redhat.

Migration to Red Hat Enterprise Linux

Ulrich Drepper
Consulting Engineer

Why Migrate to Linux?

Linux has many advantages, neither Unix nor Windows has them all:

- Wide ranging support for commodity hardware
- Same OS supported from PDAs to super computers
- Support for seven architectures in RHEL:
  - Different architectures provide
    - the same API
    - Optimizations for different tasks
  - No hardware vendor lock-in
- Completely open APIs with no advantages for any ISV

Problems when Migrating

A developer faces a number of problems:

- Different tools (compiler, linker)
  - Different options
  - Differences in the accepted language
- Differences in the programming environment
  - Difference standards (or lack thereof)
  - Noncompliance to standards
  - Closed-source libraries not available
- Different runtime characteristics
  - Same code might run faster or slower
  - If code runs slower, code needs rewrite

Preparation of Migration

A number of preparation steps can ease porting:

- The GNU tools (compiler, linker, etc) are available on other OSes as well
  - Compile project on the other OS with the tools to be used on RHEL
    - Eliminate dependencies on language extensions of old compiler
    - Use correct options for tools
  - Enable all warnings and eliminate them
    - Add `-Wall -Wextra` to compiler command line
- Identify platform-specific interfaces used
  - Solaris threads vs POSIX threads
  - DCE
  - Rewrite code without these interfaces

Compile on Linux

After the preparation getting compilation started is easy

Possible remaining problems:

- Remaining platform-specific code (e.g., `#ifdef __solaris__`)
- Remaining architecture-specific code
  - Assembler code for different architecture (SPARC or PA RISC vs x86-64)
  - Different assembler syntax (Intel vs AT&T on x86)
  - Endianess problems (big vs little)
  - 32-bit vs 64-bit issues
- Closed-source libraries not available on Linux
  - Persuade 3rd party ISV to port libraries
  - Find replacement among plethora of libraries available on Linux

Aside from Compilation

The compilation process is not the only step which needs adjustment:

- Debugging: gdb is available on other platforms as well
  - Limited GUI capabilities outside Eclipse
- Memory handling debugging:
  - Purify available for Linux
  - Non-proprietary solutions:
    - Purify-like: valgrind
    - Special compile mode: mudflap
- Profiling:
  - gprof: old-Unix style, coarse granularity, exact call tree
  - Oprofile: system-wide profiling; kernel, applications, or DSOs
  - SystemTap: detailed kernel performance analysis

Standard Compliance

Goal of standard compliance is easier migration:

- Linux complies to POSIX wherever possible
  - Minor differences exist
  - No formal POSIX testing (nobody volunteered recently to pay $$$)
- Linux supports more POSIX options than any Unix OS
  - http://people.redhat.com/drepper/posix-option-groups.html
- A program using POSIX interfaces correctly should need almost no porting
  - API specified in standard (names of headers, data types interfaces)
  - Semantic specified to a great extend
    - Programs must not use unspecified behavior
    - Difficult to test this does not happen

Standard Compliance (cont.)

- gcc with glibc, libstdc++, and libgcj implement
  - Almost all of ISO C99 (only some minor features missing)
  - Most of ISO C++
    - Accepted language very close to ISO C++ (unlike other compilers)
    - Main missing feature: `export` keyword
    - C++ library fully supported and highly optimized
  - Fortran90 support
    - Not complete, but usable
  - Java support
    - gcj supports compilation to native code: higher speed
    - gij provides interpreter
    - libgcj mostly complete library support (as of Java 2)

Java

Certified Java environments available:

- Sun JVM
  - Available for x86, x86-64, and IA-64
- IBM JVM
  - Available on all seven architectures
- BEA JVM
  - Available for x86, x86-64, and IA-64
- Soon: Apache Harmony
- J2EE stacks
  - From the JVM providers
  - Jonas
  - JBoss

Migrating from Windows

It is a completely different story:

- The Windows API has nothing in common with the POSIX API
  - No 1-to-1 mapping
  - Some Windows APIs cannot be implemented in terms of POSIX interfaces
- Use of Windows (wine) emulation libraries not real migrating
  - Incomplete, always will be since MSFT is adding to it
  - Incredibly inefficient
  - GUI incompatible with native interface
- Possible to use POSIX interfaces on Windows
  - Unix Services for Windows
  - Cygwin

Adopt to Linux Environment

Last step: make the migrated application fit in

- If MOTIF widget set is used, convert to use gtk+
  - Native look & feel
  - Interaction with Linux applications through bonobo
  - Better resource usage
- Use Linux-specific interfaces
  - For performance
  - To reduce risk in programming
- Add support for advanced security
  - Extension to SELinux policy
  - Adjust build process to take advantage of ExecShield and related security features

Questions?

Comments?

Contact: drepper@redhat.com