上海Linux产业发展专业人才培训系列教材

Linux开发篇—环境编程技术

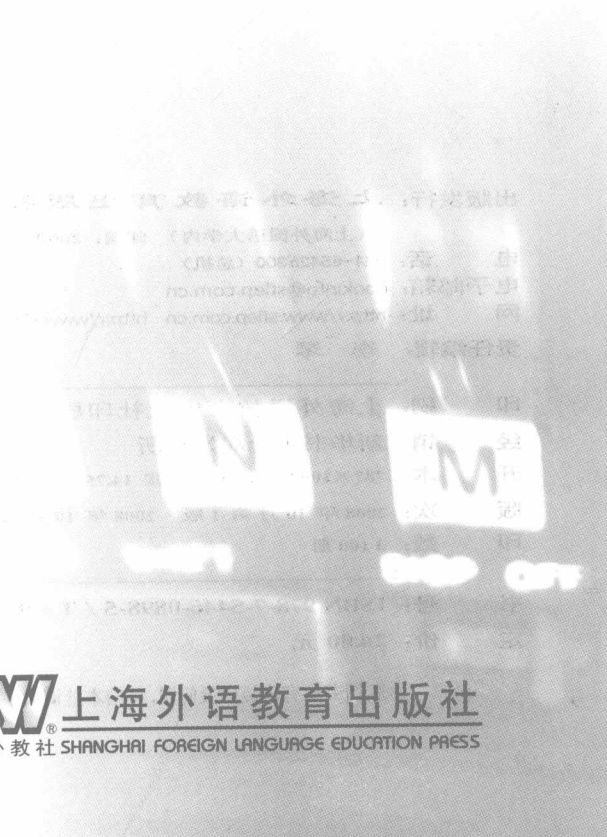
上海市信息化委员会
上海Linux产业发展专业人才培训基地 编

上海外语教育出版社
外教社 SHANGHAI FOREIGN LANGUAGE EDUCATION PRESS

上海Linux产业发展专业人才培训系列教材

Linux开发篇——环境编程技术

上海市信息化委员会
上海Linux产业发展专业人才培训基地 编



上海外语教育出版社
外教社 SHANGHAI FOREIGN LANGUAGE EDUCATION PRESS

图书在版编目(CIP)数据

Linux 开发篇：环境编程技术/上海市信息化委员会,上海 Linux 产业发展专业人才培训基地编
—上海：上海外语教育出版社,2008
(上海 Linux 产业发展专业人才培训系列教材)
ISBN 978-7-5446-0898-5

Ⅰ. L… Ⅱ. ①上…②上… Ⅲ. Linux 操作系统-技术培训-教材 Ⅳ. TP316. 89

中国版本图书馆 CIP 数据核字(2008)第 104296 号

出版发行：上海外语教育出版社
（上海外国语大学内） 邮编：200083
电　　话：021-65425300（总机）
电子邮箱：bookinfo@sflep.com.cn
网　　址：http://www.sflep.com.cn http://www.sflep.com
责任编辑：李　琴

印　　刷：上海外语教育出版社印刷厂
经　　销：新华书店上海发行所
开　　本：787×1092　1/16　印张 14.75　字数 358 千字
版　　次：2008 年 10 月第 1 版　2008 年 10 月第 1 次印刷
印　　数：3 100 册

书　　号：ISBN 978-7-5446-0898-5 / T · 0004
定　　价：24.00 元

《Linux 开发篇——环境编程技术》
编写人员

主　编　戚正伟

前言

在二十世纪九十年代初，Linux 操作系统的问世催生了一类新颖的源代码开放的自由软件。Linux 系统具有良好的安全性、可靠性、可扩展性、平台无关性和较低的价格成本，博得众多国际著名计算机研发和应用厂商的青睐和支持，正成为信息化建设的一个重要平台。在 Linux 技术传播过程中，它的应用规模和范围不断扩大，不仅是企业的边缘业务应用，而且企业的核心业务应用和关键业务应用也采用了 Linux 系统。Linux 的足迹已从服务器领域伸展到桌面系统和嵌入式应用。Linux 技术展示了当前基础软件发展的一个重要方向，为软件产业重构和软件市场发展带来了全新的理念，正成为推动 IT 应用发展的新动力和新潮流。

国家和上海市信息产业“十一五”专项规划强调指出，要集中力量支持关键基础软件产品和技术的自主研发，以及核心技术和重大产品的产业化；建立基于国产软件的大型应用示范项目，整体推进具有自主知识产权的国产基础软件的应用和发展。Linux 技术的发展为自主研制我国的基础软件提供了重要机遇。因此，在“十一五”期间，加强 Linux 技术的开发和应用，推出具有自主知识产权的 Linux 产品，着力推动基础软件和嵌入式技术的研发与应用，争取掌握若干具有自主知识产权的核心技术，是一项极为重要的任务。

2000 年以来，在国家关于促进软件产业发展政策和上海相关配套政策的激励下，上海的 Linux 产业已进入了一个平稳的发展时期，取得了长足的进步。据有关权威机构预测，未来几年内，我国的 Linux 市场将保持 40% 的增长速度，技术人才特别是高端技术人才紧缺将成为制约 Linux 产业发展的瓶颈。由此可见，在 Linux 市场快速发展的情势下，加速 Linux 技术的传播和人才的培养，是进一步推进上海 Linux 的开发、应用，完善 Linux 产业链及壮大上海软件产业的关键之一。

近几年来，上海市政府加大专项资金投入，推动 Linux 产品的研发和应用，并从发展环境、人才培养、中介机构三方面给予大力支持。在上海市信息化委员会的支持下，上海有关的研发机构、高等院校和企业联合组建了“上海 Linux 产业发展专业人才培训基地”。为了进一步推动 Linux 技术的培训，“基地”组织专

家教授编写了这套 Linux 培训系列教材。教材的编写强调以需求为牵引、应用为主线、案例为驱动，注重实践性和针对性，实验和操作的环境采用我国目前产业界广泛使用的 Linux 产品。该系列教材在结构上分为基础篇、管理篇、开发篇和嵌入式篇四大部分。其中：

《Linux 基础篇》侧重介绍 Linux 基础知识、系统安装和环境、简单的操作命令和系统管理，以及常用的应用软件使用和典型 OA 软件包操作案例等。

《Linux 开发篇——环境编程技术》侧重剖析 Linux 内核结构与环境编程，阐释 Linux 系统架构和调用接口，底层库的安装、装载和调用，自开发库的生成和维护等。

《Linux 开发篇——Linux 平台上基于 JSP 的 Web 开发》侧重说明 Linux 应用的架构和平台的接口，并通过案例开发的讲述使读者掌握有关平台语言的编程技术、开源数据库的应用开发和管理等。

《Linux 嵌入式篇——嵌入式系统开发实践》侧重通过 Linux 的应用实验来讲解相关的 Linux 技术知识，特别是嵌入式环境下 Linux 系统的结构与特性、开发工具与环境的构造和使用、实时环境下操作系统的专题原理，以及 Linux 定制/裁剪技术和性能分析等。

《Linux 管理篇——系统管理》和《Linux 管理篇——网络管理》基于上海自主开发的 Linux 产品，注重有关管理的原理分析和操作实践，分别以系统管理员、服务器管理员、网络管理员和桌面用户为对象，讲解了 Linux 系统结构、系统管理的内容和实现，以及 Linux 系统的基本网络配置和高级配置、Linux 主机在 Internet 网络中的角色配置和相关的系统安全等问题。

Linux 培训系列教材的编写工作，是在上海市信息化委员会的指导下由华东计算技术研究所的华东电脑学院组织实施的。在教材的编写过程中，整合了上海 Linux 产学研各方面的资源，博采众长，发挥了各自的优势。参与本系列教材的主要编写人员有陆起涌、张晓先、戚正伟、冯瑞等。参与整套教材的框架设计及审阅的人员有尤晋元、丁晓东、朱宗尧、金光、陆起涌、杨宗源、刘文清、孙德功、梁阿磊、魏锋、张晓先、张泉、江敏、陈涵生、余性厚等。此外，孙瑞毅和林怡在整套系列教材的资料整理和文字编辑等方面也做了许多工作。

本系列教材可作为大专院校高年级学生和研究生的教学用书或参考书，也可作为中高级工程技术人员和管理人员学习 Linux 的技术参考书。

我们希望这套培训系列教材的出版和发行，能为进一步推动上海 Linux 的普及应用、加速 Linux 人才培养、提高 Linux 技术应用技能、促进软件产业的创新发展起到积极作用。本系列教材中的不当之处在所难免，恳请读者批评指正，以便在再版时修正和完善。

上海 Linux 产业发展专业人才培训系列教材编审委员会

二〇〇八年六月

目录

第1章 绪 论

1.1 引言

所有操作系统都向它们运行的程序提供服务。典型的服务有执行新程序、打开文件、读文件、分配存储区、获得当前时间等等，本书集中阐述了 Linux 操作系统为系统程序员所提供的各种系统调用服务。本章从程序设计人员的角度快速浏览 Linux，并对书中引用的一些术语和概念进行简要的说明并给出实例。在以后各章中，将对这些概念作更详细的说明。

1.2 登录

登录 Linux 系统时，先键入登录名，然后键入口令。系统将在其口令文件，通常是/ etc/ passwd 文件中查看登录名。登录后，系统先显示一些典型的系统信息，然后就可以向 shell 程序键入命令。shell 是一个命令行解释器，它读取用户输入，然后执行命令，用户通常用终端，有时则通过文件（称为 shell 脚本）向 shell 进行输入。常用的 shell 有：

（1）Bourne shell，/ bin/ sh

（2）C shell，/ bin/ csh

(3) Korn shell, /bin/ksh

(4) Bourne-Again shell, /bin/bash

系统从口令文件中登录项的最后一个字段了解到应该执行哪一个 shell。自 V 7 以来，Bourne shell 得到了广泛应用，几乎每一个现有的 Linux 系统都提供 Bourne shell。C shell 在伯克利分校开发，所有 BSD 版本都提供这种 shell。Korn shell 在大多数 UNIX 系统上运行，但在 SVR4 之前，通常它需要另行购买，所以没有其他两种 shell 流行。

口令文件登录项中的用户 ID(user ID)是个数值，它向系统标识各个不同的用户。系统管理员在确定一个用户的登录名的同时确定其用户 ID。通常每个用户有一个唯一的用户 ID，用户不能更改其用户 ID。用户 ID 为 0 是根用户(root)或超级用户(superuser)。在口令文件中，通常有一个登录项，其登录名为 root，我们称这种用户的特权为超级用户特权。如果一个进程具有超级用户特权，则大多数文件的许可权检查都不再进行。

口令文件登录项也包括用户的组 ID(group ID)，它也是一个数值。组 ID 也是由系统管理员在确定用户登录名时分配的。一般来说，在口令文件中有多个记录项具有相同的组 ID。在 Linux 下，组被用于将若干用户集合到部门中去。这种机制允许同组的各个成员之间共享资源(例如文件)。

组文件将组名映射为数字组 ID，它通常是/etc/group。对于许可权使用数值用户 ID 和数值组 ID 是历史上形成的。系统中每个文件的目录项包含该文件所有者的用户 ID 和组 ID。在目录项中存放这两个值只需 4 个字节(假定每个都以双字节的整型值存放)。如果使用 8 字节的登录名和 8 字节的组名，则需较多的磁盘空间。但是对于用户而言，使用名字比使用数值方便，所以口令文件包含了登录名和用户 ID 之间的映射关系，而组文件则包含了组名和组 ID 之间的映射关系。例如 ls -l 命令使用口令文件将数值用户 ID 映射为登录名，从而打印文件所有者的登录名。

1.3 文件和文件系统

Linux 文件系统是目录和文件的一种层次安排，目录的起点称为根(root)，其名字是一个字符“/”。目录(directory)是一个包含目录项的文件，在逻辑上，可以认为每个目录项都包含一个文件名，同时还包含说明该文件属性的信息。文件属性有文件类型、文件长度、文件所有者、文件的许可权(例如其他用户能否访问该文件)、文件最后的修改时间等。stat 和 fstat 函数返回一个包含所有文件属性的信息结构。第 3 章将详细说明文件的各种属性。

目录中的各个名字称为文件名(filename)。不能出现在文件名中的字符只有两个，斜线(/)和空操作符(null)。斜线分隔构成路径名的各文件名，空操作符则终止一个路径名。尽管如此，好的习惯是只使用印刷字符的一个子集作为文件名字符(只使用子集的理由是：如果在文件名中使用了某些 shell 特殊字符，则必须使用 shell 的引号机制来引用文件名)。

当创建一个新目录时，自动创建了两个文件名：.(称为点)和..(称为点-点)。点引用当前目录，点-点则引用父目录。在最高层次的根目录中，点-点与点相同。

零个或多个以斜线分隔的文件名序列(可以任选地以斜线开头)构成路径名(pathname),以斜线开头的路径名称为绝对路径名(absolute pathname),否则称为相对路径名。

每个进程都有一个工作目录(working directory),有时称为当前工作目录(current working directory)。所有相对路径名都从工作目录开始解释。进程可以用 chdir 函数更改其工作目录。

例如,相对路径名 doc/ memo/ joe 指的是文件 joe,它在目录 memo 中,而 memo 又在目录 doc 中,doc 则应是工作目录中的一个目录项。从该路径名可以看出,doc 和 memo 都应当是目录,但是却不清楚 joe 是文件还是目录。路径名/ urs/ lib/ lint 是一个绝对路径名,它指的是文件(或目录)lint,而 lint 在目录 lib 中,lib 则在目录 usr 中,usr 则在根目录中。登录时,工作目录设置为起始目录(home directory),该起始目录从口令文件中的登录项中取得。

文件描述符是一个小的非负整数,内核用以标识一个特定进程正在访问的文件。当内核打开一个现存文件或创建一个新文件时,它就返回一个文件描述符。当读、写文件时,就可使用它来指定需读写的文件。

按惯例,每当运行一个新程序时,所有的 shell 都为其打开三个文件描述符:标准输入、标准输出以及标准出错。如果像简单命令 ls 那样没有做什么特殊处理,则这三个描述符都连向终端。大多数 shell 都提供一种方法,使任何一个或所有这三个描述符都能重新定向到某一个文件,例如:

```
ls > file.list
```

执行 ls 命令,其标准输出重新定向到名为 file. list 的文件上。

标准 I/O 函数提供一种带缓存的界面。使用标准 I/O 可无需担心如何选取最佳的缓存长度。另一个使用标准 I/O 函数的优点与处理输入行有关(常常发生在 Linux 的应用中)。例如,fgets 函数读完整的一行,而 read 函数读指定字节数。

我们最熟悉的标准 I/O 函数是 printf。在调用 printf 的程序中,总是包括 < stdio. h > ,因为此头文件包括了所有标准 I/O 函数的原型。

1.4 程序与进程

程序(program)是存放在磁盘中的可执行文件。使用 6 个 exec 函数中的一个由内核将程序读入存储器,并使其执行。程序的执行实例被称为进程(process)——本书的每一页几乎都会使用这一术语。某些操作系统用任务表示正被执行的程序。每个 Linux 进程都一定有一个唯一的数字标识符,称为进程 ID(process ID)。进程 ID 总是一非负整数。

用于进程控制的函数主要是如下三个:fork、exec 和 waitpid(exec 函数有六种变体,但经常把它们统称为 exec 函数)。关于进程控制详见本书第 5 章。

头文件 < unistd. h > 包含了许多 Linux 系统服务的函数原型,例如 read,write 函数。函

数原型是 ANSI C 标准的组成部分。这些函数原型如下列形式：

```
ssize_t  read(int, void *, size_t);
ssize_t  write(int, const void *, size_t);
pid_t    getpid(void);
```

最后一个的意思是：getpid 没有参数(void)，返回值的数据类型为 pid_t。提供了这些函数原型后，编译程序在编译时就可以检查在调用函数时是否使用了正确的参数。另外，因为编译程序知道参数的数据类型，如果必要的话，它就会将参数强制转换成所需的数据类型。

前面所示的 getpid 函数的原型定义了其返回值为 pid_t 类型，这也是 POSIX 中的新规定。UNIX 的早期版本规定此函数返回一整型。与此类似，read 和 write 返回类型为 ssize_t 的值，并要求第三个参数的类型是 ssize_t。

以_t 结尾的这些数据类型被称为原始系统数据类型。它们通常在头文件 <sys/ types. h > 中定义(头文件 < unistd. h > 应已包括该头文件)。它们通常以 C typedef 说明加以定义。typedef 说明在 C 语言中已超过 35 年了(并不局限于 ANSI C 标准)，它们的目的是阻止程序使用专门的数据类型(例如 int, short 或 long) 来允许对于一种特定系统的每个实现选择所要求的数据类型。如，在需要存储进程 ID 的地方，分配类型为 pid_t 的一个变量。在各种不同的实现中，这种数据类型的定义可能是不同的，但是这种差别现在只出现在一个头文件中。我们只需在另一个系统上重新编译应用程序。

当 Linux 函数出错时，通常返回一个负值，而且整型变量 error 通常设置为具有特定信息的一个值。例如，open 函数如成功执行则返回一个非负值文件描述符，如出错则返回 -1。在 open 出错时，有大约 15 种不同的 errno 值(如文件不存在，许可权问题等)。某些函数并不返回负值而是使用另一种约定。例如，返回一个指向对象的指针的函数，在出错时，大多数将返回一个 null 指针。文件 < errno. h > 中定义了变量 errno 以及可以赋予它的各种常数。这些常数都以 E 开头，另外，Linux 手册第 2 部分的第 1 页，intro(2)列出了所有这些出错常数。例如，若 errno 等于常数 EACCES，这表示产生了权限问题 (例如，没有打开所要求文件的权限)。POSIX 定义 errno 为：

```
extern int errno;
```

POSIX. 1 中 errno 的定义较 C 标准中的定义更为苛刻。C 标准允许 errno 是一个宏，它扩充成可修改的整型左值 (lvalue) (例如返回一个指向出错数的指针的函数)。对于 errno 应当有两条规则：第一条规则是，如果没有出错，则其值不会被一个例程清除。因此，仅当函数的返回值指明出错时，才检验其值。第二条是，任一函数都不会将 errno 值设置为 0，在 < errno. h > 中定义的所有常数都不为 0。

信号是通知进程已发生某种条件的一种技术。例如，若某一进程执行除法操作，其除数为 0，则将名为 SIGFPE 的信号发送给该进程。进程如何处理信号有三种选择：

(1) 忽略该信号　有些信号表示硬件异常，例如，除以 0 或访问进程地址空间以外的单元等，因为这些异常产生的后果不确定，所以不推荐使用这种处理方式。

(2) 按系统默认方式处理　对于 0 除，系统默认方式是终止该进程。

(3) 提供一个函数，信号发生时则调用该函数　使用这种方式，我们将能知道什么时候产生了信号，并按所希望的方式处理它。

产生信号的方式有很多种，键盘方式有两种：分别称为中断键 (interrupt key，通常是