# UNIX
# 系统程序设计

黄祥喜　编著

中山大学出版社

# UNIX 系统程序设计

黄祥喜　编著

中山大学出版社
·广州·

**图书在版编目(CIP)数据**

UNIX 系统程序设计/黄祥喜编著．—广州：中山大学出版社，1995.10
ISBN7－306－01071－9

Ⅰ.U… Ⅱ.黄… Ⅲ.①计算机 ②操作系统 ③程序设计 Ⅳ.TP3

中山大学出版社出版发行
（广州市新港西路 135 号　邮编 510275）
广东乳源印刷厂印刷
广东省新华书店经销
*
787×1092 毫米　16 开本　19 印张　476 千字
1995 年 10 月第 1 版　1995 年 10 月第 1 次印刷
印数：1—5000　　定价：26.00 元

本书的出版得到
**广东省自然科学基金、广东省重点课程建设基金**
**中山大学重点课程建设基金**
的资助，特此致谢。

# 内容提要

本书从UNIX程序员的角度，较详细地论述了UNIX系统调用和库函数的使用方法。内容包括：UNIX系统程序设计概述，文件和目录操作程序的设计，文件保护命令的实现，文本信息处理程序的设计，文件系统和磁盘操作程序的设计，终端操作程序的设计，进程控制程序的设计，进程通信程序的设计等。书末还提供了两个附录，对目前主要的UNIX版本(V7，BSD 4.2，系统V 3.0和4.0，POSIX，SVID)所提供的用户命令和C库函数(包括系统调用)作了详细的比较。本书内容系统，程序完整实用，语言流畅，对广大的UNIX系统开发者来讲，是一本极有实用价值的参考书。

# 前　言

UNIX 作为一个通用的分时多用户多任务操作系统，目前已成功地运行在各种档次计算机上。如巨型机 CRAY、IBM 和 NEC，大型机 DEC、IBM、BULL 和 CDC，小型机 HP、DEC、CDC 和 BULL，工作站 HP 和 SUN，微型机 IBM PC 和 MAC 等。以 UNIX 为基础的操作系统标准化工作也在积极地开展，如 IEEE 和 ISO 公布了 POSIX 标准，AT&T 公司公布了 SVID 标准。前者是通用操作系统的标准，它吸取了目前主要的 UNIX 版本(如 UNIX 系统 V、加州大学伯克利分校的 BSD 、SUN 公司的 SUNOS、Microsoft 公司的 XENIX 等)的优点，后者是 UNIX 系统 V 的专门标准。更为重要的是，UNIX 开启了"开放系统"之门。我们可以这样说，在"开放系统"时代，不学习 UNIX，必将被时代所淘汰。

UNIX 的学习可以分为三个层次：对于一般用户而言，只要学会 UNIX 的操作命令，就可满足其需要；对于系统开发者来讲，仅学会命令的使用是不够的，还要学会如何使用 UNIX 提供的系统调用和库函数，实际上命令的实现是以系统调用为基础的；如果你想了解 UNIX 的内部构造，则还要分析 UNIX 的内核结构，这就要阅读 UNIX 的内核源代码。本书是为满足 UNIX 系统开发者的需要而编写的。

本书共分 9 章，第 1 章对 UNIX 的发展历史、系统程序设计的概念、UNIX 常用系统调用的使用格式、UNIX 的常用开发工具及 SHELL 作了概要说明。第 2～4 章论述了 UNIX 的常用文件和目录操作命令的实现方法。第 5 章论述了 UNIX 的常用文本和信息处理命令的实现方法。第 6 章论述了 UNIX 的文件系统结构和常用磁盘操作命令的实现方法。第 7 章论述了 UNIX 的终端管理方法和常用终端操作命令的实现。第 8 章讨论了 UNIX 的进程控制机制，特别强调了几个进程控制系统调用的联合作用。第 9 章对 UNIX 的通信机制作了详细讨论，信号和管道是所有 UNIX 系统都有的机制，IPC 机制则是 UNIX 系统 V 特有的机制。

本书的一个突出特点是，以实际命令的实现程序的分析为主线，阐述 UNIX 常用系统调用和库函数的使用方法。这样，便于读者通过阅读实际命令的实现代码，使之既可以更深入地体会 UNIX 命令的用法，也可以学会 UNIX 系统调用和库函数的综合使用方法，还可以从系统调用的层面上了解 UNIX 的功能结构。UNIX 对网络提供了强有力的支持，此书没有涉及，我们计划在另一本书中详细讨论 。本书中的绝大多数程序都可运行在作者主持研制的 MINIX+1.0 操作系统实验环境中。MINIX+1.0 是一个教学用类 UNIX 系统，与 UNIX V7 完全兼容，是 A. S. Tanenbaum 1986 年研制的 MINIX 初版的扩充版，可运行在 IBM PC 8088、286、386 等微机上。若读者想在自己的 UNIX 系统上运行本书中的程序，可阅读本书中的两个附录。这两个附录对几个主要的 UNIX 版本(UNIX V7，BSD 4.2，系统 V 3.0 和 4.0，POSIX，SVID)所提供的系统调用和 C 库函数作了详细的比较。读者对本书中的程序稍作修改，即可将其运行在自己的 UNIX 系统上。若读者需要 MINIX+1.0 软件，可与作者联系。

阅读本书需要有 C 语言的基础。如果读者不熟悉 C 语言程序设计，则应先阅读一本 C 语言方面的教材。

本书在编写过程中，引用了 A. S. Tanenbaum 等人编写的部分程序，特此说明。

恳望读者能对本书中的不足和错误提出批评。

作　者

1995 年 7 月 27 日于广州康乐园

# 目　录

# 1 概　述

## 1.1 UNIX 的概念

### 1.1.1 UNIX 的构成

UNIX 是目前国际上流行的一种多用户分时操作系统。它可在巨型机、大型机、中型机、小型机、微型机、工作站、网络等各种硬件环境下运行。1969 年，美国 AT&T 公司的 K. Thompson 和 D. Ritchie 编写出 UNIX 第 1 版。目前流行的 UNIX 版本有 AT&T 公司的 UNIX 系统 V 4.0、美国加州大学伯克利分校的 UNIX BSD 4.2 和 4.3(SUN 工作站上的 SUNOS 是以 BSD 为基础的)、SCO 公司的 SCO UNIX 系统 V 3.2 和 4.0、Microsoft 公司和 SCO 公司的 XENIX 系统 V 2.3.2 等。这些系统尽管在内部实现、实用软件、硬件环境等方面具有不同的特点，但在用户界面和基本结构方面是一致的。图 1.1 给出了 UNIX 操作系统的基本结构。图 1.2 给出了 UNIX 系统 V 内核框图。

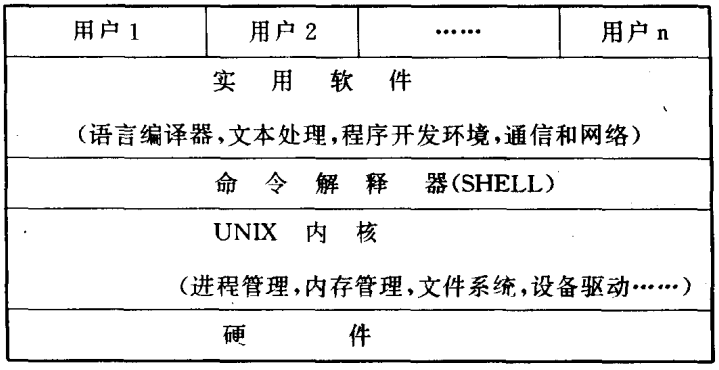


**图 1.1　UNIX 操作系统的基本结构**

### 1.1.2 UNIX 的主要特点

UNIX 能够成为目前世界上最著名的多用户操作系统，是因为它具有以下特点：

(1) 良好的可移植性。UNIX 系统的内核和核外系统程序主要是用 C 语言写的，因此，系统易于理解、修改和移植。

(2) 丰富的核外系统程序。UNIX 包含有丰富的语言处理程序、实用软件和软件开发工具。用户可通过 SHELL 命令使用它们。

(3) 良好的用户界面。UNIX 提供了丰富的用户命令和功能强大的命令解释器 SHELL。SHELL 不仅是一种交互式命令语言，同时也是一种程序设计语言。实际上，UNIX 的有些实用程序就是用 SHELL 编写的。程序员用户还可在编写汇编语言程序和 C 程序时使用 UNIX

的系统调用。

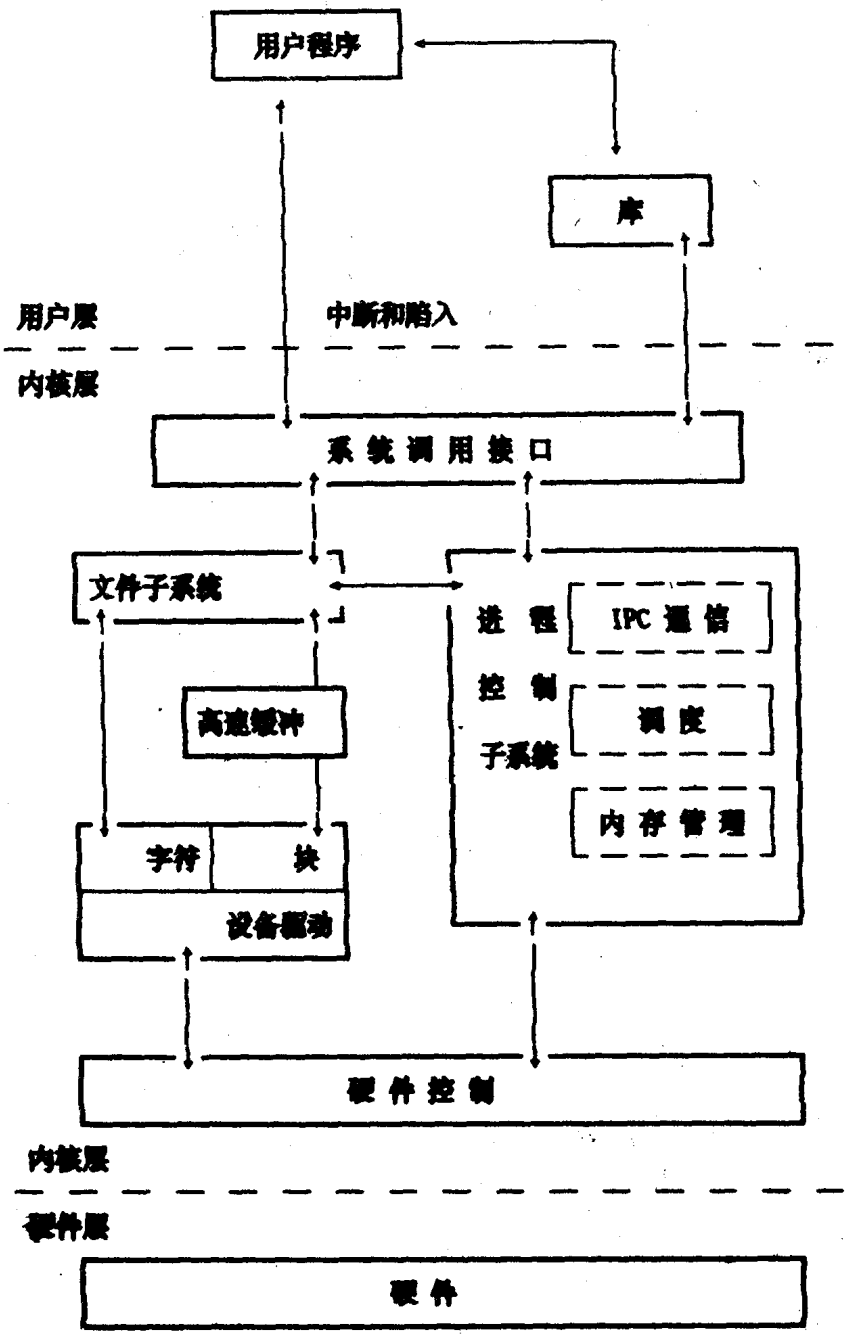


图 1.2　UNIX 系统 V 内核结构图

(4) 文件和设备的统一处理。UNIX 向用户提供了字符流的文件结构，并且普通文件、文件目录和外部设备都被视作文件，它们在用户面前具有相同的语法和语义、相同的存取方法和保护机制。

(5) 分层结构的文件系统。UNIX 向用户提供一个树形分层结构的文件系统，它由基本文件系统和若干可装卸的子文件系统组成。这种结构既有利于扩大文件存储空间，又有利于安

全和保密。

(6) 先进的设计思想。**UNIX** 是对现有操作系统技术的精炼和发展。在总体设计思想上，它突破了以往设计中贪大求全的惯例，而着眼于向用户提供一个良好的程序设计环境。在这种设计思想的指导下，**UNIX** 的内核设计得十分精巧，能够常驻内存。设计者将从内核中分离出来的大量的核外系统程序放在用户环境下运行，并注意了内核和核外程序的有机结合，使得两者作为一个整体向用户提供了良好的服务。

### 1.1.3 UNIX 的标准化

UNIX 的标准化工作是在 1980 年中期开始的。在此之前，许多厂商在 AT&T 公司的许可下，都开发了自己的 UNIX 版本。这些 UNIX 版本及在其上开发的应用软件互不兼容，给程序的移植工作带来极大的困难。1985 年，国际 UNIX 组织(UI)和 AT&T 公司公布了 UNIX 系统 V 接口定义标准 SVID。这个文件在系统调用、例行程序库和实用程序等方面描述了 UNIX 系统 V。UNIX 系统 V 的所有提供者都必须遵循这一标准。对于软件设计者来说，用 SVID 中所描述的操作编写的程序可以移植到另一个 UNIX 系统 V 上，而不管其支撑机种如何。

另一个著名的操作系统标准是计算机环境下的可移植操作系统规范 POSIX。它首先是由美国 IEEE 学会 P1003 标准化委员会制订的。1986 年，IEEE 提出了“试用 POSIX 标准草案”。这一标准得到了 IBM 等大型计算机公司和 X/OPEN 等国际上有影响的标准化组织的支持。后来，国际标准化组织 ISO 和国际电子技术委员会 IEC 以此规范为基础，联合制定了操作系统的国际标准 POSIX. 1。这个规范包括可移植操作系统规范、命令语言 SHELL 及其实用程序、系统管理界面等。我国目前也在以 POSIX. 1 规范为基础，制定操作系统的国家标准。POSIX. 1 和 SVID 两个标准是一致的，区别在于：SVID 是针对 UNIX 系统 V 制定的标准，而 POSIX. 1 则是针对任何操作系统的。就其内容来说，POSIX. 1 规范反映了 UNIX V7、UNIX 系统Ⅲ、UNIX 系统 V、UNIX BSD 4. 2 和 4. 3 等操作系统的共性。

为了实现软件在更大范围的通用性和可移植性，国际上除了对 UNIX 操作系统本身进行标准化之外，在与 UNIX 有关的网络、程序设计语言和用户界面等方面也制定了相应的标准。比如，网络方面制定了 OSI(开放系统互连)模式，数据处理方面制定了 SQL、ISAM 等标准，所有程序语言的标准由 ANSI 制定，用户界面则以 X－Windows 系统为主。这样，所有硬件厂家、系统软件开发公司、应用软件开发公司和最终用户都有标准可遵循。

### 1.1.4 MINIX 和 UNIX 的关系

UNIX 是一个功能强大的实用多用户操作系统，分析和研究起来并不是一件容易的事情。为此，国际知名操作系统学者 A. S. Tananbaum 研制了一个 UNIX 教学系统 MINIX，现在已在许多大学中使用。MINIX 的用户界面与 UNIX V7 完全兼容，但在内部实现上与 UNIX 不全相同 。近几年，我们对 MINIX 作了大量的研究和开发工作。本书中的程序都是在 MINIX 环境下开发的，现在已正确地运行在该系统中。由于 MINIX 和 UNIX V7 的用户界面兼容，在 MINIX 环境下开发的程序自然能够运行在 UNIX V7 中。UNIX 系统 V 是以 UNIX V7 为基础开发的扩充版，因此，本书中的程序稍作修改即可在 UNIX 系统 V 环境下运行。关于 MINIX 系统的详细论述，参见 A。S. Tananbaum 著《操作系统教程》(世界图书出版公司，1989 年) 和黄祥喜主编《计算机操作系统实验教程——MINIX 的使用、分析和实现》

(中山大学出版社，1994)。

## 1.2 系统程序和系统程序设计

### 1.2.1 系统程序的概念

系统程序是相对于应用程序来讲的。通常把计算机系统中可供用户直接使用的软件叫做系统程序。典型的系统程序有：

(1) 操作系统：其作用是管理计算机硬件和软件资源，为用户提供方便、易用的接口。如UNIX、DOS、OS/2、WINDOWS、WINDOWS NT 等。

(2) 编译程序：其作用是将用高级语言(如C、PASCAL)写的程序翻译成可在机器上执行的目标指令。如C 语言编译器CC 等。

(3) 汇编程序：其作用是将用汇编语言(符号语言)写的程序翻译成目标指令，如 IBM PC 机的汇编程序 MASM 或 ASM。

(4) 连接装配程序：其作用是将用户分开编译或汇编形成的若干独立的目标程序连接装配成一个整体，然后交给机器执行。如果用户在程序中使用了系统的标准函数库，则连接装配时还要调用系统的库函数，如 IBM PC 机中的 LINK 程序。CC 编译器中的某些选择项可使系统在编译时自动执行连接功能。

(5) 命令解释程序和命令实现程序：命令解释程序的作用是对用户输入的命令进行解释执行，如 UNIX 操作系统的 SHELL、DOS 的 command.com 和 dosshell 程序等。命令实现程序的作用是为用户提供丰富的命令。有的命令实现程序可能是一个单独的可执行文件(这样的命令通常称之为外部命令)，有的则属于命令解释器的一部分(这样的命令通常称之为内部命令)。DOS 中的 format、backup 等，UNIX 的 cp、ls 等都属于外部命令；而 DOS 中的 cd、copy 等命令，UNIX 中的 cd、exec 等命令都属于内部命令。

(6) 数据库管理系统：数据库管理系统是在操作系统的文件系统基础之上发展而来的。它的作用是将一个单位或部门所需的数据综合地组织在一起，构成数据库，并可使用户高效、方便地使用数据库中的数据。目前典型的数据库管理系统有 FOXBASE、ORACLE 等。

(7) 编辑程序：其作用是帮助用户建立和编辑源程序文件、数据文件或文本文件等。典型的编辑程序有 DOS 的行编辑器 EDLIN，UNIX 的全屏幕编辑器 vi、行编辑器 ed 等。

(8) 程序开发工具：其作用是帮助用户快速、高效地开发各种程序。比如调试程序可帮助用户查找和纠正程序中的错误。DOS 的 DEBUG，UNIX 的 adb、sdb 等都是常用的调试程序。

(9) 格式加工程序：其作用是将用户建立的文本文件加工成所需要的格式，然后打印输出。IBM PC 机中的 wordperfect，UNIX 中的 nroff、troff 等都是功能很强的正文格式加工程序。汉字编辑程序 WPS 兼有编辑和格式加工两个功能。

### 1.2.2 系统程序设计方法

系统程序设计是指设计系统程序的过程。在 UNIX 环境下，设计系统程序可以使用以下三种方法：

(1) 使用 SHELL 语言：UNIX 的命令解释器 SHELL 既是一种命令语言，又是一种程序

设计语言，它提供了类似高级程序语言的控制结构。用户可以使用 SHELL 语言扩充 UNIX 的命令集，编写自己所需要的系统程序，如软件安装程序。1.3 节对 shell 作了较详细的介绍。

(2) 使用开发工具：UNIX 提供了丰富的开发工具，包括文件处理、程序开发、语言开发和程序调试等工具。用户使用这些工具可提高开发程序的效率。1.5 节对 UNIX 的开发工具作了简单介绍。

(3) 使用系统调用：系统调用是操作系统向用户提供的一组功能子程序，用户可在高级语言中以库函数的形式来使用它们。在 UNIX 环境下，用户可以在 C、PASCAL 等语言中以库函数的形式来使用 UNIX 的系统调用。用户使用系统调用，既可以编写系统程序，又可以编写应用程序。实际上，UNIX 的所有系统程序(包括 SHELL 解释器及 UNIX 开发工具)都是以 UNIX 系统调用为基础编写的。1.4 节对 UNIX 的系统调用作了一般性介绍。本书以后各章将详细介绍如何使用 UNIX 系统调用编写 UNIX 的命令实现程序。

## 1.3 UNIX 的命令解释器 SHELL

作为一个命令解释器，SHELL 不属于操作系统的范围，但它是用户与操作系统之间的原始接口，是用来执行程序和编写程序的最基本工具。

SHELL 执行的命令或来自终端或读自文件。因此用户可以在提示符后打入命令马上执行，也可以编写一个 SHELL 命令程序，提交 SHELL 执行。用这两种方法，用户可以方便地建立能反应各人的不同要求和风格的环境。

### 1.3.1 SHELL 作为一种命令语言

在 UNIX 环境中，用户和系统的对话是通过 SHELL 进行的。在系统提示符下，SHELL 能够逐一解释、执行用户键入的命令。命令是由用空格分隔的字的序列写成的。第一个字是被执行命令的名字。其余的字作为参数传递给被调用的命令。

为执行一个命令，SHELL 通常建立一个新的进程并等待它的完成。完成这两个操作的是 FORK 及 EXIT 系统调用(参见 1.4 节)。使用后缀操作符 &，可使命令的运行不必等到它完成，即建立一个后台进程来执行这个命令。采用这种方法执行命令时，SHELL 在建立进程以后即报告它的进程号。

对于每一个进程，系统保留一份编号为 0,1,2,…的文件描述字的集合，它将在所有进程与操作系统之间的输入——输出交换中起作用。文件描述符 0,1,2 分别称为标准输入、标准输出及错误输出。它们均可以在某个命令进程的生存期内被重新定向，比如：

> file　　标准输出改向为 file，如果该文件不存在，则建立它。

>>file　　标准输出改向为 file，如果 file 已存在，则把标准输出放在文件 file 的尾部，否则建立该文件。

< file　　标准输入改由 file 输入。

上述各项之前均可置以一个文件描述字(数字)，以代替缺省的 0 和 1。例如，2>file 表示错误输出改向至文件 file，2>&1 表示错误输出与标准输出合并。

在一个 SHELL 程序内部，标准输入、标准输出和错误输出可用上面的形式和 exec 命令来重定向。例如：

exec >stdout 2>errout

使得随后被执行命令的缺省标准输出及错误输出改向至 stdout 及 errout。

对于后台命令，它的缺省标准输入是 /dev/null，这样可以防止后台命令 和 SHELL 同时读取同一个输入而引起混乱。

UNIX 的命令还有内部命令和外部命令之分。 外部命令的实现程序是一个独立的可执行文件，在执行它们时 SHELL 要建立进程。典型的外部命令有：

| | |
|---|---|
| cp | 复制文件 |
| kill | 发送信号给一个进程 |
| mount | 安装一个文件系统 |

内部命令是 SHELL 解释器的一部分，在执行它们时，SHELL 将不建立进程。 典型的内部命令有：

| | |
|---|---|
| cd [arg] | 将用户的当前目录改为 arg。如果 cd 不带参数，则转到 $HOME。 |
| export [name...] | 标记 name 变量为输出变量，使得 name 被自动输出给子进程的环境。若该命令不带参数，则列出具有输出状态的变量名。 |

### 1.3.2 SHELL 作为一种程序语言

SHELL 除了执行终端命令之外，还可以依次读取并执行包含在一个文件中的命令。这个文件通常称作 SHELL 程序。例如：

$ sh file [args …]

将调用 SHELL 从文件 file 中读命令，而且把 args 作为参数传送至 file。如果使用命令 chmod 把 file 的执行属性(即 x 位)置位，则命令：

$ file [args…]

与上述命令 具有同等作用。对于参数，可用 $0，$1 … $9 或 $ * 引用。

SHELL 程序可以包含类似 C 语言的控制结构，如条件语句、循环语句等。

1) 条件语句

UNIX 中的命令在执行完毕时都要送回一个状态值，SHELL 会把它放于 SHELL 变量 $? 中。用户有时要根据这些退出状态值而做不同的事。这时可利用 if 语句或 case 语句。下面给出 if 语句的语法描述：

```
if command
    then
        commands
    else
        commands
fi
```

then、else、fi 之前必须是分号或换行符。else 及其后的语句可以没有。

if 语句的含义是，当 command 的返回状态为 0(命令执行成功)时，执行 then 后面的语句序列；否则执行 else 后的语句。当 else 后跟一条件语句时，可以合写为 elif。

2) 循环语句

程序设计中，循环语句是必不可少的。SHELL 提供了三种循环语句，即 FOR 语句、UN-

TIL 语句和 WHILE 语句。这里给出 UNTIL 语句的格式：

```
until command1
    do
        command2
    done
```

UNTIL 语句的含义是：当 command1 命令为假时，执行 command2；command1 命令为真时退出循环。

SHELL 还提供了一套关于变量的管理机制。变量可以由以下命令格式赋值：

```
name=str
```

这里，名字为 name 的变量将以字符串 str 为值。

在变量名前加 $ 可引用变量的值。SHELL 在解释命令时，如果遇到不用单引号(') 引起的并且没有转义符号('\')的变量引用，将用它的值替代变量所处位置。例如：

```
b=/usr/fred/bin
mv pgm $b
```

等价于命令：

```
mv pgm /usr/fred/bin
```

又比如：

```
tmp=/tmp/ps
ps a >${tmp}a
```

等价于命令：

```
ps a >/tmp/ps a
```

上例中符号 {} 用于指明变量名。如果不用 {}，SHELL 将误认为变量名为 "tempa"。

SHELL 变量可以设置成为两种状态。用户建立的 SHELL 变量的状态均为非输出状态及非只读状态，即它的作用域仅为本 SHELL 进程，并且其值可以修改。而内部命令 export 及 readonly 可以把 SHELL 变量的状态分别置为输出状态(此变量自动作为下一条命令的环境)和只读状态(在本 SHELL 中此变量的值不可修改)。

SHELL 把一些变量的值留作系统专用变量，它们包含了系统的一些配置参数。下面作一简单介绍：

$?　　被执行的命令的返回状态，是一个十进制数字串。

$0　　正在被执行的命令名。

$1－$9　依次表示命令行中第 1 至第 9 个参数的值。可用于编写带参数的 SHELL 程序。

$#　　参数个数(十进制)。

这里，$1－$9 及 $# 均可由内部命令 set 修正。

$$　　本 SHELL 进程的进程号(十进制)。

以下的一些变量对 SHELL 有特殊含义，应避免作一般用途。这些变量一般在用户主目录中的 profile 文件中设置。

$HOME　内部命令 cd 的缺省参数。它一般存放用户的主目录(用户注册目录)

$PATH　命令的搜索路径清单。每当执行一个外部命令(区别于内部命令)时，SHELL