Package Managers
What are they and why we use them
Thoughts of an admin

• Installing software is painful
• Installing a lot of software is extremely painful
• Installing a lot of software on a lot of machines is **HELL**
舉例 - LOL

- 天賦、符文
- 道具
Before Packages

- Install from source
- Time consuming
- Varying configuration
Improvement 1

- Copy the binaries
- Setup scripts?
- Version?
- Dependencies?
Package - Basic

- Archive
- Binaries
- Supporting files
- Setup script
Package – Improvement

- Versioning
  - 1.0.x
  - 1.1.x
  - ...
- Upgrades!
Package - Dependencies

- Program A requires library B to work
- If library C is available, program A will work better/faster!
Example – Call of Duty

- Requires: DirectX
- Works better with: PhysX
Package Managers
Low Level

- Unpacking packages
- Run configuration scripts
- Dependency checks
- ...

Package Managers
High Level Usage

- Fetch from remote repositories
- Search
- Install additional packages to meet dependencies
- Handle complex upgrades
Package Repository

- Collection of packages
- Maintained by distribution
- Multiple package versions
- Package index
## Package Managers

<table>
<thead>
<tr>
<th></th>
<th>RPM Based</th>
<th>DEB Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low level</td>
<td>RPM</td>
<td>Dpkg</td>
</tr>
<tr>
<td>High level</td>
<td>YUM, up2date, ...</td>
<td>APT</td>
</tr>
</tbody>
</table>
Focus on

DEBIAN DPKG & APT
Types of Packages

- Binary Package
- Meta Package
- Virtual Package
Binary Package

- Normal package
- Source is available
- Architecture
  - Dependent
    - Compiled binary files
  - Independent
    - Scripts
    - Documents
    - Data files
Meta Package

- Depends on other packages only
- Rename
  - git-core -> git
- Default version
  - gcc -> gcc-4.6
  - Python -> python2.7
- Can be creative!
Virtual Package

- Does not really exist
- Names common functionality
- Other binary packages “provide” a virtual package
- c-compiler is provided by
  - gcc
  - gcc-4.6
  - clang
Commonly Used Commands

- apt-get
- apt-cache
- aptitude
  - Frontend to the first 2 commands
Search for a package

- `apt-cache search ^vim$`

Search pattern

State flags
Search Patterns

- Patterns are regular expressions
- Aptitude supports complex patterns
  - See reference

- Search requires guessing
  - Just like Googling
  - Keywords
Differences

- Aptitude and apt-cache have slightly different search behavior
  - Aptitude looks at package name only
  - Apt-cache looks at description as well
# State Flags

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Installed</td>
</tr>
<tr>
<td>c</td>
<td>Removed</td>
</tr>
<tr>
<td>p</td>
<td>Purged</td>
</tr>
<tr>
<td>v</td>
<td>Virtual package</td>
</tr>
<tr>
<td>B</td>
<td>Broken dependencies</td>
</tr>
<tr>
<td>u</td>
<td>Unpacked</td>
</tr>
<tr>
<td>C</td>
<td>Half-configured</td>
</tr>
<tr>
<td>H</td>
<td>Half-installed</td>
</tr>
</tbody>
</table>
Try it!

- You want to install a web server. What do you look for?
- What should you actually install?
Looking at a package

- Web interface
  - [http://packages.debian.org](http://packages.debian.org)

- CLI
  - `apt-cache show XXX`
  - `aptitude show XXX`

- Debian Control File

- Try it now!
wens@dev:~$ aptitude show vim
Package: vim
State: installed
Automatically installed: no
Version: 2:7.3.429-2ubuntu2.1
Priority: optional
Section: editors
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Uncompressed Size: 2061 k
Depends: vim-common (= 2:7.3.429-2ubuntu2.1), vim-runtime (= 2:7.3.429-2ubuntu2.1), libacl11 (>= 2.2.51-5), libc6 (>= 2.15), libgpm2 (>= 1.20.4), libpython2.7 (>= 2.7), libselinux1 (>= 1.32), libtinfo5
Suggests: ctags, vim-doc, vim-scripts
Conflicts: vim
Provides: editor
Provided by: vim-athena, vim-gnome, vim-gtk, vim-nox
Description: Vi IMproved - enhanced vi editor
   Vim is an almost compatible version of the UNIX editor Vi.

   Many new features have been added: multi level undo, syntax highlighting, command line history, on-line help, filename completion, block operations, folding, Unicode support, etc.
Package Info

- Name
- State
- Version
- Priority
- Section
- Maintainer
- Architecture
- Dependencies
- Description
Package Info - State

- installed
- not installed
- removed
- ...


Package Info - Version

- **Vim**: `2:7.3.547-6`
- `2:` -> Epoch
- `7.3.547` -> Upstream version
- `-6` -> Debian package revision
Package Info - Priority

- Required
- Important
- Standard
- Optional
- Extra
Package Info - Dependencies

- Depends
- Recommends
- Suggests
- Conflicts
- Replaces
- Breaks
- Provides

Installing / Removing

- apt-get install XXX
- aptitude install XXX

- remove/purge
- hold/unhold
- upgrade
- dist-upgrade
remove vs. purge

- Remove keeps config files in system
- Purge = Remove + delete config files
Hold/unhold

- Keep a package at a specific state/version unless explicitly requested otherwise.
upgrade vs. dist-upgrade

- upgrade only updates currently installed packages
- dist-upgrade will do anything to satisfy all new dependencies
Try it!

- Install a web server
- Then remove it
- See that config files are still there
- Then purge it
MAKING A PACKAGE
Why?

- Customize
- Software not packaged yet
  - Become a maintainer?
- Build your own
Making your first package

- **Basic structure**
  - `package/XXX`
    - Files you want to package
  - `package/DEBIAN/control`
    - Debian control file
  - `package/DEBIAN/xxx`
    - Package scripts, other control files

- `dpkg-deb --b package`
Homework – Build a meta package

- Name: nasa-hw
- Should always install
  - gcc >= 1:4.7
  - make > 3.80
- Should not install when
  - clang is installed
- Maintainer: Your ID <your email>
- Use your judgment for other fields
  - Read the documents!
- Submit control file
Too simple?

- Debian related source code
  - source/debian/rules
  - source/debian/control
  - source/debian/changelog
  - source/debian/...
Guidelines for packaging

- Debian 新維護人員手冊
- Get other package sources and see how other people do it
  - apt-get source XXX
Packages for packaging

- build-essential
- dpkg-dev
- debhelper
- CDBS
- dh_make
- lintian
**build-essential**

- Meta package that depends on essential tools for packaging
- You need to install additional tools/libraries
  - debhelper, cdbs, ...
  - lib*-dev
dpkg-dev

- Basic tools for package development
- dpkg-buildpackage
Debhelper

- Collection of commands to assist packaging
- Wraps complex steps
  - Splitting files between different packages
  - Generating shared library dependencies
  - Generate and sign binary packages
  - ...

CDBS

- Common Debian Build System
- Class based system
- Supports different build systems
  - make, ant, cmake, ...
- [http://www.slideshare.net/petereisentraut/the-common-debian-build-system-cdbs](http://www.slideshare.net/petereisentraut/the-common-debian-build-system-cdbs)
dh_make

- Prepare source code for packaging
- Always uses debhelper
- Supports CDBS
lintian

- Check packages for errors
Steps to packaging

1. Get source code
2. Unpack
3. dh_make
4. Install build dependencies
5. Check control file
6. Check installation paths
7. Build package
   ◦ dpkg-buildpackage –b
8. Check package contents, debug if necessary
   ◦ dpkg-deb –c xxx.deb
Packaging Tutorial

Try if you want

- Last semester’s homework
  - Very hard
- Build a package of “Omnitty”
- You will also need to build a package for “ROTE”
- Build it, install it, and try it out
Tips

- Packages you can look at
  - libevent
  - screen
  - tmux
  - make
HOMEWORK RECAP
• nasa-hw meta package
• Upload control file to CEIBA
• Start sooner! You need to do research to finish this assignment.