RCOS Project Proposal (Detailed): big

Ian Ooi

Summer 2012

NOTE: Suggestions, modifications (in both function and implementation) are welcome and encouraged. The name is also open for change (as well as the syntax for the commands of course). I originally came up with big simply for the novelty of being able to run `big bang` and `big crunch`.

1 Overview

There are numerous ways to store a system image, but there isn't a really solid way to store a clean, empty configuration of your system setup. As a linux user, I have found that every so often, I will try something and it will go horribly wrong and I won't know how to fix it. A user (myself included) may also want to reinstall entirely, just for a clean start, or perhaps to fix partitioning issues or add other operating systems.

Big is a proposed command line tool for creation of "system setup images." At its base, it will keep a constantly updated list (updated when new things are installed or uninstalled) of all packages currently installed and when they were installed, as well as copies of related config files and some data about the current operating system and computer (32 or 64-bit, what OS, possibly information such as what graphics card/drivers). The user will be able to specify options such as whether to store all new packages, to ask whenever a package is installed, or to never store newly installed packages (user manually calls big to grab all currently installed programs). Furthermore, the user will be able to specify whether to always store related config files, to ask for each, or to never store config files. "Config file" refers to general config files, as well as additional files (e.g. .vimrc, .bashrc, .minecraft folder, .xmonad folder) and where such files are if big is unable to locate them.

An additional feature would be to actually store packages or archives as well as just the name/author/location/where to retrieve it from, for cases where a user has written their own source, performed significant modifications on a piece of software that they would like to carry over to another setup, or they have a specific version of a package they want to keep (or if it is no longer supported, maintained, or available). As with the other features, the user will be able to specify whether to do this automatically, only for certain packages, to ask each time, or to never do this.

It would also be useful to create a Windows version, to allow for backing up programs (storing installers) in case of a virus or system problem, similar to system restore.

1.1 Why is this different than a system image?

Big will not store a user's files, besides configuration files, or special cases such as if the user specifies that the .minecraft folder should be saved, as opposed to a system image. There are many tools for system images, and especially for Windows, there are tools such as the included System Restore tool, but there are no such tools that I have found for linux.

The general recommendation is to back-up your system before bad things happen, such as running a backup on `/` right after you have installed everything you want. This however, requires you to run the backup very early, and doesn't solve the problem of discovering a package later (e.g. you have been using your current Debian setup for a year, and a few things are working sub-optimally now after you attempted to fix a bug, and you decide to use xmonad as your preferred window manager).

Similar software:

- Windows System Restore
- partimage (disk clone utility)
- clonezilla (disaster recovery/disk clone/deployment)

1.2 Why is this useful?

The purpose is to allow a user who has a specific setup that they prefer for productivity or personal happiness (they always want to have vim, xmonad, firefox, android sdk, haskell, perl, python, wine, gnome, cmake, and latex installed) 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 bitem 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 Software, Libraries, and Related Research

To develop this software, it will be necessary to research a method of "listening" for when an install is happening, "latching onto" the package manager or otherwise detect what is installed on the user's system (such as by looking through `/bin`, `/usr`, `/lib`, and `/var`). In the former method, the program will either automatically run when, pacman/dpkg/etc. is run, storing the name of the package and some related information such as title of the software and author. This will require further research into how to extract such information from a package (for `.deb`, read /DEBIAN/control). In the latter method, big will need to run through the file system to detect what is installed.

Additionally, a method of finding related config files must be researched, as well as a way to write all of this to an archive of some kind, or to a disk image the user can burn to removable media for safe-keeping.

For a possible Windows version, a method of storing installers, discovering the program names, and retrieving program setup data from `C:\Program Files` will need to be found. Big should also be tested on Mac, and additional changes for this platform should be investigated (I have no idea about Mac).

3 Development Plan

The project will likely be devloped 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. read info from a `.deb` package and stores it in a manageable format in a manifest file
2. step through directory, check settings from configuration file, either store/don't store packages appropriately. for specific options see "Commands to be Implementation" below
3. run through file system and grab configuration files associated with packages (also grabbing associated directories from packages)
4. find a way to latch onto package manager and grab the temporarily downloaded `.deb`
5. grab ppa's etc. from package manager
6. run through file system and grab installed programs
7. create an archive and/or disk image which can be backed up
8. in cases where the user wants to burn the archive to a disk, split into multiple archives
9. extend to other Unix/Linux/BSD platforms
10. build into packages/install script, at least have src and Makefile to build with and README with instructions
11. extend to Mac?
12. extend to Windows? (essentially a whole new program, is this worth building and/or maintaining?)

4 Commands to be Implemented

As an implementation note, when running crunch, big should fail gracefully, printing an error and continuing (such as when an included file/package cannot be found). When searching for a package, if multiple are found, the user will be prompted to choose one, which will cause the manifest file to update. When running bang, big should also fail gracefully, redirecting the output from installing each package to a logfile, only printing its own errors and prompts to the screen unless the user is prompted by a package.

- `big crunch` grab everything from the filesystem and store the information about packages as well as config files, storing info in a manifest file (default without any options is to create a new big archive, can be changed in configuration file)
- `big bang [archive file location]` unpack everything from an archive, read the manifest file, and install packages (either stored locally or from package manager)
- `big --set [option]` write an option to the config file (config file can also just be edited in file system, store in `~/.big`?). user can specify where configuration file is. this is also where the user can turn on whether big runs
- `big crunch --include=[option]` set how big handles crunching of packages. options are:
  - `all` means include all package files (downloads from package manager or specified source site if not available) in manifest
  - `ask` means to ask for each program whether to store in manifest
  - `never` means don't store anything in manifest (it can still grab conf files and packages, but bang will not run on this archive, prompt informing user of this)
  - `allexcept` means store all except for a specified list which big will ignore
  - `allexceptask` means store except for a specified list (specified in config file) for which big will ask
  - `neverexcept` means ignore except for a specified list which big stores
  - `neverexceptask` means ignore except for a specified list for which big will ask
- `big crunch --store-config=[option]` set how big handles crunching of config files for this usage. options are:
  - `all` means store all config files
  - `ask` means to ask for each program whether to grab config files
  - `never` means don't store config files
  - `allexcept` means install all except for a specified list which big will ignore
  - `allexceptask` means ask except for a specified list (specified in config file) for which big will ask
  - `neverexcept` means install none except for a specified list which big installs

  - `neverexceptask` means ignore except for a specified list for which big will ask

- `big crunch --store-packages=[option]` set how big handles crunching of packages. options are:
  - `all` means store all package files (downloads from package manager or specified source site if not available)
  - `ask` means to ask for each program whether to grab package files
  - `never` means don't store package files
  - `allexcept` means store all except for a specified list which big will ignore
  - `allexceptask` means store except for a specified list (specified in config file) for which big will ask
  - `neverexcept` means ignore except for a specified list which big stores
  - `neverexceptask` means ignore except for a specified list for which big will ask

- `big crunch -a [name of existing big archive to add to]` or `--archive [name of existing big archive to add to]` specify an existing setup image to add to. if no name is specified, will default to the file specified in the big configuration file.

- `big crunch -n` or `--new` forces creation of a new big (default, can be changed in big configuration file)

- `big crunch -o [file name/path]` or `--out [file name/path]` creates archive file in specified location

- `big crunch -l` or `--latest` adds to the most recently used archive

- `big [bang/crunch] -f [filepath]` or `--local-file [filepath]` attempts to include local file or package at filepath in the bang/crunch

- `big [bang/crunch] -p [package name]` or `--p [package name]` attempts to include package in the bang/crunch

- `big -h` or `--help` display help (ignores all other commands if help specified)