

Lab 03.01

Debian Packages

IN719 Systems Administration

1 Introduction

Often in the process of operating a set of computer systems we find that we need to install some custom software. The process of doing this may require copying executable and configuration files, perhaps along with other resources. It may also be necessary to verify that dependencies are met on the target systems, and we might need to carry out some other procedures to complete the installation. Since we want to make the installation process reliable and repeatable, we might be tempted to write a shell script to manage the installation process. But before we do that we should remember that there is already a tool that does this. That tool is a *package manager*, and to use it we just need to make a package.

Package managers do not just help install software packages. They keep track of what packages are installed, they manage package upgrades, and they remove packages that are no longer required. Debian and its derivatives have an excellent package manager that we will use to create and install a package in this lab. While other distributions use different package managers, the basic principle remains the same.

2 The problem

In the `week03` folder on GitHub there's a simple shell script called `interactive_cowsay`. We want to install it in `/usr/bin`. But our script has a *dependency*, the Debian package `cowsay` that is available from standard repositories. We'll create a `.deb` file for our program that addresses these requirements. To begin, copy the `interactive_cowsay` script onto the lab server. Create a subdirectory of your home directory called `interactive_cowsay_package`. We'll use it to set up our package.

3 Preparing the package

1. We want to install the executable file under `/usr/bin`, so create a `usr/bin` subdirectory tree under `interactive_cowsay_package`. Move the `interactive_cowsay` file into `interactive_cowsay_package/usr/bin`.
2. We want to get the ownership and permissions right at this stage. Make sure that `root` owns the `usr` subdirectory and everything under it. Make sure that everybody can read and execute `interactive_cowsay`.
3. Create another subdirectory of `interactive_cowsay_package` called `DEBIAN`.
4. Inside `DEBIAN`, create three text files: `control`, `postinst`, and `prerm`. `postinst` is a shell script that does whatever tasks must be done after the package files are installed. `prerm` is a shell script that does whatever must be done before removing package files when uninstalling. We don't need to do anything special at those time with our package, so both files just need to contain the following:

```
#!/bin/sh
exit 0
```

5. The control file contains information about your package. Use the following information, it is explained below.

```
Package: interactive-cowsay
Version: 1.0
Architecture: all
Essential: no
Depends: cowsay
Installed-Size: 512
Maintainer: Tom Clark <tclark@op.ac.nz> (use your own information here)
Description: Provides an interactive front end for cowsay
```

Package What you chose to name your package. It shouldn't conflict with another package's name and it can only contain alphanumeric characters, hyphens, and full stops.

Version Whatever numbering scheme you want, but the standard x.y numeric format is best. Don't use hyphens, e.g., 2.4-2.

Architecture Our package works on all hardware architectures.

Essential Our package is not essential. If you try to uninstall essential packages, you will get a warning message.

Depends Our package depends on the package `cowsay` to work properly.

Installed-Size Our package will take up 512 bytes of disk once it's installed.

Maintainer Whom to blame for this fiasco.

Description A short description of the package.

6. Now you're ready to make the package. Move back up to your home directory, or wherever the directory `interactive-cowsay_package` is located. Issue the command `dpkg -b interactive-cowsay_package interactive-cowsay-1.0.deb`. This should produce a `.deb` file named `interactive-cowsay-1.0.deb`.

4 Installing the package

Now you can install the package with the command `sudo dpkg -i interactive-cowsay-1.0.deb`. Try it. What happens? The problem is that your package depends on the `cowsay` package. But `dpkg` isn't smart enough to find and install it for you. For that, you need `apt-get`, and to use it for your new package, you need to set up a Debian *repository*. Setting up and running a repository is not hard, but it's beyond the scope of today's exercise. However, you can complete the install by installing `cowsay` yourself and then installing your new package.

Once you have installed the package, you can check its status with the command `dpkg -s interactive-cowsay`. Finally, try out your package by running the command `interactive-cowsay`. If after trying it you are unconvinced that it will be useful, you may remove it with the command `sudo dpkg -r interactive-cowsay`.