

# 第 3 章 用户界面

月出皓兮 苏曙光老师的课堂笔记

2022 年 3 月 16 日

用户环境是指计算机用户工作的软件环境。

用户界面 (User Interface, UI) 是用户与操作系统内核进行交互和信息交换的媒介。其分为操作界面和系统调用。

操作界面可分为 3 类: 操作命令、批处理、图形用户界面。

系统调用是操作系统内核为应用程序提供的一系列服务/函数, 是提供给程序员在编程时使用的接口, 是用户程序取得操作系统服务的唯一途径。

请注意本章节各个名词间的逻辑关系。

## 目录

<b>1 用户环境</b>	<b>3</b>
<b>2 用户界面</b>	<b>3</b>
<b>3 操作界面</b>	<b>3</b>
<b>4 操作命令</b>	<b>3</b>
4.1 DOS 操作命令 . . . . .	4
4.2 LINUX 的典型命令 . . . . .	4
4.3 重定向命令 . . . . .	4
4.4 管道命令 . . . . .	5
<b>5 批处理命令</b>	<b>5</b>
5.1 批处理 (Windows): BAT/PowerShell . . . . .	5
5.2 Shell 脚本: Shell Script . . . . .	6
5.2.1 Shell . . . . .	6

5.2.2	Shell 的主要类型 . . . . .	6
5.2.3	Shell 的主要功能 . . . . .	7
5.2.4	运行脚本程序的三个方法 . . . . .	8
5.2.5	变量与引用 . . . . .	8
5.2.6	键盘输入和屏幕输出 . . . . .	8
5.2.7	条件判断和分支语句 . . . . .	8
5.2.8	循环语句 . . . . .	9
<b>6</b>	<b>用户图形界面</b>	<b>9</b>
<b>7</b>	<b>系统调用</b>	<b>9</b>
7.1	系统调用的特点 . . . . .	9
7.2	系统调用的实现形式 . . . . .	10
7.2.1	DOS 系统调用 . . . . .	10
7.2.2	Linux 系统调用 . . . . .	10
7.2.3	隐式系统调用 . . . . .	11
7.3	系统调用的执行过程（中断） . . . . .	11
7.4	Linux 系统调用 . . . . .	11
7.4.1	Linux 系统调用的工作原理 . . . . .	11
7.4.2	Linux 系统调用的代码实现 . . . . .	12
7.4.3	系统调用函数的调用方法 . . . . .	12

## 1 用户环境

用户环境是指计算机用户工作的软件环境。其包括桌面环境和命令行环境。

用户环境构造是指按照用户要求和硬件特性安装和配置操作系统。

操作系统不仅需要提供重要的内核功能，也需要提供操作命令和界面以及系统用户手册。

## 2 用户界面

用户界面 (User Interface, UI) 是用户与操作系统内核进行交互和信息交换的媒介。其可分为操作界面，系统调用 (System Call，系统功能调用，程序界面)。

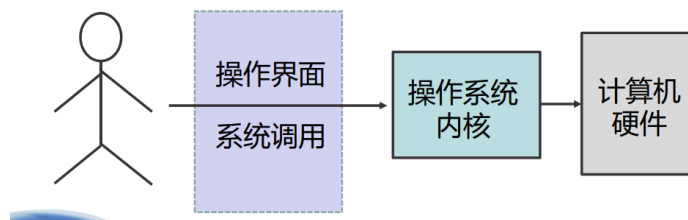


图 1: 用户、用户界面、操作系统内核与计算机硬件的关系

## 3 操作界面

操作界面可分为 3 类: 操作命令、批处理、图形用户界面。

## 4 操作命令

操作命令又称交互式命令。

其中，操作命令除了可以直接在命令行上输入使用，还有管道和重定向这两种特殊执行方式。

在 Linux、UNIX 系列操作系统中，操作命令在一个称为 Shell 的控制台环境下运行。

## 4.1 DOS 操作命令

1. 文件管理: COPY、COMP、TYPE、DEL、REN
2. 磁盘管理: FORMAT、CHKDSK、DISKCOPY、DISKCOMP
3. 目录管理: DIR、CD、MD、RD、TREE
4. 设备工作模式: LS、MODE
5. 日期、时间、系统设置: DATE、TIME、VER、VOL
6. 运行用户程序: MASM、LINK、DEBUG

## 4.2 LINUX 的典型命令

1. 用户管理: w、id、last、cut、crontab
2. 系统配置: chkconfig、rpm、uname、head、cat、hostname、lspci、lsusb、lsmod、env、ifconfig、iptables、route、netstat。
3. 进程管理: ps、top、kill、Ctrl+C
4. 文件操作: ls、mkdir、cp、cat
5. 磁盘与分区操作: df、du、mount、fdisk、swapon、hdparm、dmesg

## 4.3 重定向命令

操作系统定义了两个标准输入和输出设备。

重定向操作即把命令缺省的输入来源或输出方向修改为其他文件/设备。

输入/输出文件	设备	文件编号
标准输入文件	键盘	<b>0</b>
标准输出文件	显示器	<b>1</b>
标准错误输出文件	显示器	<b>2</b>

图 2: 标准输入设备和输出设备文件编号

重定向操作的例子 (Linux):

将标准输出重定向到文件 `ls /etc/ > etc.log`。

对错误不进行任何处理, `ErrCmd 2> /dev/null`

类别	操作符	说明
输入重定向	<	将命令输入由默认的键盘更改/重定向为指定的文件
输出重定向	>	将命令输出由默认的显示器更改/重定向为指定的文件
	>>	将命令输出重定向并追加到指定文件的末尾
错误重定向	2>	将命令的错误输出重定向指定文件（先清空）。
	2>>	将命令的错误输出重定向指定文件（追加到末尾）。
输出与错误组合重定向	&>	将命令的正常输出和错误输出重定向指定文件。

图 3: 重定向操作符及含义

## 4.4 管道命令

特殊的重定向操作，程序的输出作为另一个程序的输入。

管道操作符 | 用于连接左右两个命令，将 | 左边命令的执行结果（输出）作为 |。

注意管道对错误信息是没有处理能力的。

# 5 批处理命令

批处理与脚本程序都是在在控制台环境下自动处理一批命令。比较常见的有 Windows 批处理程序，Linux Shell 脚本程序。

## 5.1 批处理 (Windows): BAT/PowerShell

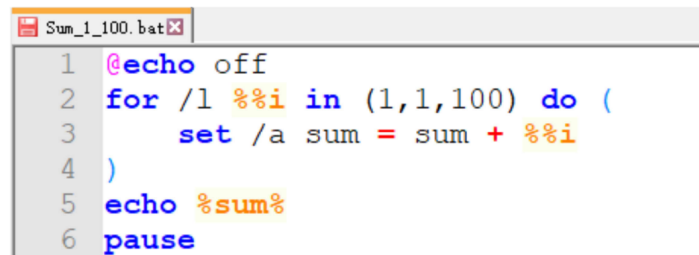
批处理程序命令可编程。批处理程序支持变量替换、条件、转移、循环、注释等简单语法。此外，批处理程序命令也像普通命令十样，支持在命令行上使用参数来扩展命令的使用方式。

批处理程序是文本文件，批处理文件的扩展名为.bat ( Batch 的缩写)。其可直接执行，由 command 解释执行。

应当注意批处理文件中变量的生命周期将与控制台生命周期同步。

常用的批处理指令如下：

1. 回显命令：echo 和 @，echo off 关闭命令本身的输出
2. 注释命令：rem 和::
3. 跳转命令：goto



```
1 @echo off
2 for /l %%i in (1,1,100) do (
3     set /a sum = sum + %%i
4 )
5 echo %sum%
6 pause
```

图 4: 求 1-100 之间所有整数和的批处理程序命令

4. 命令行传递给批处理的参数: %0 %1 %2 %3 %4
5. 判断指令: if 条件 (指令 1) else (指令 2)
6. 设置变量: set 引用变量可在变量名前后加%, 即% 变量名%。set /a  
p=39
7. 循环命令: for。例 1: for %%i in (c: d: e: f:) do echo %%ili, 例 2:  
for /l %%i in (2,1,8) do echo %%i
8. pause 命令。

## 5.2 Shell 脚本: Shell Script

脚本 (Script) 通过类似程序的方式执行具有一定逻辑顺序的命令序列完成较复杂的功能和人机交互。脚本程序保存在文本文件中; 脚本程序是 Shell 命令语句的集合。

### 5.2.1 Shell

Shell 是操作系统与用户交互的界面, Shell 表现为通过控制台执行用户命令的方式, 本身不执行命令, 仅仅是组织和管理命令。

### 5.2.2 Shell 的主要类型

Shell 的主要类型有 UNIX 和 LINUX 环境下的 bsh、csh、ksh、bash, 以及 Windows 环境下的 PowerShell。

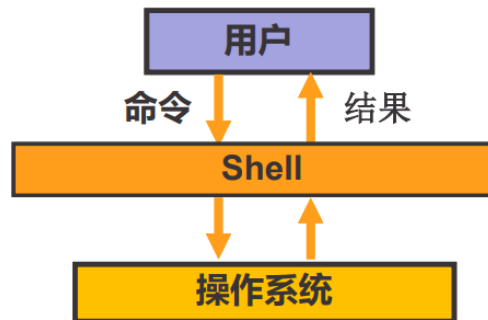


图 5: Shell 在系统中的地位

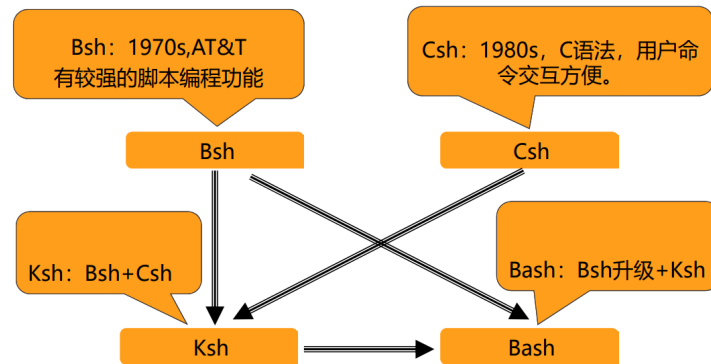


图 6: Shell 的发展与分类

### 5.2.3 Shell 的主要功能

1. 命令行编辑功能 (左右方向键、退格键、Del、Home、End、Ctrl + u, Ctrl + k)
2. 命令和文件名补全功能 (Tab)
3. 命令历史功能 (上下方向键、\$ history、/.bash\_history)
4. 命令别名功能
5. 提供作业控制功能
6. 具有将命令序列定义为功能键的功能
7. 重定向与管道
8. Shell Script 脚本编程

```
1 UserName="SuShuguang"
2 echo ${UserName}
3 for file in `ls /etc`
4 echo $file
5 read -p "Please Input Your Name" YourName
6 echo $YourName
```

图 7: 变量引用和定义的方式

#### 5.2.4 运行脚本程序的三个方法

1. 直接运行（用缺省版本的 Shell 运行脚本程序），可能需要使用 `#chmod +x test.sh` 为 Shell 脚本加上执行权限。
2. 在脚本文件首行指定 Shell。`#!/bin/bash` —— `#!` 必须顶格，后接 shell 全路径
3. 使用某个特定版本的 Shell 执行脚本。`$bash test.sh`。可从 `/etc/shell` 获知所有可用 shell 及其绝对路径。

#### 5.2.5 变量与引用

定义变量主要有两种方式: 一是, 通过显式的直接赋值语句来定义变量, 变量名和等号之间不能有空格。二是, 通过语句给变量间接赋值来声明变量。

引用变量的时候, 在变量名前面使用美元符号 `$` 作为前缀。

#### 5.2.6 键盘输入和屏幕输出

`read` 命令的主要选项: `-n`: 限制读取 `N` 个字符就自动结束读取。`-p`: 给出提示符。`-s`: 静默模式。

`echo` 命令的主要选项: `-n`: 不要在最后自动换行 `-e`: 使用转义符。

#### 5.2.7 条件判断和分支语句

```
if condition
then
    statement(s)
fi
-eq -gt -le 等判断语句
```



### 5.2.8 循环语句

```
for var in item1 item2 ... itemN
do
    command1
    command2
    commandN
done

while condition
do
    Command
done
```

## 6 用户图形界面

用户图形界面 (GUI, Graphic User Interface)  
窗口, 图标, 菜单, 按钮, 鼠标 (消息, 事件)

## 7 系统调用

系统调用 (System Call, System Service Call) 是操作系统内核为应用程序提供的一系列服务/函数。例如: `printf,exit,fopen,fgetc,21H(09)`。

### 7.1 系统调用的特点

一般涉及核心资源或硬件的操作  
系统调用运行于核态。  
系统调用过程会产生中断: 自愿中断  
系统调用数量众多

### 7.2 系统调用的实现形式

使用指令: `SVC N`  
`SVC`: SuperVisor Call, 访管指令

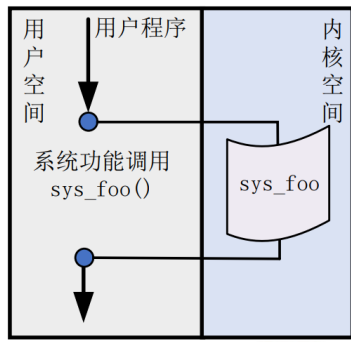


图 8: 系统调用的基本概念

编号	名称	功能	调用参数
00	setup	安装根文件系统	EBX=硬盘参数表地址
01	exit	退出进程	EBX=退出码
02	fork	创建进程	
03	read	读文件	EBX=文件描述符, ECX=缓冲区首址, EDX=字节数
04	write	写文件	EBX=文件描述符, ECX=缓冲区首址, EDX=字节数

图 9: Linux 部分系统调用

SVC 是中断指令，N 为中断号

### 7.2.1 DOS 系统调用

INT 21H

AH 中存放系统调用号 ( $\leq 256$ )

### 7.2.2 Linux 系统调用

INT 80H

EAX 中存放系统调用号

### 7.2.3 隐式系统调用

在高级语言中使用，包含中断指令/系统调用。

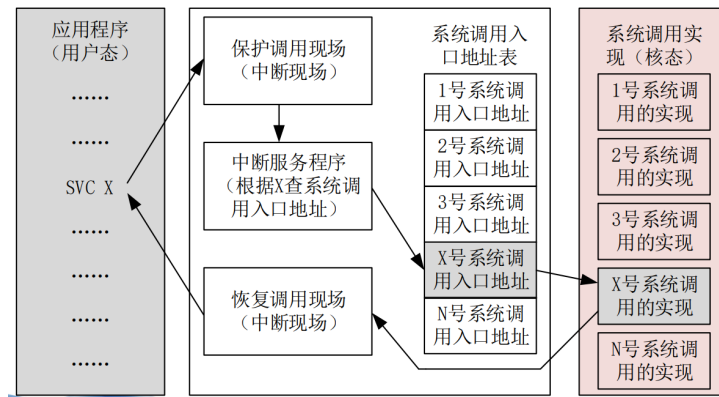


图 10: 系统调用的执行过程（中断）

### 7.3 系统调用的执行过程（中断）

### 7.4 Linux 系统调用

#### 7.4.1 Linux 系统调用的工作原理

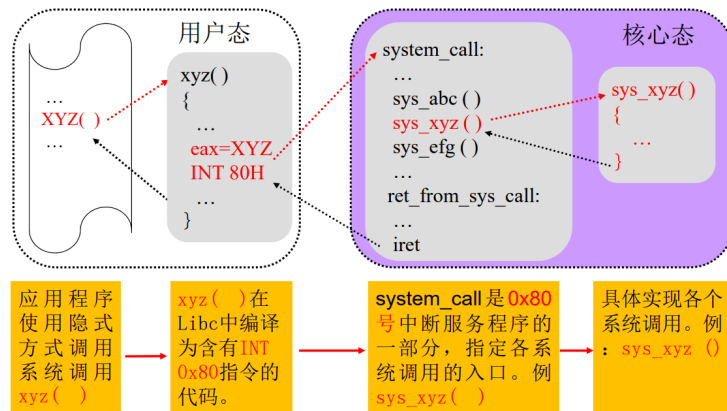


图 11: Linux 系统调用的工作原理

1. 应用程序调用封装有特定系统调用的库函数。
2. 库函数展开为内含 `INT 0x80` 指令和系统调用编号的汇编指令，调用相应的系统调用。
3. 进入 `INT 0x80` 的中断处理函数，并调用相应的系统调用函数。

4. 系统调用函数完成用户请求的服务。
5. 返回应用程序。

#### 7.4.2 Linux 系统调用的代码实现

1. 0x80 中断初始化, 每执行 INT 0x80 指令时, 产生一个中断使系统陷入内核空间并执行 0x80 号中断处理函数 `system_call()`。
2. 服务程序 `system_call()`。以 `%eax` 遍历表 `sys_call_table` (系统调用处理函数指针表), 查找对应的服务子程序。`CALL *SYMBOL_NAME(sys_call_table)(%eax, 4)`。
3. 系统调用编号的声明。格式: `#define __NR_CallName ID`
4. 系统调用函数的声明。格式: `.long sys_XXXX`
5. 系统调用函数的定义。格式: `asmlinkage int sys__mycall()`

#### 7.4.3 系统调用函数的调用方法

直接调用 (funcname 不带 `sys_` 前缀) :

```
type = syscall(__NR_funcname, arg1, arg2, ...)
```