1.IO进程课程特点

1. 用户空间和内核空间交互过程
2. IO进程课程逻辑思维不强，IO进程会学习100个函数接口（记忆）
3. 每天学习的接口通过xmind总结
4. 每个函数接口一定要多写几遍
5. DC23032-IO进程课程笔记:

   http://note.youdao.com/noteshare?id=599b325bbeed4f7a25e8408bac2d4262

2.IO进程课程大纲

1. IO知识

   标准IO

   文件IO
2. 目录操作
3. 库的制作及使用

   动态库

   静态库
4. 进程

   进程的相关概念，进程的创建，进程的调度方式，进程执行状态，

   进程间通信等
5. 线程

   线程的相关概念，创建，销毁，线程间通信等

   7.5天

3.最早接触的IO

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

在使用printf/scanf函数的时候需要包含stdio.h，它就是标准，输入，输出的头文件。

它就是标准IO函数的接口。

4.IO相关知识

4.1IO的种类

IO分为：标准IO，文件IO。

标准IO：它是库函数

文件IO：它是系统调用

4.2什么是库函数，什么是系统调用

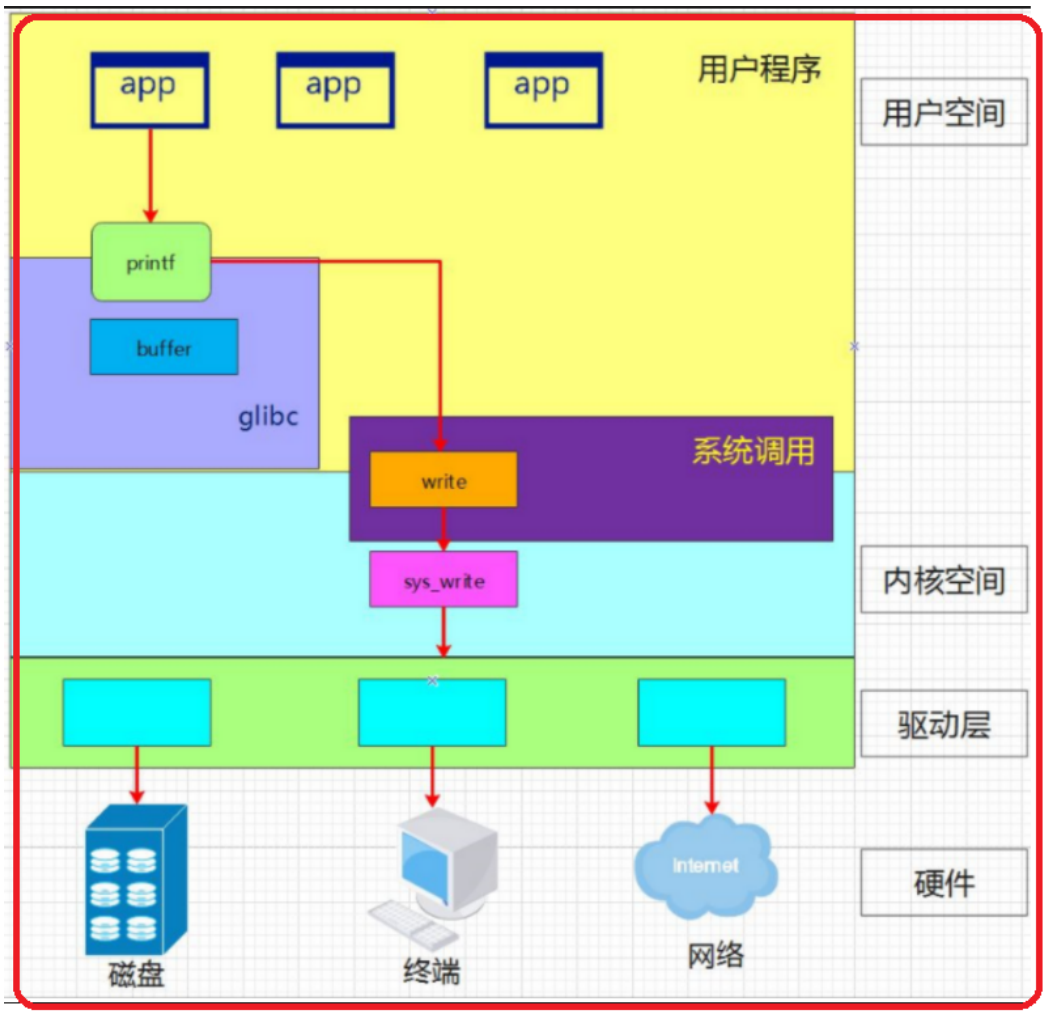
系统调用：从用户空间进入内核空间的一次过程就是一次系统调用，

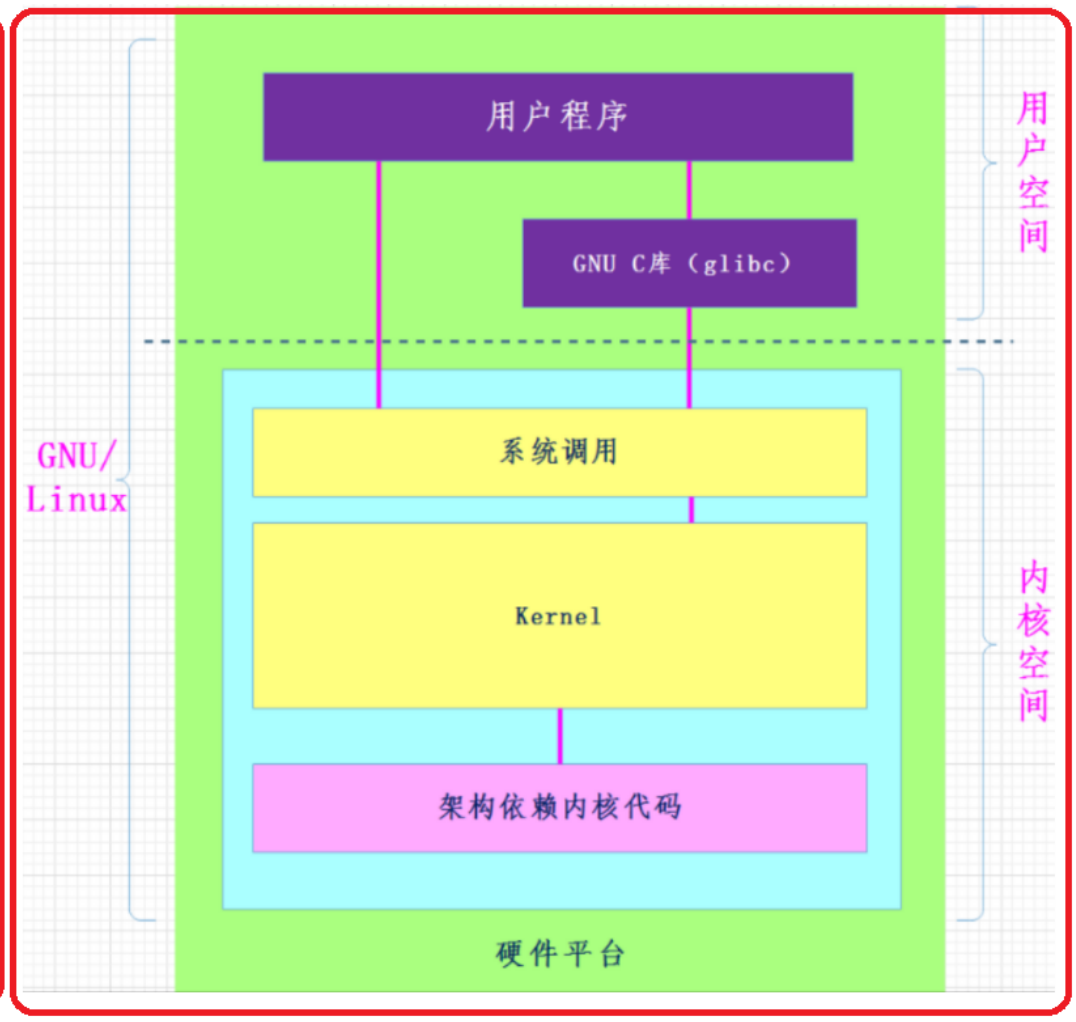
系统调用没有缓冲区，系统调用的效率比较低。系统调用的可移植性

比较差。

库函数：库函数 = 缓冲区 + 系统调用，库函数的效率比系统调用的高。

库函数的可移植性比较强。





4.3常用接口列举

标准IO:printf/scanf/fopen/fclose/fread/fwrite/fgetc/fputc/fgets/fputs...

文件IO:open/read/write/close...

5.标准IO

5.1FILE结构体

FILE是标准IO中的一个结构体，当使用fopen打开一个文件的时候，

返回值就是FILE *(文件指针)，这个指针指向一个FILE结构体，这个

结构中就记录了打开文件的各种信息（打开的方式，关联的系统调用，

光标等等）。以后在对这个文件进行读写的时候否是通过这个FILE *

指针完成的。在一个正在执行的程序默认有三个文件指针已经可以使用

了stdin,stdout,stderr。

类型	功能
stdin	标准输入
stdout	标准输出
stderr	标准出错

```
typedef struct _IO_FILE FILE;

struct _IO_FILE {
    int _flags;         //打开文件的方式
    char* _IO_buf_base; //标准IO缓冲区的起始地址
    char* _IO_buf_end;  //标准IO缓冲区的结束地址
    int _fileno;        //和系统调用相关的成员
    ...
}
```

5.2fopen打开文件的函数

5.2.1fopen函数功能

```
1   可执行程序或 shell 命令
2   系统调用(内核提供的函数)
3   库调用(程序库中的函数)
4
man 1 ls  :查看ls命令的man手册
man 2 open:查看open系统调用的man手册
man 3 fopen:查看fopen库函数的man手册
```

```c
#include <stdio.h>

FILE *fopen(const char *pathname, const char *mode);
功能：使用标准IO打开文件
参数：
    @pathname:打开文件的路径及名字  "/home/linux/hello.txt"  argv[1]
    @mode:打开文件的方式 "r" "r+"   "w" "w+"
        r       :以只读方式打开文件，将光标定位到文件的开头
        r+      :以读写方式打开文件，将光标定位到文件的开头
        w       :以写的方式打开文件，如果文件不存在就会创建文件，如果文件
                 存在就清空文件，将光标定位到文件的开头
        w+      :以读写的方式打开文件，如果文件不存在就会创建文件，如果文件
                 存在就清空文件，将光标定位到文件的开头
        a       :以追加的方式打开文件，文件不存在会创建文件，将光标定位到文件的结尾
        a+      :以读和追加的方式打开文件，文件不存在会创建文件，
                 如果上来就读会将光标定位在文件的开头，如果写光标定位到文件的结尾。
返回值：成功返回FILE *(文件指针)，失败返回NULL（置位错误码）
```

5.2.2fopen函数使用实例

```c
#include <stdio.h>

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

    // 1.以只写的方式打开文件，不存在创建，存在清空
    // if((fp = fopen("./hello.txt","w"))==NULL){
    //     printf("fopen error\n");
    //     return -1;
    // }

    // 2.以只读方式打开文件，文件不存在会报错，文件存在以只读方式打开
    if((fp = fopen("./hello.txt","r"))==NULL){
        printf("fopen read error\n");
        return -1;
    }
    return 0;
}
```

5.3fclose关闭文件函数

5.3.1fclose函数功能

```c
#include <stdio.h>

int fclose(FILE *stream);
功能：光标文件
参数：
    @steam:文件指针    #define EOF (-1)
返回值：成功返回0，失败返回EOF(置为错误码)
```

5.3.2fclose函数实例

```c
#include <stdio.h>

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

    // 以只读方式打开文件，文件不存在会报错，文件存在以只读方式打开
    if((fp = fopen("./hello.txt","r"))==NULL){
        printf("fopen read error\n");
        return -1;
    }

    // 关闭文件
    fclose(fp);
    return 0;
```

```
}
```

5.3fgetc读字符函数

5.3.1fgetc函数功能

```
#include <stdio.h>

int fgetc(FILE *stream);
功能：从stream对应的文件中读取一个字符，将字符ascii返回，
    每读一个字符之后，光标会向后移动一个字节。
参数：
    @stream:文件指针
返回值：成功返回读取到字符的ascii,失败或者到文件的结尾返回EOF
```

5.3.2fgetc函数实例

```
abc

```

```c
#include <stdio.h>

int main(int argc, const char* argv[])
{
    FILE* fp;
    // 1.以只读方式打开文件
    if ((fp = fopen("./hello.txt", "r")) == NULL) {
        printf("fopen error\n");
        return -1;
    }
    // 2从文件中读字符
    char ch;
    // 从fp对应的文件中读取一个字符串，将读取到的字符
    // 赋值给ch变量，光标会向后移动一个字节
    ch = fgetc(fp);
    printf("ch = %c\n", ch);
    ch = fgetc(fp);
    printf("ch = %c\n", ch);
    ch = fgetc(fp);
    printf("ch = %c\n", ch);
    ch = fgetc(fp);
    printf("ch = %d\n", ch);
    ch = fgetc(fp);
    printf("ch = %d\n", ch);

    // 3.关闭文件
    fclose(fp);
    return 0;
}
```

```
linux@ubuntu:~/work/day1$ ./a.out
ch = a
ch = b
ch = c
ch = 10
ch = -1
```


5.3.3fgetc函数练习

练习：请编写一个统计文件行号的应用程序，要求文件名

通过命令行传递给程序。执行的方式如下：

./a.out filename

```c
#include <stdio.h>

// ./a.out filename
int main(int argc, const char* argv[])
{
    FILE* fp;
    int line = 0;
    // 1.校验命令行参数个数
    if (argc != 2) {
```

```
        printf("input error,try again\n");
        printf("usage: ./a.out filename\n");
        return -1;
    }

    // 2.以只读的方式打开文件
    if ((fp = fopen(argv[1], "r")) == NULL) {
        printf("fopen error\n");
        return -1;
    }

    // 3.循环从文件中读字符，判断是否是换行，
    // 如果是换行让line++;，当读取到文件的结尾EOF时退出循环
    char ch;
    while ((ch = fgetc(fp)) != EOF) {
        if (ch == '\n')
            line++;
    }

    // 4.打印行号
    printf("line = %d\n", line);

    // 5.关闭文件
    fclose(fp);
    return 0;
}
```

5.4fputc写字符函数

5.4.1fputc函数的功能

```
#include <stdio.h>

int fputc(int c, FILE *stream);
功能：将c对应的字符写入到stream对应的文件中，
    光标会自动向后移动一个字节
参数：
    @c:被写的字符
    @stream:文件
返回值：成功返回被写字符的ascii,失败返回EOF
```

5.4.2fputc函数的实例

```
#include <stdio.h>

int main(int argc, const char* argv[])
{
    FILE* fp;
    if ((fp = fopen("test.txt", "w")) == NULL) {
        printf("fopen error\n");
        return -1;
    }

    fputc('a',fp); //向文件中写入'a'字符
    fputc(98,fp);  //将'b'字符写入到文件中
    fputc('\xa',fp); //将'\n'写入到文件中
    // 10  '\n' '\xa' '\12'

    fclose(fp);
    return 0;
}
```

5.4.3fputc函数的练习

练习：使用fgetc/fputc实现文件的拷贝，执行的方式如下：

./a.out srcfile destfile

> 注：将srcfile文件拷贝到destfile中

```
#include <stdio.h>

// ./a.out srcfile destfile
int main(int argc, const char* argv[])
{
    FILE* fp1,*fp2;
    int line = 0;
    // 1.校验命令行参数个数
    if (argc != 3) {
        printf("input error,try again\n");
```

```
        printf("usage: ./a.out srcfile destfile\n");
        return -1;
    }

    // 2.以只读的方式打开源文件，以只写方式打开目标文件
    if ((fp1 = fopen(argv[1], "r")) == NULL) {
        printf("fopen src error\n");
        return -1;
    }
    if ((fp2 = fopen(argv[2], "w")) == NULL) {
        printf("fopen dest error\n");
        return -1;
    }
    // 3.循环拷贝
    char ch;
    while ((ch = fgetc(fp1)) != EOF) {
        fputc(ch,fp2);
    }

    // 4.关闭文件
    fclose(fp1);
    fclose(fp2);
    return 0;
}
```

6.作业

1.使用fgetc完成从终端输入字符串的功能

要求：1支持输入空格（'\n'停止，满停止）,2对越界检查

```
#include <stdio.h>

char* mygets(char* arr, int n)
{
    int i = 0;
    char ch = 0;
    // 1.检查参数是否合法
    if ((arr == NULL) || (n <= 0)) {
        printf("%s 输入的参数无效\n", __func__);
        return NULL;
    }
    // 2.循环读取字符,循环退出条件i>=n-1,ch=='\n',ch==EOF
    while ((i < n - 1) && ((ch = fgetc(stdin)) != '\n') && (ch != EOF)) {
        arr[i++] = ch;
    }
    // 3.将字符串结束符放到输入中
    arr[i] = '\0';

    // 4.处理垃圾字符
    if ((i == n - 1) || (ch == EOF))
         while (((ch = fgetc(stdin)) != '\n') && (ch != EOF));

    return arr;
}

int main(int argc, const char* argv[])
{
    char arr[10] = { 0 };

    mygets(arr, 10);

    printf("arr = %s\n", arr);

    return 0;
}
```