# RCOS Project Proposal: big

Ian Ooi

Summer 2012

## 1 Overview

Big is a proposed tool for generating configurations. The main program will use the system's package manager to create a list of installed programs to be retrieved, updating as the user adds and removes packages. It will also browse the filesystem to discover other installed programs, as well as for common configuration files and related installations, such as printers, vimrc, bashrc, and xinitrc. As opposed to something like Clonezilla, Chef, or Puppet, big is made to expedite generation of system images, or archives similar to a system image.

Files besides configurations and necessary installed files would be ignored, aiding in situations where a certain package or configuration file is causing an issue. Users can set options for what big should and should not grab for the archive, as well as any additional, specific locations or files that should be saved, such as the vim syntax folder or .xmonad folder. For packages or programs that are no longer available or were compiled from source, there will be an option to store the package, or the source code and even to set it to make and install automatically.

An analogy would be that it should produce system configurations and scripts to install it similar to how cmake produces necessary makefiles.

The purpose is to allow a user who has a specific setup that they prefer for productivity or personal happiness to:

1. recover after accidental unwanted software changes such as deletion or changes to the system which cause unwanted changes
2. recover their setup after hardware damage, unwanted software changes, or to transfer their setup to new systems, such as additional computers or additional OS's (have the same packages on both Debian and ArchLinux)
3. otherwise backup their programs and configuration without storing their files to allow for expedited "clean" installs

## 2 Development Plan

The project will likely be developed in C/C++, though other languages could be considered (Python, Perl, or Haskell perhaps). Since these languages are also extensible through C/C++, it is plausible that there would be a combination of languages used for development.

The initial platform to develop for will be Debian/Ubuntu, targeting `.deb` packages and aptitude/synaptic/dpkg first, before extending to additional platforms.

Tentative development plan:

1. grab all of the installed packages from dpkg (dpkg -l) and generate a simple script to reinstall them
2. read configuration script and choose which packages to include in the script (including dependencies?)
3. step through directory, check settings from configuration file, either store/don't store packages appropriately. options are:
   - store everything

- prompt user
- check a list of “store only these”
- “store these, ignore others”
- “store these and ask about others”
- “ignore these, store others”
- “ignore these and ask about others”
- “ignore all”

4. run through file system and grab configuration files associated with packages (also grabbing associated directories from packages)
5. find a way to latch onto package manager and grab the downloaded .deb the user is unpacking (hooks)
6. grab stored repositories (Ubuntu’s ppas)
7. run through file system and grab installed programs
8. create an archive and/or disk image which can be backed up
9. in cases where the user wants to burn the archive to a disk, split into multiple archives
10. extend to other Unix/Linux/BSD platforms
11. build into packages/install script, at least have src and Makefile to build with and README with instructions