Aakash application development

http://aakashlabs.org

Indian Institute of Technology, Bombay

February 17, 2013

AAKASH
tablet for every indian
NME-ICT (MHRD), AAKASH PROJECT, IIT BOMBAY

ICFOSS

Outline

GNU/linux as an alternate OS on Aakash

- runs entirely from `SDcard` in native mode
- not all FOSS applications can be ported *as it is*
- Android is not the only OS on phones/tablets
  - Ubuntu on phones
  - Firefox OS

GNU/linux as an alternate OS on Aakash

- runs entirely from `SDcard` in native mode
- not all FOSS applications can be ported *as it is*
- Android is not the only OS on phones/tablets
  - Ubuntu on phones
  - Firefox OS

Preparing your GNU/linux build

cross compiler toolchain

- download cross-compiler toolchain
-
  ```
  sudo dpkg-reconfigure -plow dash
  ```
-
  ```
  export PATH=</your/codesourcery/bin/path>:$PATH
  ```

uboot

-
  ```
  git clone -b sunxi
  https://github.com/androportal/uboot-allwinner.git
  ```
-
  ```
  sudo apt-get install u-boot-tools
  ```
-
  ```
  make a13_olinuxino
  CROSS_COMPILE=arm-none-linux-gnueabi-
  ```

Preparing your GNU/linux build

kernel dependencies

- fakeroot, build-essential, gcc, kernel-package, make, automake, libncurses5-dev, git

compilation

-
  ```
  make ARCH=arm a13_defconfig
  ```

-
  ```
  make ARCH=arm menuconfig
  ```

-
  ```
  make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage
  ```

-
  ```
  make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- INSTALL_MOD_PATH=out modules
  ```

-
  ```
  make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- INSTALL_MOD_PATH=out modules_install
  ```

Preparing your GNU/linux build

root filesystem

- http://cdimage.ubuntu.com/ubuntu-core/releases/12.10/release/ubuntu-core-12.10-core-armhf.tar.gz
- `qemu-user-static`

Preparing your SDcard

- partioning
- copy uboot
- copy kernel
- releasing your GNU/linux image

Managing your application using version control

- why version control ??
- types of version control

`git`

- entirely written in `C`
- recommended for large projects, eg. `linux kernel`
- the way `git` handles `branches`, and `conflicts` is *owesome!!*
- `github.com`, `gitorious.org`, etc

using `git`

- initialize you project
- adding files to `git`
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using git

- initialize you project
- adding files to git
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using git

- initialize you project
- adding files to git
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using git

- initialize you project
- adding files to git
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using `git`

- initialize you project
- adding files to `git`
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using `git`

- initialize you project
- adding files to `git`
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using `git`

- initialize you project
- adding files to `git`
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using `git`

- initialize you project
- adding files to `git`
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using git

- initialize you project
- adding files to git
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

using git

- initialize you project
- adding files to git
- commit your changes
- pushing it to remote host
- revert to know state
- branching and merge
- handling conflicts
- pull requests
- clone

why document our project?

- No documentation, No project!
- User and developer's guide
- Contributing to existing code

sphinx

- plain text(`.rst`)
- even **Documentation** can be *version controlled*

- `sudo apt-get install python-sphinx`
- `sphinx-quickstart`
- `sudo apt-get install rst2pdf`
- `pandoc`

`http://docs.python.org/devguide/documenting.html`

sphinx

- plain text(`.rst`)
- even **Documentation** can be *version controlled*

- `sudo apt-get install python-sphinx`
- `sphinx-quickstart`
- `sudo apt-get install rst2pdf`
- `pandoc`

http://docs.python.org/devguide/documenting.html

sphinx

- plain text(`.rst`)
- even **Documentation** can be *version controlled*

- `sudo apt-get install python-sphinx`
- `sphinx-quickstart`
- `sudo apt-get install rst2pdf`
- `pandoc`

`http://docs.python.org/devguide/documenting.html`

app release

- `play.google.com`
- host it on `github.com` or your custom page
- releasing version (`git tag`)

Questions?

References and links

- github.com/androportal/linux-on-aakash
- http://git-scm.com/book