Cut the Cost of Porting: Linux Standard Base Compliant Applications

Al Stone

Hewlett-Packard Company

ahs3@fc.hp.com

HP WORLD 2002
Conference & Expo

What is the LSB?

- A working group of the Free Standards Group...
  - http://www.freestandards.org
  - http://www.linuxbase.org
- A Family of Specifications....
  - SUS, SVID, FHS, X 11, OGL, ISO...
  - It documents what exists, does not create new standards
- What does it do?
  - The LSB defines a binary interface for application programs that are compiled and packaged for LSB-conforming implementations
- Why do I care?
  - Minimizes porting costs between distributions
  - Makes it possible to easily support multiple distributions
  - Develop once, deploy in many environments

Where Are We Today?

- Specification Versions:
  - LSB 1.1, January 2002
  - Includes lsbappchk and other tools
  - LSB 1.2, June 2002
  - Architecture-specific addenda for IA-32, PPC, s390 and IA-64
- LSB Futures
  - Adds breadth
  - Compiling significant additions
  - Candidates at http://www.linuxbase.org/futures/candidates
- Certification Process
  - Defined for distros and applications
  - Pilot program under way

What is in the LSB Spec?

- Two flavors:
  - gLSB: general, architecture-neutral LSB specification
  - archLSB-machine: architecture-dependent LSB specification
    - archLSB-IA32, archLSB-PPC32, and archLSB-IA64 available
- The sections:
  - I: intro, related standards, terminology, and such
  - II: object file formats (ELF), sections needed, symbol mapping and versioning for C/C++
  - III: dynamic linking
  - IV: base libs (libc, libm, libpthread, libdl, libcrypt, librt)
  - V: utility libs (libz, libncurse, libutil)
  - VI: graphics libs (libX11, libXext, libSM, libICE, libXt, libGL)
  - VII: package format (RPM v3) and installation
  - VIII: commands, utilities, and their options
  - IX: standard shell (bash)
  - X: users and groups
  - XI: file system layout (FHS 2.2)
  - XII: cron jobs, init scripts, conventions in init scripts

Mechanics

- LSB Environment:
  - Header files
  - Stub libraries
- Chroot Build Environment
  - requires sshd
  - requires 2.4.x kernel
  - exported users need local home directories
- lsbcc wrapper script
- LSB Test Suite
  - for distros
  - lsbappchk

Configuring Debian

- Update:
  - `apt-get update`
- Kernel:
  - `apt-get install kernel-image-2.4.18-686`
- Dependencies:
  - `apt-get install libz2-1.0 libc6 libpopt0 zlib1g`
- LSB Dev Applications:
  - ftp://ftp.freestandards.org/pub/lsb/lsbdev
  - `dpkg -i lsb-rpm-4.0.3-1.0.3_i386.deb`
  - `dpkg -i lsbdev-base_1.2.2_i386.deb`
  - `dpkg -i lsbdev-chroot_1.2.2_i386.deb`
- Edit /etc/lsbdev files
- Install lsbappchk
  - ftp://ftp.freestandards.org/pub/lsb/test_suites/released-1.1.0/binary/application
  - `alien -k lsbappchk-1.2.2-1.i386.rpm`
  - `dpkg -i lsbappchk_1.2.2-1_i386.deb`

Configuring Red Hat

- Kernel: make sure you're using 2.4.x kernel (7.1/7.2/7.3)
- Dependencies:
  - install binutils, glibc-devel, openssh-server
- Install dependencies for lsb-rpm:
  - `rpm -q gawk fileutils textutils mktemp shadow-utils`
- LSB Dev Applications:
  - ftp://ftp.freestandards.org/pub/lsb/lsbdev
  - `rpm -Uvh lsb-rpm-4.0.3-1.0.3.i386.rpm`
  - `rpm -Uvh lsbdev-base-1.2.2.i386.rpm`
  - `rpm -Uvh lsbdev-chroot-1.2.2.i386.rpm`
- Edit /etc/lsbdev files
- Install lsbappchk
  - ftp://ftp.freestandards.org/pub/lsb/test_suites/released-1.1.0/binary/application
  - `rpm -Uvh lsbappchk-1.2.2-1.i386.rpm`

Run the lsbdev Environment

- Start up the chroot environment
  - `/etc/init.d/lsbdev { start | stop }`
- Login to the environment:
  - `/usr/bin/slogin -p 5436 localuser@localhost`

Simple LSB-Compliant App

- Simple 'hello world':

```
#include <stdio.h>
int main(int argc, char *argv[]) {
   printf (“hello world\n”);
   return 0;
}
```

- Compile it:

  -
    ```
    gcc -o hw hello.c -L/usr/lib/lsb -
    I/usr/include/lsb -Wl,--dynamic-linker=/lib/ld-
    lsb.so.1
    ```
- Run it:

```
$ ./hw
hello world
```

- Test it:

```
$ lsbappchk hw
```

Simple Non-LSB-Compliant Application

- Simple domain name:

```
#include <stdio.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
   char domain[BUFSIZ];
   int rc;
   rc = call_my_non_lsb_getdomainname((char *)&domain, BUFSIZ);
   if (rc == 0) {
      printf (“domain is: %s\n”, domain);
   }
   return rc;
}
```

- Compile it:
  - `lsbcc -o dn dn.c`
  - What happens?
- Test it:

```
$ lsbappchk dn
call_my_non_lsb_getdomainname
```

Complex LSB-Compliant Application

- We'll use mutt
- Cd someplace useful, get the source

```
$ cd mutt
$ rsync -v ftp.mutt.org::mutt/mutt-1.3.27i.tar.gz .
$ tar xvzf mutt-1.3.27i.tar.gz
$ ftp ftp://space.mit.edu/pub/davis/slang/slang-1.4.5.tar.gz
$ tar xvzf slang-1.4.5.tar.gz
```

- Build slang:

```
$ CC=lsbcc ./configure -prefix=`pwd`/install
$ make
$ make runtests
$ make install
```

- Build mutt

```
$ echo “#undef SYS_SIGLIST_DECLARED” >> config.h
$ CC=lsbcc ./configure –with-mailpath=/var/mail \
  –with-slang=../slang-1.4.5/install
$ make
```

- Check the results:

```
$ ldd mutt
$ lsbappchk mutt
```

Cut the Cost of Porting: LSB Compliant Applications

The End