



MOVING FROM SOLARIS TO RED HAT ENTERPRISE LINUX

A QUICK START GUIDE FOR APPLICATION DEVELOPERS

BY MICHAEL SOLBERG, SENIOR CONSULTANT, RED HAT CONSULTING

2 INTRODUCTION

2 THE SHELL ENVIRONMENT

2 ksh on Red Hat Enterprise Linux

2 Core Shell Utilities

3 Environment Variables

3 RPM PACKAGE MANAGER

3 Overview

4 Installing Software

4 Upgrading Software

5 Verifying Software

6 Building RPM Packages

6 Using yum and up2date

6 DEVELOPER TOOLS

6 Overview

7 The GNU Compiler Collection

7 The GNU Linker

8 FURTHER READING



INTRODUCTION

Red Hat® Enterprise Linux® is a standards-based UNIX®-like operating system that provides a development environment similar to Sun™ Solaris™. The shell and development tools are very familiar to anyone with experience working in UNIX environments. This paper looks to highlight some of the major differences between Solaris and Red Hat Enterprise Linux to help application developers port their code. It will cover differences in the shell environment, using the RPM package manager, and using the GNU development tools. This is not an exhaustive document; it's meant to get Solaris developers working on Red Hat Enterprise Linux quickly.

THE SHELL ENVIRONMENT

KSH ON RED HAT ENTERPRISE LINUX

As with most UNIX-like operating systems, the system shell on Red Hat Enterprise Linux is a version of the Bourne shell. Red Hat Enterprise Linux implements the GNU Bourne-Again shell, or Bash. Bash is compliant with IEEE Standard 1003.1 from the POSIX specification, and most scripts written for `/bin/sh` on Solaris should run fine under `/bin/sh` on Red Hat Enterprise Linux. Red Hat Enterprise Linux also ships the Korn shell, so porting ksh scripts from Solaris is relatively simple.

Solaris 10 ships with Korn shell version 88, whereas Red Hat Enterprise Linux ships version 93.¹ ksh-93 is the current version of AT&T Korn shell and adds support for associative arrays and floating point arithmetic. On the whole, ksh-93 is backwards compatible with ksh-88. Here are a few notable exceptions:

- Variables in functions declared with the `name()` syntax are no longer local.
- ksh-93 does not set the `ERRNO` environment variable.
- Testing for string equality via the `==` operator is supported, but obsolete. Instead, use `===`.
- The `-x` option to `alias` is no longer supported.
- The `-f` option to `typeset` is no longer supported.
- The output formats of some built-in functions including `set`, `typeset`, and `alias` has changed.

CORE SHELL UTILITIES

Most Solaris users will probably be accustomed to shell utilities being slightly different between various standards. Unlike Solaris, Red Hat Enterprise Linux does not ship different sets of utilities for differing standards (i.e. `/usr/xpg4/bin` on Solaris). Most of the Red Hat Enterprise Linux utilities are from the GNU project, and many of them favor BSD semantics (similar to the commands in `/usr/ucb` on Solaris). In addition, GNU utilities support long options as well as the standard "short" options in UNIX. These options are prepended with two dashes instead of one (i.e. `--help` instead of `-h`). Check the manual when in doubt about certain options.

¹ AT&T Korn shell is provided as a technology preview in Red Hat Enterprise Linux 4 and the default ksh in Red Hat Enterprise Linux 5. Red Hat Enterprise Linux 4 ships with Public Domain ksh by default. Public Domain ksh is not fully compatible with AT&T Korn shell. ksh-93 is available on Solaris under `/usr/dt/bin/ksh`.



Here are some examples of the differences between the two sets of utilities that need to be considered when porting shell scripts:

- GNU cat does not support the `-s` option.
- GNU sum uses the BSD method for computing checksums, not the System V (this behavior is available on Solaris using the `“-r”` option). For System V behavior, use the `“-s”` flag.
- GNU sort does not support the `“-y”`, `“-z”`, or the `“+”` options, which are listed as obsolete in Solaris 10.
- The default ordering of files in the output from `ls` differs on Red Hat Enterprise Linux.
- The `df` command on Red Hat Enterprise Linux behaves like the BSD `df (/usr/ucb/df)`.
- GNU echo behaves like the Berkeley echo (`/usr/ucb/echo`) and does not implement backslash escapes.
- GNU uname differs quite a bit from Solaris uname.
- The GNU install command takes different options that both `/usr/bin/install` and `/usr/ucb/install` on Solaris.

ENVIRONMENT VARIABLES

Certain environment variables that control the behavior of various system utilities differ between Red Hat Enterprise Linux and Solaris. Some variables used in Solaris (such as `NETPATH`, `MSGVERB`, and `SEV_LEVEL`) will have no effect on the environment in Red Hat Enterprise Linux.

Red Hat Enterprise Linux supports the POSIX internationalization locale standards, and the `LANG` and `LC_` environment variables are populated and respected in the shell environment. Calls to `setlocale()` should work as expected.

However, the XSI extension NLS variables (i.e. `NLS`, `NLSPATH`) are not set by default. Programmers should rely on the `LANG` environment variable for setting and retrieving the current locale in the shell environment. In addition to the standard `LC_` variables (`LC_CTYPE`, `LC_COLLATE`, `LC_MONETARY`, `LC_NUMERIC`, `LC_TIME`, and `LC_MESSAGES`), additional variables are available in Red Hat Enterprise Linux, including `LC_PAPER`, `LC_NAME`, `LC_ADDRESS`, `LC_TELEPHONE`, `LC_MEASUREMENT`, and `LC_IDENTIFICATION`.

The `TZ` environment variable is not set by default on Red Hat Enterprise Linux. It is, however, supported. Setting the `TZ` environment variable will override the timezone settings defined by `/etc/localtime`.

RPM PACKAGE MANAGER

OVERVIEW

Developers new to Red Hat Enterprise Linux will want to acquaint themselves quickly with RPM, the package management tool used to install, upgrade, and verify software on many GNU/Linux systems. RPM packages are very similar to Solaris packages.

The software on a Red Hat Enterprise Linux system is composed of a set of RPM packages that belong to groups. Like Solaris packages, RPM packages may require other RPM packages for installation and contain pre- and post-install scripts. RPM packages can also be verified and uninstalled much in the same way that Solaris packages can.

A major difference between Solaris packages and RPM packages is that while Solaris packages are “patched” with fixes and updates, RPM packages are “upgraded.”



INSTALLING SOFTWARE

All RPM package operations (installation, upgrading, verification, and removal) are done with the rpm tool. The manual page for rpm provides comprehensive documentation of the command.

To install an RPM package, use the “-i” flag to rpm. The “-v” (verbose) and “-h” (hashes) options are commonly specified, as well. For example:

```
[root@rhel-x86_64-as-4 ~]# rpm -ivh mozldap-6.0.4-2.x86_64.rpm

Preparing...                               ##### [100%]
 1:mozldap                                  ##### [100%]
```

If a package dependency is not installed, rpm will give an error listing the missing dependency.

```
[root@rhel-x86_64-as-4 ~]# rpm -ivh mozldap-devel-6.0.4-2.x86_64.rpm

error: Failed dependencies:
        mozldap = 6.0.4-2 is needed by mozldap-devel-6.0.4-2.x86_64
```

To resolve the error, either install the necessary dependency package first, or install the two packages together.

```
[root@rhel-x86_64-as-4 ~]# rpm -ivh mozldap-6.0.4-2.x86_64.rpm mozldap-devel-6.0.4-2.x86_64.rpm
rpm
Preparing...                               ##### [100%]
 1:mozldap                                  ##### [ 50%]
 2:mozldap-devel                            ##### [100%]
```

Note that in addition to specifying RPM files to install on the local filesystem, you can also specify packages by URL (e.g. http://10.2.24.250/install/mozldap-6.0.4-2.x86_64.rpm).

UPGRADING SOFTWARE

Software upgrades are handled very differently with RPM than with Solaris. Unlike Solaris, RPM has no concept of “patches.” Whereas on a Solaris system, an administrator applies patches to installed software packages, on a Red Hat Enterprise Linux system, an administrator upgrades installed packages. This dramatically simplifies management of software updates.

For example, if there was an update to the mozldap package used in the previous example, the package would be upgraded on the system using the following command:

```
[root@rhel-x86_64-as-4 ~]# rpm -Uvh mozldap-6.0.4-3.x86_64.rpm

Preparing...                               ##### [100%]
 1:mozldap                                  ##### [100%]
```

This upgrades mozldap by removing mozldap-6.0.4-2 and installing mozldap-6.0.4-3.



When deploying a new Red Hat Enterprise Linux system that doesn't have the `mozldap` package, there's no need to install the 6.0.4-2 release and then upgrade; an administrator can simply install the 6.0.4-3 release of the package.

A note on RPM version numbering: There are two fields used to determine the version of an RPM package, the version and the release. In the example above, the package upgraded to is the third release of the 6.0.4 version of the Mozilla LDAP SDK. The version number corresponds to the software developer's version of the software, whereas the release number corresponds to the software packager's release of the package.

VERIFYING SOFTWARE

RPM packages contain a manifest of the files that they install. You can access the manifest using the query option to `rpm`, `"-q."`

```
[root@rhel-x86_64-as-4 ~]# rpm -ql mozldap
/usr/lib64/libldap60.so
/usr/lib64/libldif60.so
/usr/lib64/libprldap60.so
/usr/lib64/libssldap60.so
/usr/share/doc/mozldap-6.0.4
/usr/share/doc/mozldap-6.0.4/README.rpm
```

A verbose listing of the manifest is available with the `"-v"` option.

```
[root@rhel-x86_64-as-4 ~]# rpm -qlv mozldap
-rwxr-xr-x  1 root  root           215400 Feb 26 01:19 /usr/lib64/libldap60.so
-rwxr-xr-x  1 root  root           11768 Feb 26 01:19 /usr/lib64/libldif60.so
-rwxr-xr-x  1 root  root           21168 Feb 26 01:19 /usr/lib64/libprldap60.so
-rwxr-xr-x  1 root  root           49720 Feb 26 01:19 /usr/lib64/libssldap60.so
drwxr-xr-x  2 root  root              0 Feb 26 01:19 /usr/share/doc/mozldap-6.0.4
-rw-r--r--  1 root  root           4010 Jan 11  2006
/usr/share/doc/mozldap-6.0.4/README.rpm
```

To determine which package a file on the filesystem is a part of, use the `"-qf"` option:

```
[root@rhel-x86_64-as-4 ~]# rpm -qf /usr/lib64/libldap60.so
mozldap-6.0.4-3
```

To verify the package, use the `"--verify"` flag. If the installed files from the package match the manifest, there will be no output. If a file has changed since the package was installed, `rpm` will output the differences:

```
[root@rhel-x86_64-as-4 ~]# rpm --verify mozldap
.....UG..  /usr/lib64/libldap60.so
```



In this example, the user and group ownership of the file `libldap60.so` does not match the package manifest. Other attributes checked by RPM are the size (S), mode (M), MD5 sum (5), device (D), link path (L), and modification time (T). Any missing files will also be printed. Files designated as configuration files in the package will have a “c” in front of the filename, as it is expected that they will be modified. RPM treats configuration files differently than other files and has special logic to ensure that configuration changes are not erased during package upgrades.

You can also use the `rpm` command to fix some verification issues. The “`--setperms`” and “`--setugids`” switches will restore the proper mode and ownership of files in a package.

BUILDING RPM PACKAGES

RPM packages are built with the `rpmbuild` tool. To create a package, a developer writes a “spec” file describing the software package. This file is similar in many respects to the “prototype” file used in making Solaris packages. The spec file and the source code for the software are then used along with any patches to the software to create an RPM package. Detailed information on building RPM packages is available in the Fedora® Project RPM Guide².

USING YUM AND UP2DATE

Manually copying RPMs and resolving dependencies by hand can be a daunting task. To help simplify the process of installing and updating software, Red Hat Enterprise Linux 5 ships with `yum` (the Yellow dog Updater, Modified³). Earlier versions of Red Hat Enterprise Linux ship with the `up2date` tool. These tools can install and update software from a centralized software repository while resolving dependencies automatically. Software deployment can be further simplified by using the Red Hat Network to manage the software installed on systems.

DEVELOPER TOOLS

OVERVIEW

Red Hat Enterprise Linux ships with a robust development environment based around the GNU Compiler Collection (GCC). There are analogs to all of the common UNIX developer tools, including `make`, a debugger; a Yacc-compatible parser generator (`bison`); and a lex replacement named `flex`. The GNU C library implements a system interface compatible with POSIX (ISO/IEC 9945) specifications. Differences between UNIX tools and Red Hat Enterprise Linux tools tend to lie in the extensions to those standards. For example, the Sun C compiler has many extensions available via `#pragmas` that are not available in the GNU C Compiler.⁴

² <http://docs.fedoraproject.org/drafts/rpm-guide-en/>

³ <http://yum.baseurl.org/>

⁴ For a thorough discussion of pragma analogues, see the Solaris to Linux Porting Guide by Ulrich Drepper (<http://people.redhat.com/drepper/sol-porting.pdf>)



THE GNU COMPILER COLLECTION

GCC supports the ISO 9899:1990 C standard and the ISO 14882:1998 C++ standard, much like the Sun C compiler. ANSI-compliant code should translate between compilers with little modification. GCC also provides a “-std” flag, which allows the user to turn on and off GNU extensions to the C standards, similar to the “-xc99=none” flag for the Sun C compiler. Note that like the Sun C compiler, GCC does not fully support the C99 (ISO 9899:1999) C standard.

Some function prototypes may be in different header files, depending on the standard used at compile time. For example, the `crypt()` function will not be found in `unistd.h` unless `_XOPEN_SOURCE` is defined. While there are compile flags for ANSI standards, the various UNIX standards are only available through preprocessor directives.

GCC command syntax is very similar to the Sun C compiler syntax. To compile a C program into an executable, use the following syntax:

```
gcc -g hello.c -o hello
```

For optimization, use the “-O” flag. Note that GCC uses the syntax “-O2” instead of the Sun C compiler syntax “-fast”. For C++, use `g++` instead of `gcc` to specify linking against the GNU C++ library. To compile position-independent code for use in shared libraries, use the “-fPIC” option:

```
[root@rhel-x86_64-as-4 liba]# gcc -fPIC -g -c a.c
[root@rhel-x86_64-as-4 liba]# gcc -shared -Wl,-soname,liba.so.1 \
-o liba.so a.o
```

Linker options are similar to Sun linker options. “-Wl” passes options following it directly to the linker, “-L” adds paths to the library search path, and “-l” specifies a library to link against.

THE GNU LINKER

The GNU linker (`ld`) is similar to the Solaris linker. As on Solaris, it is normally invoked by the compiler and not directly.

GNU `ld` produces ELF-format executable files and libraries similar to those used on Solaris. As on Solaris, static linking is strongly discouraged on Linux. To create shared objects with GNU `ld`, pass the “-shared” flag to the linker (this is the same as the “-G” option on Solaris).

Similar environment variables affect the runtime linker on Linux as on Solaris. For example, `LD_LIBRARY_PATH` can be used to influence the dynamic library search path. `LD_RUN_PATH` also sets the run path for the GNU linker, although specifying a library run path at compile and link time is also generally discouraged on Linux.

Instead, when installing third party libraries into directories outside of the standard library search path, the directories containing the libraries should be added to `/etc/ld.so.conf`, and the `ldconfig` command should be used to update the system `ld.so` cache. For more information on the dynamic library loader on Red Hat Enterprise Linux, see the `ld-linux.so` manual page.



FURTHER READING

The following documents provide in-depth information for developers porting code to Red Hat Enterprise Linux.

Red Hat Enterprise Linux Developer Tools Guide, Red Hat

→<http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/devtools/>

Fedora Project RPM Guide, Eric Foster-Johnson

→<http://docs.fedoraproject.org/drafts/rpm-guide-en/>

The GNU C Library, Free Software Foundation, Inc.

→<http://www.gnu.org/software/libc/manual/>

GCC Online Documentation, Free Software Foundation, Inc.

→<http://gcc.gnu.org/onlinedocs/>

Documentation for Binutils, Free Software Foundation, Inc.

→<http://sourceware.org/binutils/docs-2.19/>

GNU coreutils Manual, Free Software Foundation, Inc.

→<http://www.gnu.org/software/coreutils/manual/>

LEARN MORE ABOUT RED HAT CONSULTING

www.redhat.com/consulting

EUROPE, MIDDLE EAST AND AFRICA

00800 7334 2835

www.europe.redhat.com

europe@redhat.com

Turkey: 00800 448 820 640

Israel: 1809 449 548

UAE: 80004449549

ASIA PACIFIC

+65 6490 4200

www.apac.redhat.com

apac@redhat.com

ASEAN: 800 448 1430

Australia and New Zealand:

1800 733 428

Greater China: 800 810 2100

India: +91 22 3987 8888

Japan: 0120 266 086

Korea: 080 708 0880

NORTH AMERICA

1-888-REDHAT1

www.redhat.com

LATIN AMERICA

+54 11 4341 6200

www.latam.redhat.com

info-latam@redhat.com