LSB 简介

Unix/Linux 标准化历史

标准化目前已经成为 Linux 系统上的一个热门话题。实际上，在 Linux 诞生之初，这个问题就得到了重视。当 Linus 在开发 0.01 版本的 Linux 内核时，就开始关注 POSIX 标准的发展，他在 /include/unistd.h 文件中定义了几个与 POSIX 有关的宏，以下内容就节选自 0.01 版本内核的 /include/unistd.h 文件：

```
/* ok, this may be a joke, but I'm working on it */
#define _POSIX_VERSION 198808L
```

下面我们就从 POSIX入手开始介绍 Unix/Linux 方面的标准化发展历程。

POSIX

Unix 1969 年诞生于 AT&T 贝尔实验室，并在 1973 年使用 C 语言进行了重写，从此就具有了很好的可移植性。但是当 AT&T 在 1984 年由于分拆而得以进入计算机领域的市场之后，却引发了 Unix 业界的一场大战。当时最为主要的两个版本是 AT&T 的 System V 和伯克利的 BSD。二者在技术方面（例如终端）和文化方面都存在很多分歧，导致应用程序很难在不同的系统上平滑地进行移植，为了解决这个问题，IEEE（Institute of Electrical and Electronic Engineers）的 1003 委员会着手开发了一系列标准，这就是后来的 POSIX（Portable Operating System Interface for UNIX）标准。其目的是为那些兼容各种 UNIX 变种的应用程序制定应用程序编程接口（API）规范，从而确保这些应用程序的兼容性。这些标准后来被 ISO/IEC 采纳，成为 ISO/IEC 9945 标准。

POSIX 在 15 份不同的文档中对操作系统与用户软件的接口进行了规范，主要内容包括3个部分：

- POSIX 系统调用
- POSIX 命令和工具
- POSIX 兼容测试

同时还提供了一套 POSIX 兼容性测试工具，称为 PCTS（POSIX Conformance

Test Suite）。

后来 POSIX 标准又进行了很多扩充，主要包括：

- POSIX.1，核心服务：主要集成了 ANSI C 标准，包括进程创建和控制、信号、浮点异常、段错误、非法指令、总线错误、定时器、文件和目录操作、管道、C 标准库、I/O 端口和控制
- POSIX.1b，实时扩展：包括优先级调度、实时信号、时钟和定时器、信号量、消息传递、共享内存、异步和同步 I/O、内存锁
- POSIX.1c，线程扩展：包括线程创建和控制、线程调度、线程同步、信号处理

POSIX 最初的设计目标是为 Unix System V 和 BSD Unix 等 Unix 系统上的特性制定规范，使其可以实现更好的可移植性。但是很多其他系统都也兼容POSIX 标准。例如，微软的 Windows NT 就兼容 POSIX 标准的实时部分（POSIX.1b），而 RTOS（LynxOS real-time operating system）也与 POSIX 标准兼容。Windows 上可以通过安装 Windows 的 Services for UNIX 或 Cygwin 来增强对 POSIX 标准的兼容度。

Open Group

Open Group 是现在 Unix 商标的拥有者，其前身是 X/Open。X/Open 是 Unix 厂商在 1984 年成立的一个联盟，它试图为众多 Unix 变种定义一个安全公共子集，因此即使在 Unix 混战的年代，也得到了比较好的发展。在 1993 年，包括主要 Unix 公司在内的75 家系统和软件供应商委托 X/Open 为 Unix 制定一个统一的规范。X/Open在现有标准基础上，增加了对终端进行处理的 API 和 X11 API，并全面兼容 1989 ANSI C 标准，最终诞生了第一版本的单一 Unix规范（Single Unix Specification，简称 SUS）。

X/Open在 1996 年与 OSF（开放软件基金会）进行合并，成立了 Open Group 组织，专门从事开放标准的制定和推广工作，并对很多领域提供了认证，包括 Unix 操作系统、Motif 和 CDE（Common Desktop Environment）用户界面。

Austin Group

Austin Group 是在 1998 年成立的一个合作技术工作组，其使命是开发并维护 POSIX.1 和 SUS 规范。Austin Group 开发规范的方法是"write once, adopt everywhere"，即由 Austin Group 制定的规范既会成为 IEEE POSIX 规范，又会成为 Open Group 的 技术标准规范，以后又会被采纳为 ISO/IEC 的标准。新开发的规范后来就被标准化为 ISO/IEC 9945 和IEEE Std 1003.1，并成为 SUSV3 的核心

部分。

这种独特的开发模式最大限度地利用了业界领先的工作成果，将正式的标准化工作转化成了一个唯一的行为，并且吸引了广泛的参与者。Austin Group 目前有 500 多个参与者，工作组的主席是 Open Group 的 Andrew Josey。

在90 年代中期，Linux 也开始了自己的标准化努力。实际上，Linux 一直都试图遵守 POSIX 标准，因此在源代码级上具有很好的兼容性，然而对于 Linux 来说，仅仅保证源码级的兼容性还不能完全满足要求：在 Unix 时代，大部分系统都使用的是专有的硬件，软件开发商必须负责将自己的应用程序从一个平台移植到其他平台上；每个系统的生命周期也很长，软件开发商可以投入足够的资源为各个平台发布二进制文件。然而 Linux 使用的最广泛的 x86 通用平台，其发行版是如此众多，而发展却如此之快，软件开发商不可能为每个发行版都发布一个二进制文件，因此就为 Linux 上的标准化提出了一个新的要求：二进制兼容性，即二进制程序不需要重新编译，就可以在其他发行版上运行。

实际上，在 Linux 社区中第一个标准化努力是文件系统层次标准（Filesystem Hierarchy Standard，FHS），用来规范系统文件、工具和程序的存放位置和系统中的目录层次结构，例如 ifconfig 命令应该放在 /usr/bin 还是 /usr/sbin 目录中，光驱应该挂载到 /mnt/cdrom 中还是 /media/cdrom 中。这些需求最终共同促进了 Linux Standard Base（LSB）项目的诞生。

LSB目前是 FSG（Free Standards Group）中最为活跃的一个工作组，其使命是开发一系列标准来增强 Linux 发行版的兼容性，使各种软件可以很好地在兼容 LSB 标准的系统上运行，从而可以帮助软件供应商更好地在 Linux 系统上开发产品，或将已有的产品移植到 Linux 系统上。

LSB 以 POSIX 和 SUS 标准为基础，并对其他领域（例如图形）中源代码的一些标准进行了扩充，还增加了对二进制可执行文件格式规范的定义，从而试图确保 Linux 上应用程序源码和二进制文件的兼容性。

回页首

LSB 简介

LSB 是 Linux 标准化领域中事实上的标准，它的图标（请参看图 1）非常形象地阐述了自己的使命：对代表自由的企鹅（Linux）制定标准。给定企鹅的体形和三维标准之后，软件开发者就可以设计并裁减出各色花样的衣服（应用程序），这样不管穿在哪只企鹅身上，都会非常合身。

图1. LSB 图标

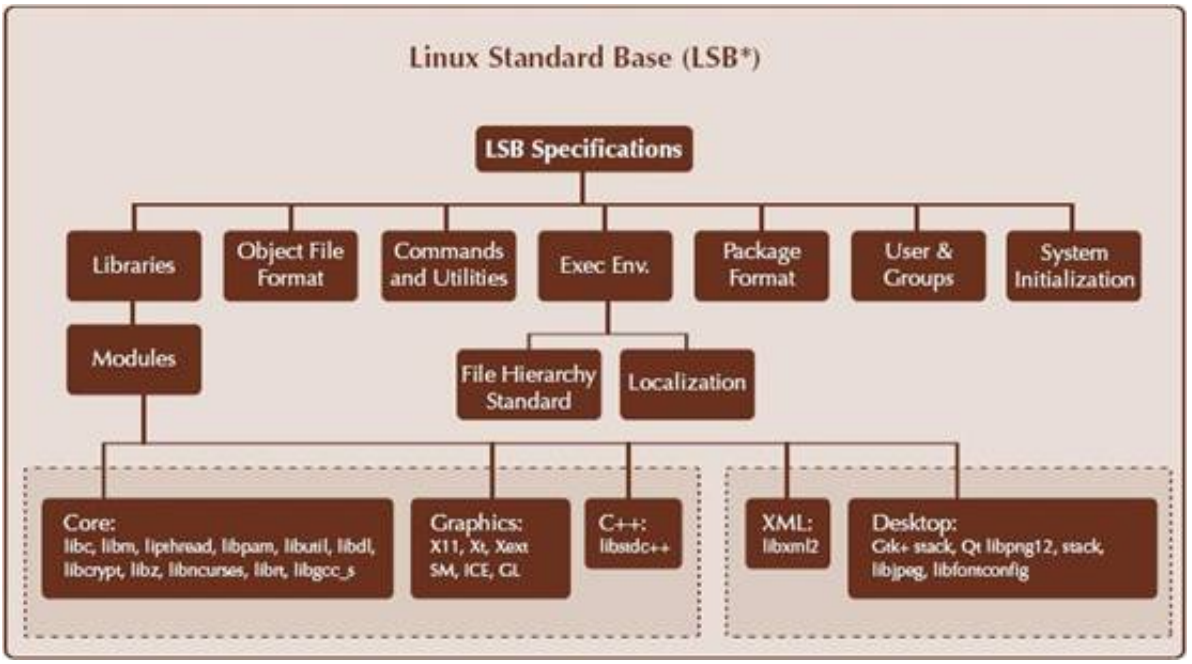
在现有标准基础上，LSB 制定了应用程序与运行环境之间的二进制接口，这主要是基于以下标准：

- Single UNIX Specification（SUS）
- System V Interface Definition（SVID）
- compilers for the Intel Itanium processor
- C++ ABI
- System V Application Binary Interface（ABI）

同时，LSB 充分吸取了 UNIX 标准化努力所取得经验和教训，回避了这些标准的一些问题。例如，POSIX 仅仅定义了编程接口的标准，但是它却无法保证二进制的兼容性。而诸如 OSF/1 之类的标准虽然试图解决二进制兼容性的问题，但是限制却太为严格。LSB 在二者之间达成了一个平衡，它包含了一个二进制兼容层，同时消除了 POSIX 与 OSF/1 之间存在分歧的地方。

LSB 对各个库提供的接口以及与每个接口相关的数据结构和常量进行了定义，图2 给出了 LSB 3.1 环境中所包含的组件。这些组件包括开发者所需要的共享库（包括 C++），文件系统层次结构（FHS）、对象文件格式、命令和工具、应用程序包、用户和组、系统初始化等所采用的规范：

图2. LSB 规范包含的组件



为了保证 LSB 项目的良好运行，LSB 采用了自己的完整组织架构来负责整个项目的运行，包括主席、选举委员会、执行委员会：

- 主席：由选举委员会选举产生，任期两年，负责 LSB 项目的整体运作，并对 FSG 和社区代表 LSB项目。目前的主席是 Debian 的创始人 Ian Murdock.。
- 选举委员会：由所有对 LSB 作出贡献的人组成，负责在主席任期期满时选举下一任主席。
- 执行委员会：由主席和各个子项目的领导人以及对 LSB
- 项目有重大贡献的人组成，可以对外作为 LSB 项目的发言人。

LSB 项目包含几个子项目（也称为工作组），分别负责不同的职责范围，简介如下：

- Specification SubGroup：负责开 LSB 规范的开发与维护，还要负责 ISO/IEC 23360（即 ISO LSB 标准的维护）。具体职责如下：
  - 维护 LSB 规范数据库
  - 编写 LSB 规范
  - 开发并维护生成规范文档所需要的工具

- LSB Tools SubGroup：负责以下子项目的开发和实现：
  - SI（Sample Implementation）：遵守 LSB 规范的一个参考实现
  - Development Environment：开发符合 LSB 规范的应用程序的开发环境
  - Application Battery：符合 LSB 规范的样例应用程序，例如 lsb-apache

- LSB Test SubGroup：负责按照 LSB 规范的定义，开发一些测试套件，来验证用户环境和应用程序是否符合 LSB 规范，主要包括：
  - LSB Runtime Tests：包括ANSI、POSIX、LSB-OS、线程、用户和组、FHS、国际化、PAM（可插入认证模块）等测试套件

    - LSB VSW4/XTS5 Test：Xlib11及其扩展库的测试套件
    - LSB C++ Test：C++ 测试套件

- LSB Desktop SubGroup：负责开发与桌面有关的规范的测试套件，用来验证用户环境和应用程序是否符合 LSB 规范，主要包括：
    - OpenGL 库
    - PNG12库
    - JPEG 库
    - Fontconfig 库
    - GTK+ Stack 库
    - QT3/4 库
    - XML2 库

- LSB Future SubGroup：负载开拓 LSB 的新领域，将已经发展比较成熟可以进行标准化但 LSB 尚未涉及的领域纳入 LSB 标准范围内

LSB 的标准化流程

LSB 对于标准的制定和推广遵循务实的原则，它自己不会自行制定标准然后强行要求业界接受，而是把业界中已经成熟的技术和规范采用标准化的形式固定下来，然后大力加以推广，这样可以更广泛地为软件供应商和用户接受。

一个新领域要想纳入 LSB 标准的范畴，必须经过以下 3 个步骤：

1. 鉴定：确认这个领域是否已经足够成熟，是否具有稳定的 ABI/API，是否需要进行标准化，以及是否依赖于尚未标准化的领域。

2. 调研：调查上游软件维护者是否还在积极维护，软件是否稳定，是否具有很好的向后兼容性。

3. 实现：将该领域加入 LSB 数据库、编写规范、编写测试套件、并将其加入开发环境、SI 和 APPBAT。

经过这 3 个步骤之后，LSB Future SubGroup 就会将其提交给 LSB 工作组，将其包含到 LSB 的下一个版本中进行发布，并对外提供认证服务。

在制定好标准并开发出测试套件之后，为了区分系统或应用程序是否兼容 LSB 标准， FSG 提供了 LSB 标准认证服务。任何 Linux 发行版厂商和应用程序开发商都可以进行 Linux 认证，目前提供的认证有两种：

- LSB 运行环境：为平台供应商提供的 LSB 标准认证
- LSB 应用程序：为应用程序开发商提供的 LSB 标准认证

对于平台供应商来说，经过 LSB 认证之后，就可以确保自己的系统所提供的服务都是标准的，任何遵守 LSB 标准的应用程序都可以很好地在自己的系统上运行；而对于应用程序开发商来说，其意义则刚好相反：不需要担心自己的应用程序在遵守 LSB 标准的系统上的可移植性问题。

LSB 认证过程包含以下步骤：

- 注册：要进行 LSB 认证的第一个步骤是在 https://www.freestandards.org/index.php?title=Special:Userlogin [1] 上先创建一个帐号，并注册您的公司和产品。
- 测试和验证：使用 LSB 提供的测试工具，在您的测试系统上运行，并对结果进行分析（详细内容请参看本系列的下一篇文章），确保您的系统或应用程序遵守 LSB 规范。
- 最终审计：在准备好正式提交测试结果之后，需要先签署 LSB 认证协议和 LSB 商标许可协议，并向 FSG 支付认证所需要的费用。然后 FSG 会有专人对测试结果进行审计，如果一切正常，就通过了 LSB 认证。通过 LSB 认证的产品都会在 http://www.freestandards.org/en/Products 公开发布。目前已经 Redhat、Suse、RedFlag 等公司都已经通过了 LSB 3.0 的认证，对于 LSB 3.1 的认证正在进展中。

通过 LSB 认证之后，所认证的产品可以贴上 "LSB Certified" 的标签进行销售了。

认证问题报告

在运行 LSB 所提供的测试工具时可能会出现部分测试用例失败的情况，其原因可能是产品本身的问题，例如 FHS 标准要求系统中必须存在 /media/ 目录，而在某些系统中，这个目录可能并不存在，此时就可能会导致相应的测试套件失败，错误信息如下：

```
10|715 /tset/LSB.fhs/root/media/media-tc 00:55:04|TC Start, scenario ref 720-0
15|715 3.7 5|TCM Start
400|715 1 1 00:55:04|IC Start
200|715 1 00:55:04|TP Start
520|715 1 30056 1 1|Reference 3.11-1(A)
520|715 1 30056 1 2|The /media directory exists and is searchable
520|715 1 30056 1 3|/media: directory not found
520|715 1 30056 1 4|exit code 1 returned, expected 0
```

但是有时可能并非是测试环境的问题，而是测试套件本身的问题，或者是由于系统中存在一个公认但却暂时无法修复的问题，此时并不影响 LSB 的认证的结果。如果出现这种问题，测试人员可以将这个问题反馈给 LSB（http://www.freestandards.org/cert/prsubmit.php），经过确认之后，LSB 会在一个 waiver 文件中列出这种情况，并将对应的测试套件暂时剔除，并尝试在下一个版本中进行修复。我们也可以从 http://www.freestandards.org/cert/pr.php [2] 上查看已经发现的问题。

回页首

LSB 的历史、现状和将来

LSB 项目最初发起于 1998 年 5 月，其项目启动宣言得到了 Linus Torvalds、Bruce Perens、Eric Raymond 等人的签名支持，当时的目标是建立一系列构建 Linux 发行版所采用的源代码应该遵循的标准，并提供一个参考平台。2000 年 5 月，LSB 成为 Free Standards Group（FSG） 的一个工作组。FSG 是一个独立的非盈利组织，专注于通过开发和促进标准来加速开源软件的发展。

从 2001 年 6 月发布第一个正式版本的规范以后，LSB 规范几乎每 6 个月都会进行一次更新。截止到 2005 年 7 月发布的 3.0 版本为止，LSB 重点关注的是服务器端的使用，这与 Linux 在服务器端得到了广泛的应用是一致的。这个规范已经被 ISO 采纳为国际标准 23360。

目前最新的版本规范是 2005 年 10 月发布的 LSB 3.1，目前它可以支持 7 种体系结构：

- X86_64
- PPC32
- PPC64
- S390x

由于平台的差异，所有的规范除了有一个通用的版本之外，还都存在一个适用于特定平台的版本，其中的内容是完全适用于这个平台的。

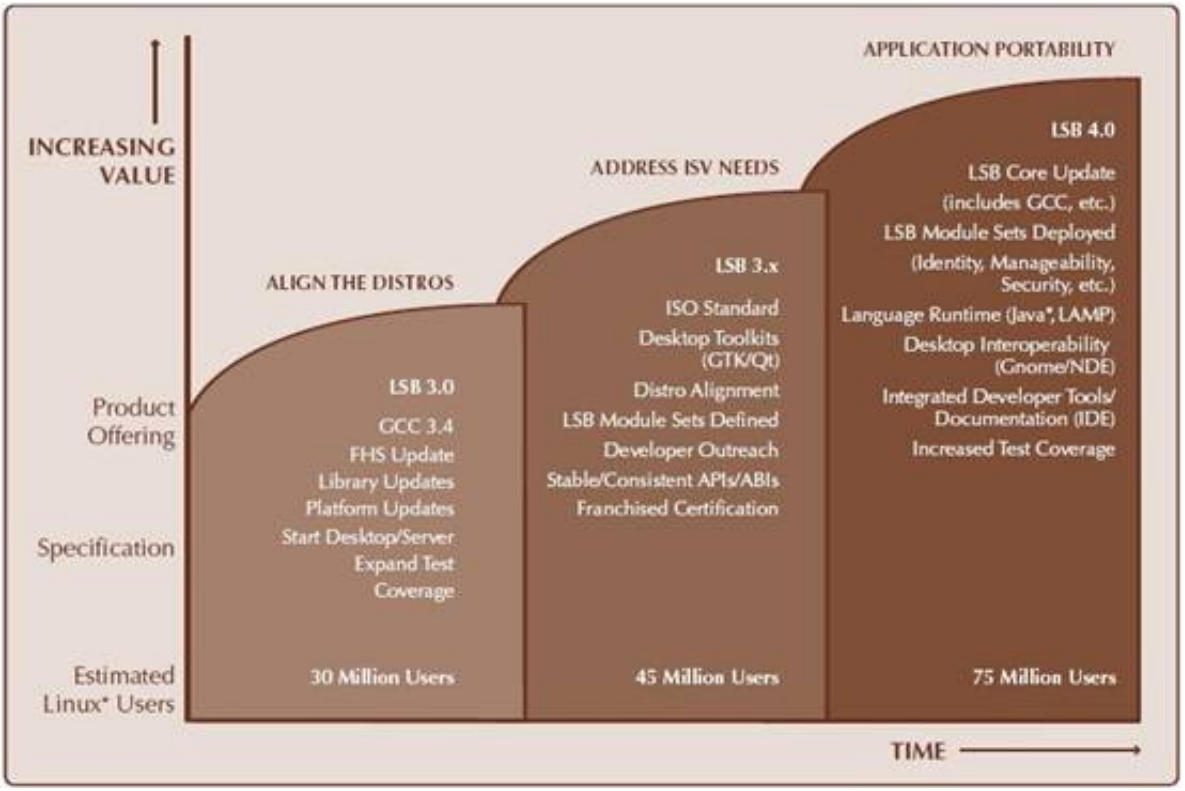
与上一个版本相比，LSB 3.1 版本的规范主要是增强了对桌面系统的标准化支持，增加了对 GTK 和 QT GUI 工具包的标准化。另外，LSB 还调整了自己的路线图，以便可以与主流的 Linux 发行商（Redhat、Novell、Asianux、Debian 等）的发行计划更好地吻合，并吸引了更多 Linux 发行商的参与，对开发工具和文档进行

了改进，还与各个国家的组织（例如中国电子技术标准化研究所，CESI）进行认证方面的合作。

下一个版本 LSB 3.2 将在 2007 年第二季度发布，将主要增加 freedesktop.org 的标准和跨桌面的交互操作性。

LSB 4.0 将在 2008 年发布，它将实现更好的二进制兼容性，并增加对 Perl、Python、LAMP、Java 等语言的标准化支持。详细路线图及各个主要版本的特性如图 3 所示：

图3. LSB 主要版本的路线图



回页首

实例：lsb_release 的规范定义和实现

我们知道，在 /etc 目录中有一个文件可以查看当前系统的版本信息，在 RHEL4U3（Red Hat Enterprise Linux 4 Update 3）上这个文件是 /etc/redhat-release：

```
# cat /etc/redhat-release
Red Hat Enterprise Linux ES release 4 (Nahant Update 3)
```

在 SLES9SP3（SUSE LINUX Enterprise Server 9 Service Pack 3）上这个文件是 /etc/SuSE-release ：

```
# cat /etc/SuSE-release
SUSE LINUX Enterprise Server 9 (i586)
VERSION = 9
PATCHLEVEL = 3
```

我们可以看出，在这两个发行版上，不但使用的文件不同，文件的内容和格式也完全不同。如果开发人员在自己的程序中使用这些信息，他们就很难使用一个统一的接口来获取发行版本的信息，因此必须为每种平台都定制一个脚本或开发一个程序才能实现这种功能，这无疑会增加很多工作量，而且所生成的程序的可移植性也会很差。

为了解决这个问题，LSB 规范中增加了对 lsb_release 接口及其输出格式的定义：lsb_release 的功能是打印与发行版本相关的信息，必须实现以下选项（详细规范的定义请参考http://refspecs.freestandards.org/LSB_3.1.0/LSB-Core-generic/LSB-Core-generic/lsbrelease.html ）：

表1. LSB对lsb_release 接口定义

选项	功能	输出格式
-v, --version	显示发行版所兼容的 LSB 规范版本号	LSB Version:\tLSBSpecVersion（其中多个 LSBSpecVersion以冒号分隔）
-i, --id	显示发行版的字符串 id 信息	Distributor ID:\tDistributorID
-d, --description	显示发行版的单行文本描述	Description:\tDescription
-r, --release	显示发行版的版本号	Release:\tRelease
-c, --codename	显示该发行版本的代码号	Codename:\tCodename
-a, --all	显示以上所有信息	各个选项遵守以上规定
-s, --short	以简短形式显示以上信息	文本，格式无具体要求
-h, --help	显示帮助信息	文本，格式无具体要求

我们前面已经介绍过，通过 LSB 认证发行版或软件可以得到 FSG/LSB 的授权，贴上 "LSB Certified"的标签进行销售。实际上，在通过 LSB 认证的系统上，我们可以看到一个与 LSB 有关的包，其中包含了 LSB 规范中对 lsb_release 接口规范的实现。lsb_release 是 LSB-Core-generic 规范中要求的一个接口，各种平台（目前可以支持的 7 种平台： IA32、IA64、X86_64、PPC32、PPC64、S390 和 S390x）上都应该提供这个接口的实现。在 RHEL4U3 上，我们可以找到一个名为 redhat-lsb 的包（RHEL4U3 遵守的是 LSB 3.0 版本的规范，因此这个包的版本是 redhat-lsb-3.0-8.EL），该包的内容如下：

```
# rpm -ql redhat-lsb
/bin/mailx
/etc/lsb-release.d
/etc/lsb-release.d/core-3.0-ia32
/etc/lsb-release.d/core-3.0-noarch
/etc/lsb-release.d/graphics-3.0-ia32
/etc/lsb-release.d/graphics-3.0-noarch
/etc/redhat-lsb
/etc/redhat-lsb/lsb_killproc
/etc/redhat-lsb/lsb_log_message
/etc/redhat-lsb/lsb_pidofproc
/etc/redhat-lsb/lsb_start_daemon
/lib/ld-lsb.so.3
/lib/lsb
/lib/lsb/init-functions
/usr/bin/lsb_release
/usr/lib/lsb
/usr/lib/lsb/install_initd
/usr/lib/lsb/remove_initd
/usr/sbin/redhat_lsb_trigger.i386
/usr/share/man/man1/lsb_release.1.gz
```

而在 SLES9SP3 中，这个包的名字是 lsb（lsb-3.0-4.8），它包含的内容如下：

```
# rpm -ql lsb
/etc/lsb-release
/etc/lsb-release.d
/etc/lsb-release.d/graphics-2.0-ia32
/etc/lsb-release.d/graphics-2.0-noarch
/etc/lsb-release.d/graphics-3.0-ia32
/etc/lsb-release.d/graphics-3.0-noarch
/lib/ld-lsb.so.2
/lib/ld-lsb.so.3
/usr/bin/lsb_release
/usr/share/man/man1/lsb_release.1.gz
```

我们可以看出，在这两个发行版上的两个包中存在一些共同的文件：

- **/usr/bin/lsb_release**
- **/lib/ld-lsb.so.3**

其中 /lib/ld-lsb.so.3 在两个系统上都是一个符号链接：

```
# ls -l /lib/ld-lsb.so.3
lrwxrwxrwx  1 root root 13 Apr 20 03:04 /lib/ld-lsb.so.3 -> ld-linux.so.2
```

系统中提供的大部分应用程序都是链接到了 ld-linux.so.2 上，但是兼容 LSB 标准的应用程序都可以链接到 ld-lsb.so.3 上（由于 SLES9SP3 还兼容 LSB 2.0 的规范，因此系统中还存在一个 /lib/ld-lsb.so.2 库，也是指向 ld-linux.so.2 的符号链接；而 RHEL4U3 不兼容 LSB 2.0 的规范，因此没有这个库）。

而 /usr/bin/lsb_release 就是对 lsb_release 接口的具体实现，在这两个系统上都是一个 shell 脚本。下面我们分别在这两个系统上执行 lsb_release -a 命令，在 RHEL4U3 上的结果如下：

```
# /usr/bin/lsb_release -a
LSB Version:
:core-3.0-ia32:core-3.0-noarch:graphics-3.0-ia32:graphics-3.0-noarch:log
Distributor ID: RedHatEnterpriseES
Description:    Red Hat Enterprise Linux ES release 4 (Nahant Update 3)
Release:        4
Codename:       NahantUpdate3
```

在 SLES9SP3 上的结果如下：

```
# /usr/bin/lsb_release -a
LSB Version:
core-2.0-noarch:core-3.0-noarch:core-2.0-ia32:core-3.0-ia32:graphics-2.0-ia32:
graphics-2.0-noarch:graphics-3.0-ia32:graphics-3.0-noarch
Distributor ID: SUSE LINUX
Description:    SUSE LINUX Enterprise Server 9 (i586)
Release:        9
Codename:       n/a
```

我们可以看出，在这两个系统上，lsb_release 命令的位置、用法、输出格式都是相同的，因此开发人员可以使用它编写兼容性非常好的程序。

再看一下在这两个系统上与 lsb_release 有关的包，我们会发现二者之间有一些区别，例如在 SLES9SP3 上存在一个 /etc/lsb-release 文件，其中存放的是所遵循的 LSB 标准的版本，内容如下：

```
# cat /etc/lsb-release
LSB_VERSION="core-2.0-noarch:core-3.0-noarch:core-2.0-ia32:core-3.0-ia32"
```

而在 RHEL4U3 上并没有这个文件，但是在 /etc/lsb-release.d 目录中的文件却比 SLES9SP3 上多：

- **/etc/lsb-release.d/core-3.0-ia32**
- **/etc/lsb-release.d/core-3.0-noarch**
- **/etc/lsb-release.d/graphics-3.0-ia32**
- **/etc/lsb-release.d/graphics-3.0-noarch**

而在 SLES9SP3 上只有以 graphics 开头的文件。仔细查看一下 /usr/bin/lsb_release的实现我们就会发现，所遵守的 LSB 规范的列表既可以保存在 /etc/lsb-release 文件中，也可以以文件的形式放到 /etc/lsb-release.d 目录中。因此 LSB 只是对接口定义进行了规范，但却没有限定具体的实现，这样既可以为发行版供应商提供充分的自由，又为应用程序开发人员提供了一致的接口，可以得到最大限度的推广和应用。

回页首

标准化的 Linux 操作系统可以为应用程序开发者提供一个开发应用程序的良好平台，使他们开发的应用程序可以非常平滑地移植到其他发行版本上。LSB 通过定义一系列规范，并提供标准测试套件和开发环境，可以帮助开发人员更容易地开发遵守规范的应用程序，辅助供应商构建更标准的系统。在本系列的下一篇文章中，我们将介绍如何使用 LSB 标准提供的测试工具来验证系统和应用程序是否遵守 LSB 规范。

1. https://www.freestandards.org/index.php?title=Special:Userlogin
2. http://www.freestandards.org/cert/pr.php