-7-30 15:19:02
4,  2007-7-30 15:19:03
5,  2007-7-30 15:19:04
```

分享：
1.图片打马赛克
2.文件加密/解密

2.进线程(4天)