Building and Installing Custom Software for the Efika MX53

Chris Jenkins - Genesi USA, Inc

Prerequisites: Software

- `.deb`: Debian program package file, containing program files and metadata (dependencies, version, etc)
- `dpkg`: **D**ebian **P**ac**k**a**g**e Manager, used to install software from `.deb` files
- `apt-get`: smart front end for `dpkg`, aware of online repositories (located in `/etc/apt/sources.list`)
- `make`: common software build tool
- `git`: Open source version control software

Example: Cheese

Cheese is a common web-camera booth software distributed with our Ubuntu. We had to make some changes to it to accommodate fixes to GStreamer[1] plugins and kernel drivers for our camera. We will use this as an example of how to modify existing open source software to make custom packages for Efika MX53 systems, such as the Drónov Slimbook.

[1]: GStreamer is an open source multimedia framework for Linux

Cheese: Fork the repo

- “Forking” is taking a software repo and making a copy of your own with a different remote
  - GitHub makes this easy. You can fork a repo from the web interface of any public repo hosted there. You can fork software with any repo hosting service or version control, however.
- If you’ve forked a repo using GitHub’s web interface, you’ll next need to clone it locally.

  -
    ```
    git clone http://github.com/forked-remote/cheese.git
    ```
  - Unless the software is colossal (like the Linux Kernel) you should build your software on Efika MX53 systems, such as the Drónov Slimbook, itself
  - If there are several branches, make sure you checkout the most appropriate one:

    ```
    git checkout -t origin/slimbook
    ```

Cheese: Test Local Build

- Read the `README` / `INSTALL` files.
- Almost all significant software has dependencies.
  - Cheese has a list of these in README. Install with `apt-get install ${PACKAGES}`
  - Running `./configure` will also tell you if there are any unmet dependencies
  - Package names are not always the same as software names, some research is required.
  - You also often need the "`-dev`" packages, which include the headers for a given software package
  - You can search for packages using `apt-cache search ${SEARCH_STRING}`
- Test the build:

  -
    ```
    $ ./configure
    $ make
    ```
- If it builds without error, you are ready to make your changes
  - If there are errors, run `make clean` and start debugging. The are usually issues with dependencies and / or paths.

Cheese: Testing Changes

- You’ve made your changes and made sure they compile.

  Running `make` will create an executable that you can test.
- If for some reason that doesn’t work (permissions, paths, etc) you can install the software on your local machine
  - Be sure you uninstall any default versions of this software before proceeding!
  - `sudo make install`
- Finally, if the software already supports debian packaging you can test your changes by creating a `.deb` (shown on the next slide) and installing through `dpkg -i`

Cheese: Creating `.deb`

- When you are ready to package your software, navigate to the root of the repo
  - If you are in an “unclean” state after testing a build, do:

    ```
    git reset --hard HEAD
    ```
  - If you made any changes that you are not ready to test yet, do:

    ```
    git stash
    ```
- Build software package:
  - `dpkg-buildpackage`
  - Optional `-us` and `-uc` flags to skip signing sources or changelog, resp.
  - If this is first build, you may have to install `dpkg-buildpackage` build dependencies
- Results will be in parent directory of repo, including:
  - `*.changes`: file containing changes made to the package by your build
  - `*.dsc`: summary of package contents
- If you wish to add your own packages to the Efika MX53 or Drónov Slimbook software repo, contact Genesi

genesi