



## 第二章 操作系统用户界面

### ◆ 主要任务

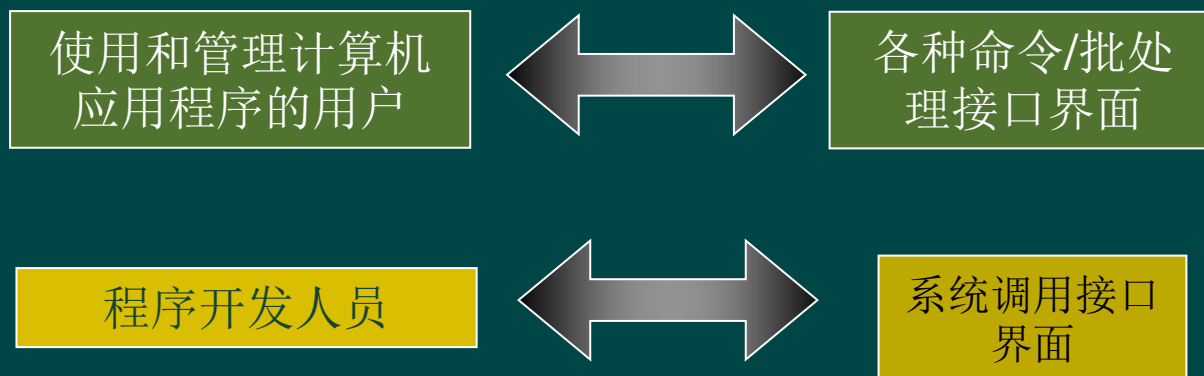
- ◆ 了解操作系统的用户工作环境；
- ◆ 掌握操作系统的两大用户界面；
- ◆ 理解运行一个用户程序的过程；
- ◆ 理解系统调用的含义及其实现过程；

### ◆ 难点

- ◆ 系统调用的含义及其实现过程

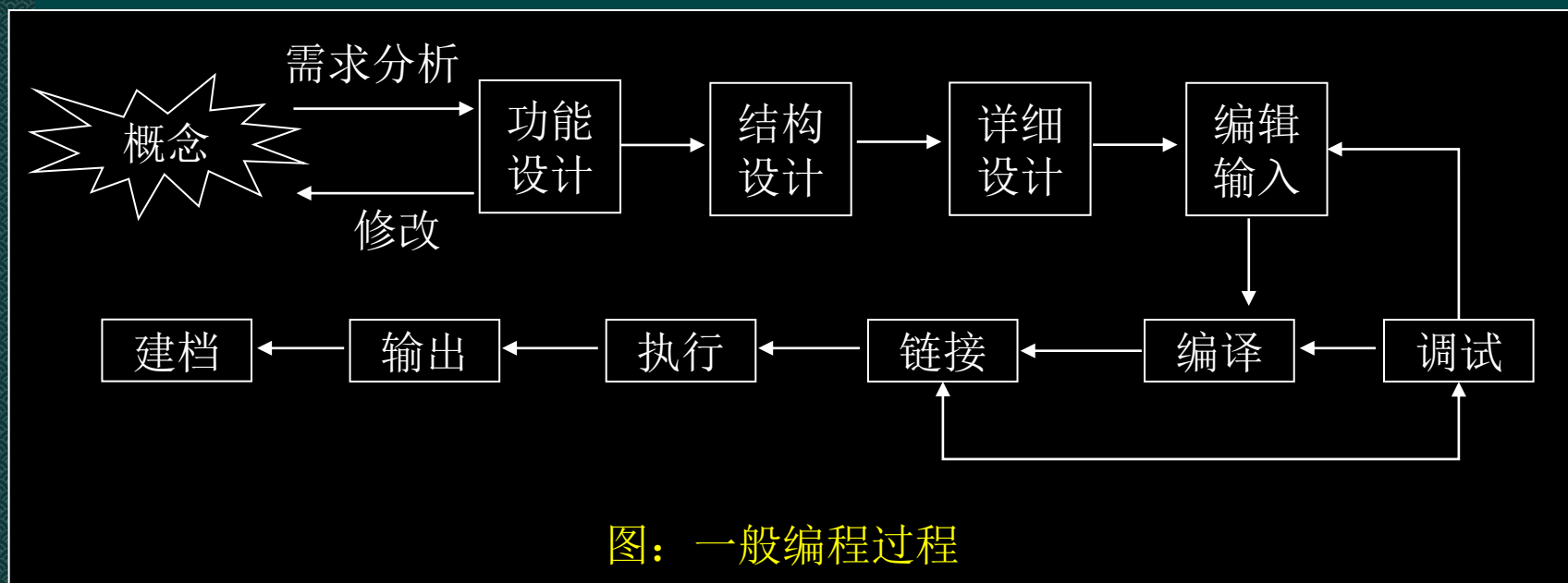
# 用户界面简介

- ◆ 用户界面是操作系统的重要组成部分。它负责用户和操作系统之间的交互。即用户通过用户界面向计算机系统提交服务需求，计算机通过用户界面向用户提供用户所需要的服务。
- ◆ 主要两类用户及接口界面：



## 2.2 一般用户的输入输出界面

### 2.2.1 作业的定义



- 作业： 1 项任务从向计算机提交到运行结束的全过程

## 2.2 一般用户的输入输出界面

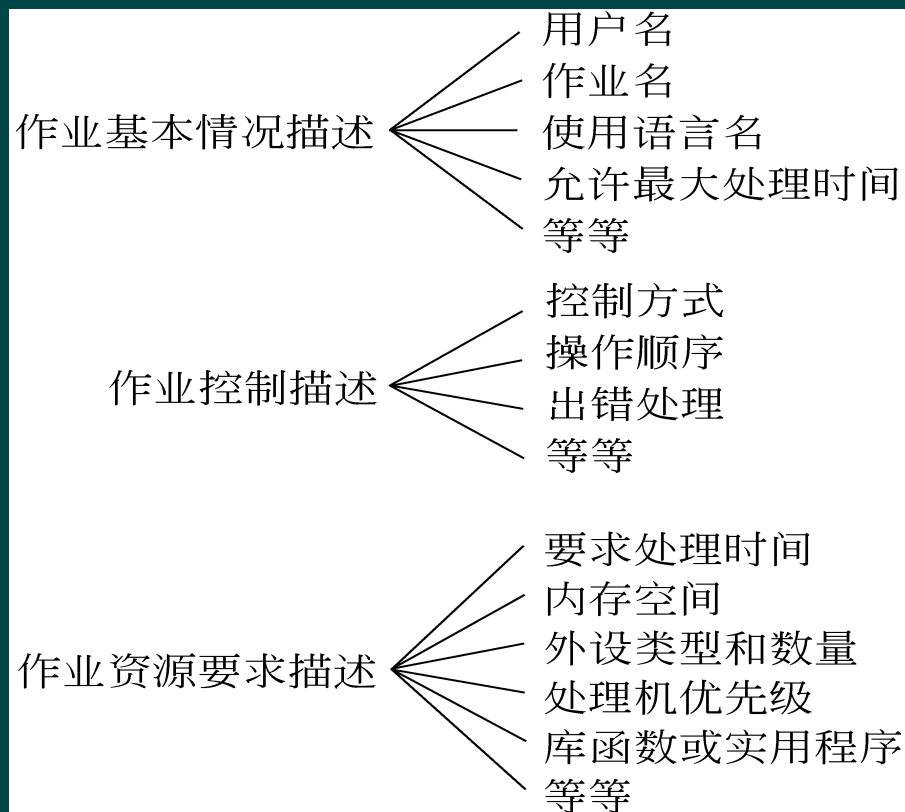
### ◇ 2.2.2 作业组织

◇ 程序

◇ 数据

◇ 作业说明书

作业由上3部分组成  
用在批处理和大型机  
中。微机和工作站没  
有作业的概念，除非  
使用批处理文件。



图：作业说明书的主要内容(以批处理.bat文件表达)



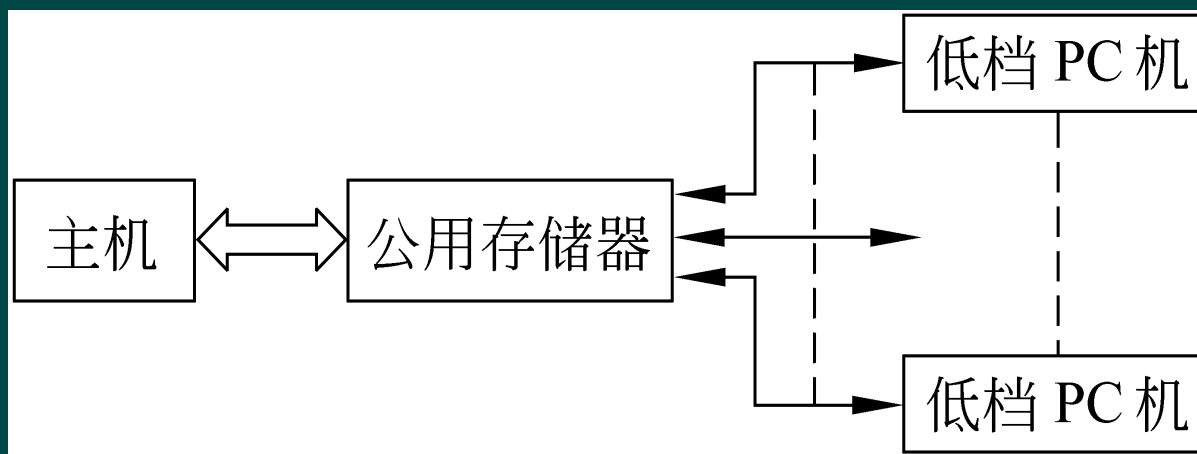
## 2.2 一般用户的输入输出界面

### 2.2.3 一般用户的输入输出方式

1. 联机输入方式

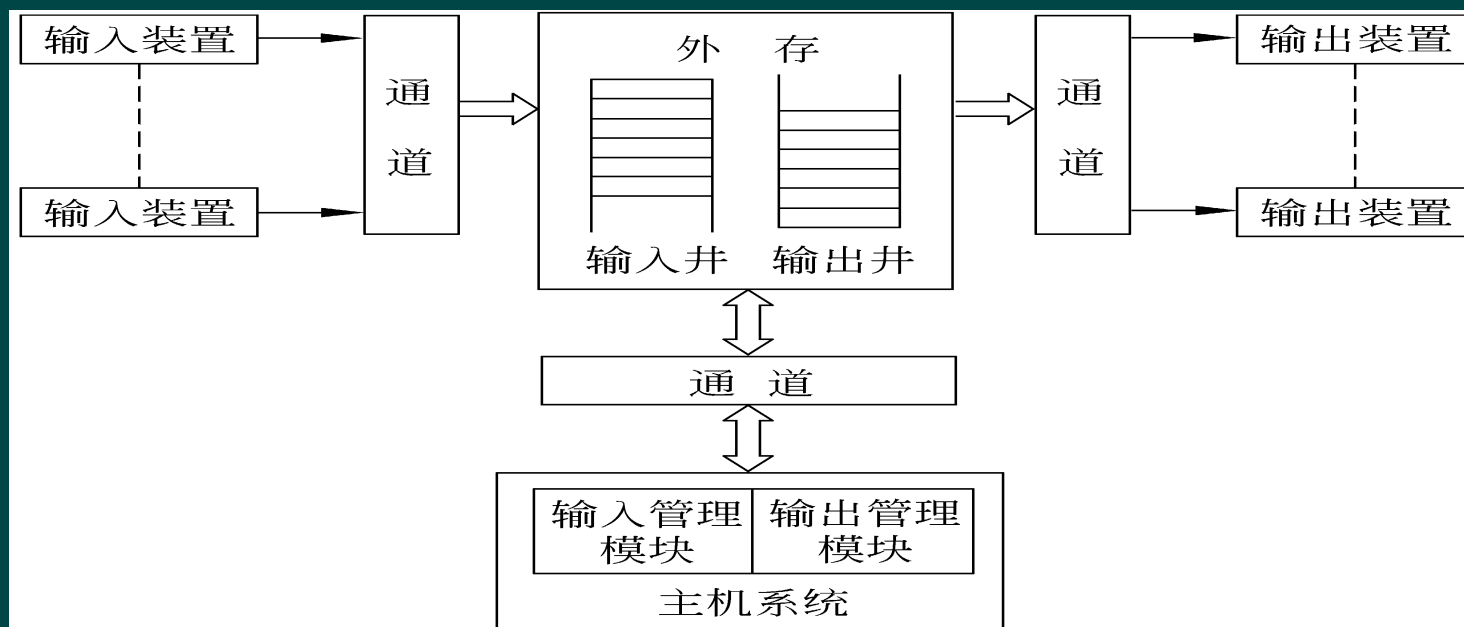
2. 脱机输入方式

3. 直接耦合方式



## 2.2一般用户的输入输出界面

### 4. SPOOLING系统（外围设备同时联机）



5. 网络联机方式: 网络上IO设备与网上的某台主机进行的输入/输出操作.

## 2.3 OS的用户界面种类

- ◆ 多数情况下，用户通过操作系统与机器硬件打交道，而不是直接操作机器硬件。操作系统提供的使用界面有三种形式：交互中端命令（行命令或可视化点击）、作业控制语言（如批处理BAT文件）、系统调用命令（如ASM、C语言、BASIC语言等写程序时使用的命令语句，用DEBUG调试时可直接使用系统调用）。

## 2.3 OS的用户界面种类

- ◆ **交互中断命令**：每个输入命令（不论是行敲的还是点击的）都被操作系统中的命令解释程序所接受，该程序分析收到的命令，然后调用系统中相应的程序（模块）执行。
- ◆ **作业控制语言**：这是批处理系统的主要界面形式。针对某一任务，用作业控制语言写一个作业说明书（即用户作业的处理步骤），连同任务程序、数据一起提交给机器运行。



## 2.3 OS的用户界面种类

- ◆ **系统调用命令**：操作系统为用户提供一组系统调用命令，用户将这些系统调用命令写在程序中，执行到这些命令时，将发生自愿性中断，进入操作系统转到相应的处理（程序）模块完成所要求的服务。
- ◆ **系统调用命令分为**：**文件/目录命令**（建立、打开、关闭、读写等）；**进程命令**（创建、杀死、跟踪子进程）；**进程通信命令**（发/接消息、发/收信号、发/收信件）；**资源调配命令**（申请/释放资源等）。

## 2.3 命令控制界面 (Linux)





## 2.3 命令控制界面 (Linux)

### ◆ Linux Shell是一种交互型命令解释程序

Shell程序由以下6部分组成:

- ◆ 命令或其他Shell程序
- ◆ 位置参数
- ◆ 变量及特殊字符
- ◆ 表达式比较
- ◆ 控制流语句
- ◆ 函数

```
mkdir backup
for file in `ls`
do
    cp $file backup/$file
    If [ $? -ne 0 ]; then
        echo "copying $file error"
    fi
done
```

## 2.3 命令控制界面（Windows）

### 2.3.2 Windows命令控制界面：

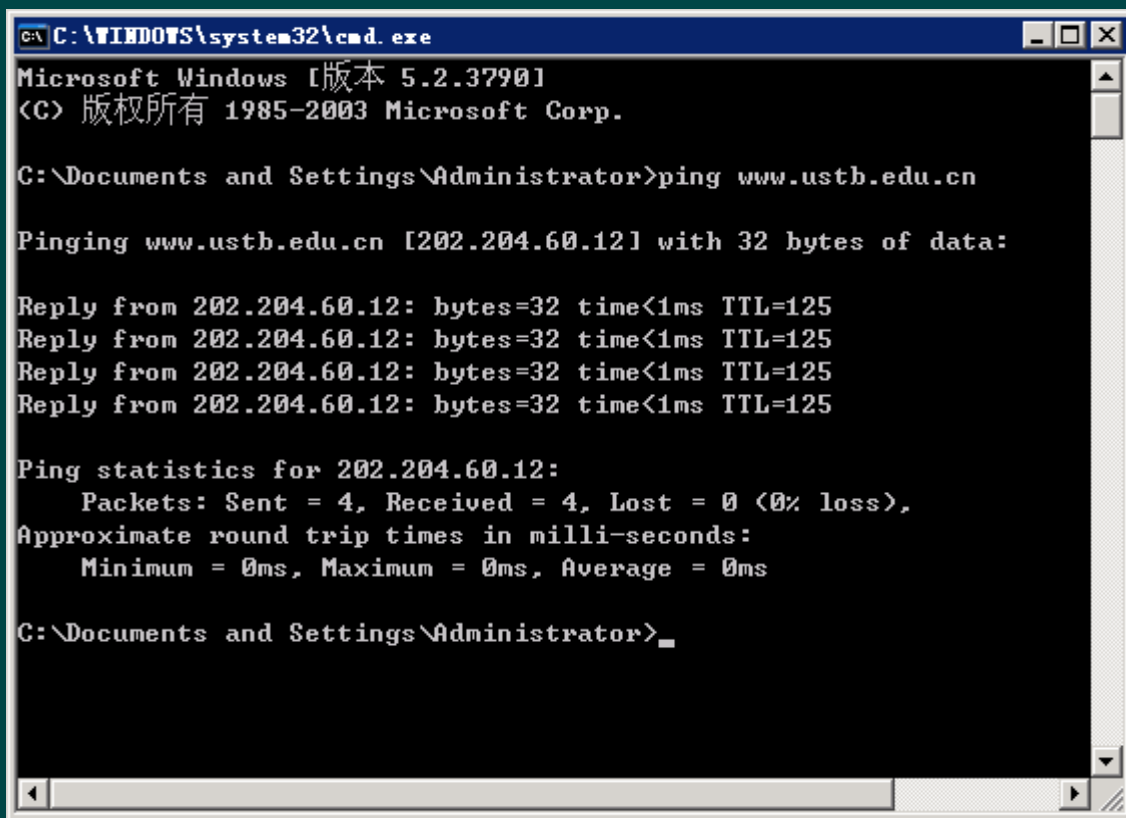
- ◇ 命令解释器（相当于Linux的Shell）
- ◇ 窗口交互部分



## 2.3 命令控制界面（Windows）

◆ Windows通过命令行解释器cmd为用户提供了强大的命令行控制界面，主要有4类命令：

- ◆ 系统信息命令
- ◆ 系统操作命令
- ◆ 文件系统命令
- ◆ 网络通信命令



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>ping www.ustb.edu.cn

Pinging www.ustb.edu.cn [202.204.60.12] with 32 bytes of data:

Reply from 202.204.60.12: bytes=32 time<1ms TTL=125
Reply from 202.204.60.12: bytes=32 time<1ms TTL=125
Reply from 202.204.60.12: bytes=32 time<1ms TTL=125
Reply from 202.204.60.12: bytes=32 time<1ms TTL=125

Ping statistics for 202.204.60.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\Administrator>
```

## 2.3 命令控制界面（Windows）



系统信息命令：time,date,mem,driverquery,systeminfo等

系统操作命令：shutdown, runas ,taskkill

文件系统命令：copy,del,mkdir

网络通信命令：ping,netstat,route

这些命令通过cmd.exe命令处理程序来执行。

## 2.3 命令控制界面 (Windows)



exam2.bat:

@ echo off

mem > %1/meminfo.txt

echo generate memoryinfo ok!

exam3.bat:

@ echo off

type %1\\*.txt

echo type ok!

@ echo off

mkdir test

call exam2.bat test

call exam3.bat test

echo call ok!

pause



## 2.4 系统调用

◆ **系统调用**是操作系统提供给编程人员的唯一接口。编程人员利用系统调用，在源程序一级动态请求和释放系统资源，调用系统中已有的系统功能来完成与机器硬件部分相关的工作以及控制程序的执行速度等。大致可以分为如下6类：

1. **设备管理**：请求和释放设备以及启动设备操作
2. **文件管理**：包括对文件的读、写、创建和删除
3. **进程管理**：进程的创建、执行、撤销等
4. **进程通信**：进程之间传递消息或信号
5. **存储管理**：获取作业占据内存区的地址等
6. **线程管理**：包括线程的创建、调度、执行、撤销等



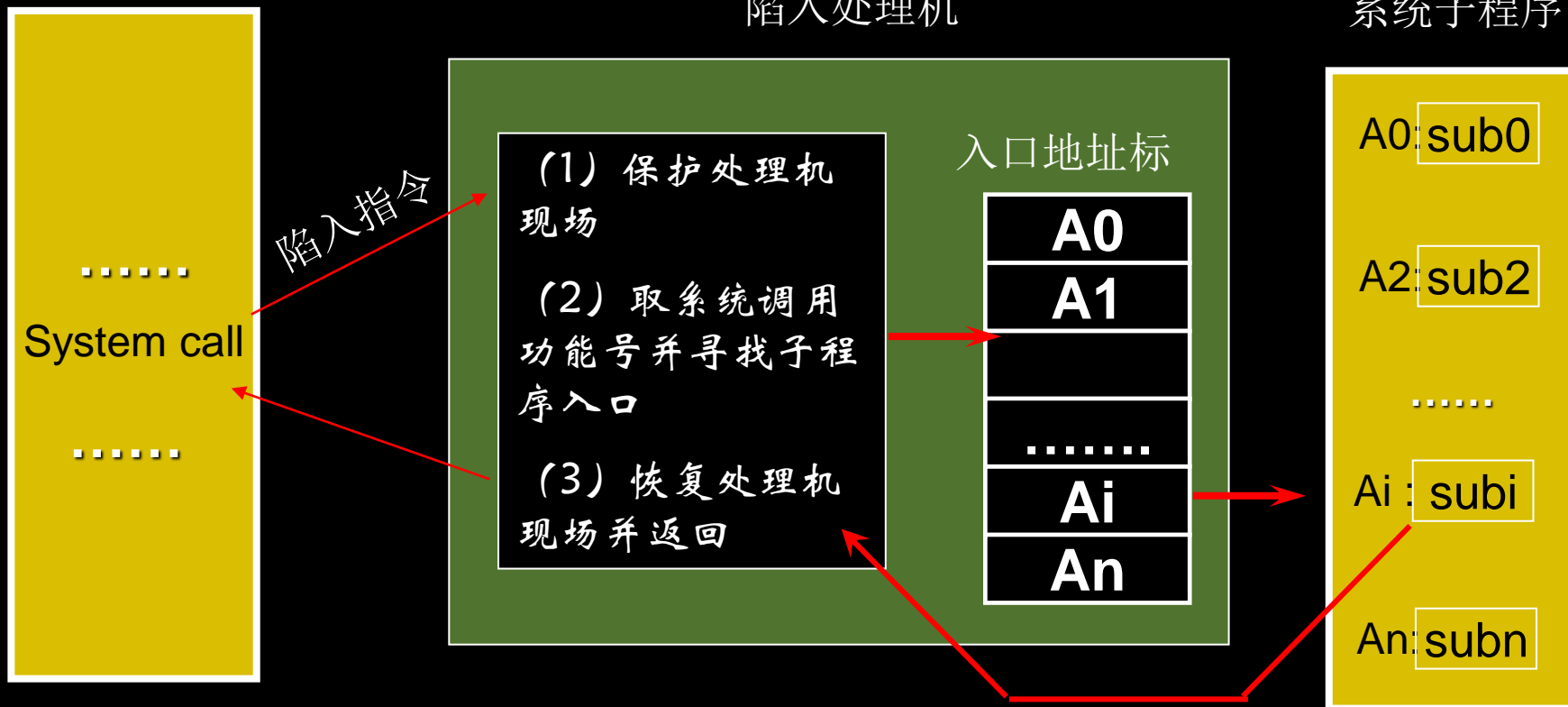
## 2.4 系统调用执行过程



用户程序

陷入处理机

系统子程序



## 2.4 系统调用执行过程

```
#include <fcntl.h>
#include <sys/stat.h>
#define SIZE 1
void filecopy(char * infile, char * Outfile)
{
    char Buffer[SIZE];
    int in_fh, Out_fh, Count;
    if((In_fh = open(Infile, O_RDONLY)) == -1)          /* 以只读模式打开输入文件 */
        printf("Opening infile");
    if((Out_fh = open(Outfile, (O_WRONLY|O_CREAT|O_TRUNC),
(S_IRUSR|S_IWUSR)) == -1)                          /* 以读写模式新建一个文件 */
        printf("Opening Outfile");
    while((Count = read(In_fh, Buffer, sizeof(Buffer))) > 0) /* 循环地向缓冲区读入输入文件内容 */
    if(write(out_fh, Buffer, Count) != Count)
        /* 将缓冲区读入的内容写到输出文件中去 */
        printf("Writing date");
    if(count == -1)
        printf("Reading date");
    close(In_fh);          /* 关闭输入文件 */
    close(Out_fh);         /* 关闭输出文件 */
}
```



## 2.5 Linux系统调用

◇ Linux提供的系统调用，功能大致如下：

- ◇ 设备管理的系统调用
- ◇ 文件系统操作的系统调用
- ◇ 进程控制的系统调用
- ◇ 存储管理的系统调用
- ◇ 管理用户系统的系统调用
- ◇ 通信的系统调用



## 2.5 Linux系统调用

```
int main()
{
    int fd; //打开文件描述符
    fd=open("System_call.txt",O_CREAT,0640); //打开文件系统调用
    if(fd==-1){
        printf("ERROR,Open File Failed!\n");
        exit(0);
    }
    write(fd,"This is System_call",
        sizeof("This is System_call")); //写入数据系统调用
    close(fd);
    return 0;
}
```



## 2.6 Windows系统调用

◇ Windows操作系统提供给程序员的编程界面称为应用编程接口API，常用的API函数大致如下：

- ◇ 窗口管理类
- ◇ 图形设备接口类
- ◇ 系统服务类
- ◇ 国际特性类
- ◇ 网络服务类

## 2.6 Windows系统调用



```
#include<windows.h>
int main()
{
    HANDLE hFile; //句柄
    DWORD num;
    hFile=CreateFile(
        "testAPI.txt",GENERIC_WRITE,
        0,NULL,
        OPEN_ALWAYS,FILE_ATTRIBUTE_NORMAL,
        NULL);    //建立文件testAPI
    WriteFile(
        hFile,"Hello API!",
        sizeof("Hello API!"),&num,
        NULL);    //往文件写入数据API
    CloseHandle(hFile);    //关闭文件API
    return 0;
}
```



# 习题

- 2.1 什么是作业、作业步？
- 2.2 作业由哪几部分组成？各有什么功能？
- 2.3 作业的输入方式有哪几种？各有何特点？
- 2.5 操作系统为用户提供哪些接口？它们的区别是什么？
- 2.7 什么是系统调用？系统调用与一般用户程序有什么区别？与库函数和实用程序又有什么区别？
- 2.8 简述系统调用的实现过程。
- 2.9 为什么分时系统没有作业的概念？
- 2.10 编写一个简单的SHELL程序，实现文件的复制功能。