SOFTWARE INSTALLATION

Gotta have it

DELIVERY!

- Software is delivered in one of two manners:
  - Source form - requires compiling
  - Binary form - generally wrapped up in a package

WHICH IS BEST?

- Both formats have their advantages and disadvantages..
  - Compiling from source can provide higher performing machine code, plus it gives the option of selecting features and configurations only available at compile time.
  - Pre-compiled software is easier - it alleviates the [possible] headaches of compiling, and if distributed in a package format, provides built-in management functionality.

COMPILING

- Compiling from source can be tricky.
- First of all, the development tools and packages must be installed, most importantly: `gcc` and `make`.
- `gcc`: The GNU C Compiler. The de facto compiler for open source software.
- `make`: GNU Make. A development tool which uses a rules-based configuration syntax to determine and run all of the necessary commands needed to build a software project.

COMPILING BASICS

- The basic steps for compiling a software package:
  - Download the source tarball
  - `cd` into the extracted directory
  - Read the `INSTALL` and/or `README` file, follow directions!
  - `./configure`
  - `make`
  - `make install`

COMPILING GONE WRONG

- The previous steps are for well-maintained open source projects that follow the GNU coding standards, and make use of a very cool tool called `autoconf`.

- Sometimes it's not that simple. The `README` and `INSTALL` files can help explain the build process.

- If an error comes up during compilation, try reading the error message, and if it makes sense, fix whatever the problem is (permission issue, for example). If the message seems to be in a foreign language, try googling the name of the software plus the error message.

- Past that, learning to code is your next best bet. :)

PACKAGES

- Installing a software package is pretty straight forward.
- There are a few different package formats out there. The two most popular are:
  - `rpm`: Redhat Package Manager
  - `deb`: Debian package

RPM

- RPM's provide full software packaging features: pre-install scripts, post-install scripts, dependencies, meta information, and an installed software database to name a few.

- The RPM system maintains a database of all installed software on a machine - this is useful for tracking and updating reasons, as well as dependency verification and software management.

RPM

- `rpm`: The Redhat Package Manager tool. Provides interface to RPM system, performing queries, installs, upgrades, uninstalls and general database maintenance operations.
  - `-i` option: install the given package
  - `-q` option: query the database
  - `-e` option: erase the given package from the system

RPM EXAMPLES

YUM

- Not yum as in “This is yummy!”
- `yum`: Yellowdog Updater Modified
- Supports package installation over the network through repositories. Config: `/etc/yum.conf`
- `RPM` backend
- Simple interface

YUM EXAMPLES

DPKG

- `dpkg` is akin to `rpm`. It is the backend package workhorse for Debian based distributions.

- `dpkg` provides similar features and functionality as `rpm`. For example:

  - `-i`: install a package

  - `-l`: list installed packages

  - `-r`: remove and installed package

APT TOOLS

- The `APT` tools are akin to `YUM`. They provide support for installing packages remotely and handling dependencies.
- `apt-get`: Install/upgrade a package
  - Supports package installation over the network through sources. Config: `/etc/apt/sources.list`
  - Originally for `dpkg` backend, but now also `RPM` backend
  - Simple interface
- `aptitude`: TUI frontend for managing packages

EXERCISES

- Browse through the manpage for ‘rpm’. Study the “Query” section.

- Use your new knowledge to produce an alphabetized listing of the names for every installed package on your system.

- To what package does ‘/usr/bin/time’ belong?

```
slideshow.end();
```