OPS 102
OPERATING SYSTEMS for PROGRAMMERS

Software Installation

Chris Tyler

Software Developers have Unique Needs

Software developers have special software needs which are different from normal computer users. Our needs include:

- Development tools: editors, debuggers, testing tools, version control systems, and performance profilers
- Libraries and Modules, often in multiple versions
- Headers for Libraries

Development vs. Production

- Once software has been *developed*, it is deployed to the *production* systems on which it will be used. (These systems may be individual client PCs, servers, or other devices).

- Development and production contexts are often very different!

Development vs. Production Environments

Development Environment	Production Environment
Many versions of the software exist in parallel. For example, there may be a stable version which has been released and for which only bug fixes are being developed, and one or more pre-release versions, for which major enhancements are under development.	**Only one version of the software is usually installed on production systems.**
Source code, header files, developer's documentation, and version control information are present alongside the built software.	**Only the built software is normally present** (executable programs, user documentation, and program assets such as graphics, fonts, and sample data).
The software is often written, built, and tested in the software developers' personal workspace (i.e., within their home directory). Software developers may or may not have system administration capabilities.	**The software is installed in system directories which are not writable by normal users** (or, the software is installed and operated in *containers*). Installation, updating, and removal of the software may only be performed by users with system administration capabilities.
... is managed by Software Developers.	*... is managed by System Administrators.*

An Industry Saying

Beware of programmers who carry screwdrivers.

— Leonard Brandwein

Respect

- Software Development and System Administration are complimentary disciplines. A wise practitioner of either discipline respects the skilled practitioners of the other discipline!

- These meet in the interdisciplinary art of "DevOps".

Shared Libraries

- Library files (collections of functions / methods / procedures / subroutines) that are accessed by binary executables at runtime (when the program is executed).

- Same concept, different names:

  - Linux: Shared Object file (.so)
  - Windows: Dynamic Link Library (.dll)
  - MacOS: Dynamic Library (.dylib) and Shared Object file (.so)

Shared Libraries

- Shared libraries provide three main **benefits**:

1. **Reduced memory consumption** when multiple programs use the same version of the same library (just one copy of the library is loaded regardless of the number of programs using that library)

2. **Reduced storage requirements** because the library contents are not copied into each executable file, increasing their size

3. **Independent updating** of libraries and executables – libraries can be updated as new versions become available (e.g. bug fixes and security improvements), and programs will access the updated versions the next time they are executed

Shared Libraries

- Shared libraries present three main **challenges**:

1. **Version conflicts** may exist between different versions of the same libraries
2. **Multiple copies of libraries** may be installed in different directories on the system
3. **It can be difficult to track dependencies** - which libraries (and which versions) are needed by each piece of software

Installation Scopes

- *System* (or *Machine*)
  - Software is installed in system directories that are protected (regular users cannot write to these directories).

- *User*
  - Software is installed in a user's home directory (or subdirectories of their home directory).

System Software Directories in Windows

- Executables
  - C:\Program Files, C:\Program Files (x86)
  - C:\Windows\System, C:\Windows\SysWOW64
- Libraries
  - C:\Program Files, C:\Program Files (x86)
  - C:\Windows\System, C:\Windows\SysWOW64
- Header Files
  - C:\Program Files, C:\Program Files (x86)

. . . Executables and libraries are usually in the same directory on Windows

System Software Directories in Linux

- Executables
  - /bin, /usr/bin
  - /sbin, /usr/sbin
- Shared Libraries
  - /usr/lib, /usr/lib64
- Header Files
  - /usr/include

. . . Executables and libraries have their own directories on Linux

Demo

Let's take a tour

of the system directories

on both Linux and Windows . . .

Packaging Systems

- Software files can be collected along with metadata (including dependency information) in a *package file*.
- Package management software can then install these packages, using a local database on the system to track all of the installed software.
- A combination of software and online software repositories enable automated package retrieval, dependency resolution, and software updating.

Packaging Systems - Linux

- There are multiple system-level packaging systems in use in Linux. The two most popular file formats are *rpm* and *deb* (these names refer both to the package extension and the name of the software that manages those packages).

- An additional layer of software (*yum/dnf* for rpm-based systems and *apt* for deb-based systems) handles interactions with online repositories including package retrieval, dependency resolution, and software updating.

- Various programs are available to provide a GUI for *yum/dnf* and *apt*.

Package Management - Linux

Let's see system package management on

a Linux system in action . . .

Packaging Systems - Windows

- MSI (from "Microsoft Installer") is the traditional package format used on Windows; MSIX (from "Microsoft Installer XML") is an evolved version of MSI now used on Windows.

- It has also been popular to package Windows software as self-installing .exe files – the software is distributed as an executable binary that also contains an archive of the software which it installs when run. Since the executable file cannot be readily verified, this is considered an unsafe way to install software (compared to using an MSIX file) - nonetheless, it is still a popular approach.

Packaging Systems - Windows

- The Microsoft Store is a type of repository which can be accessed by the Microsoft Store app (WinStore.App.exe).

- The *winget* command is a built-in Windows CLI tool which will search, install, and remove package from the Microsoft Store as well as the winget repositories. Many open source software packages are available this way. By default, *winget* will try to install with a User scope, but this can be changed with the "--scope machine" argument.

- There is a third-party GUI available for winget called *WingetUI*.

Demo: WINGET

Let's see WINGET in action:

- `winget search` *`string`*
- `winget show` *`package`*
- `winget install` *`package`*
- *(test Gimp)*
- `winget uninstall` *`package`*

Building and Installing Open Source Software

- Most open source software can be built fairly easily as long as the required tools, libraries, and headers are available.

- Since it can't be installed into the system directories without administrator capabilities, it will need to be installed into your home directory (~ on Linux or %HOME% on Windows).

Demo: Building and Installing wget

Let's see an open source

package being built and installed.

PATH and LD_LIBRARY_PATH

- When needed, you can adjust your PATH environment variable so that programs installed in new locations can be found.

- If you need to install shared libraries in new locations, you can also add the directories containing those libraries to PATH (Windows) or to LD_LIBRARY_PATH (Linux)