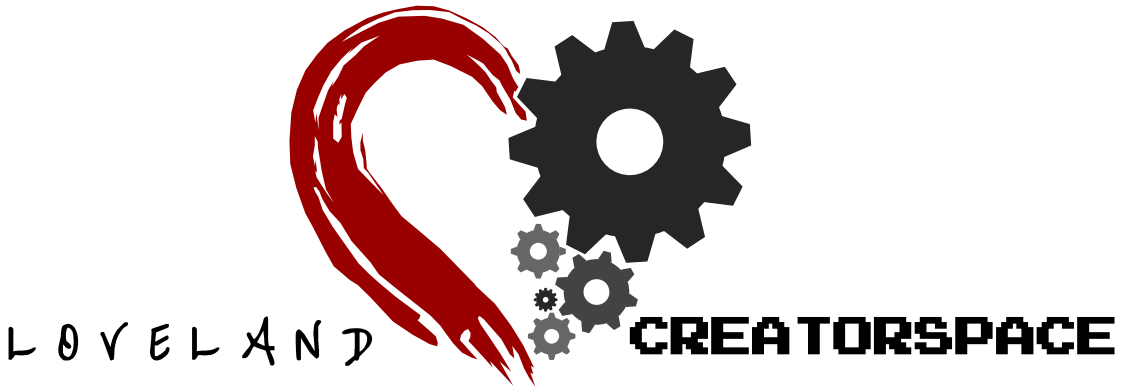
Introduction To

Free/Open Source Software

Development Process



What Is A Binary

- A representation of a computer program
- Computer instructions encoded into the CPU's native representation
- Each instruction is a number
- Very low-level instructions (assembly level)
- Generally not considered
  - Human readable
  - Modifiable

What Is Source Code

- A representation of a computer program
- The format the SW was originally written in
- Encoded as text
- Human readable
- Readily modifiable in any old text editor
- Can be much more abstract than CPU instructions

Proprietary Software

(slight generalizations)

- You get just the binary, not the source code
- You are allowed to use it with restrictions like:
  - How many copies you can use
  - Who can use it
  - Where you can use it (CPU architecture, OS, geography)
  - What you can use it for
- You can't fix problems
  - You don't have the source
  - The license probably prohibits it (e.g. DMCA)

Free/Open Source Software

(slight generalizations)

- You get the source code
- You can run it wherever/whenever you like
- You can modify it
- You can fix bugs
- You can share it, and modified versions

F/OSS Licenses

- There are many (too many?)
- Differ in:
  - Requirement to share source changes when sharing binaries
  - Requirement for attribution
  - Modifications allowed?
- Not all are compatible!
  - (ability to be mix/match in the same project)
- Not all are OSS

How Does A Project Get Started

- Someone writes something for fun
  or to solve a problem
- They publish the source code (github,
  sourceforge, Google code, FTP site...)
- Someone else discovers it, finds it useful
- They improve the software
- They publish their modified version,
  or send changes back to the original author

Getting The Software

- Download an archive of a release
  e.g. congruity-5.tar.bz2
  using a web browser or script
- Use a source code control system, e.g.
  `git clone git://git.code.sf.net/p/congruity/code`
- Build it, run it, etc.

Perhaps Something Is Wrong

- It crashes
- It generates the wrong result
- It's missing a feature you need
- There's no documentation
  - Very useful to fix this,
    and doesn't require coding ability

What To Do

- Report the problem;
  someone might fix it for you
- Otherwise, go fix it!
- You have the source code
- Once you've fixed the code, you want to contribute the change back to the project
- How? Send a patch

Communication & Help

- Different for each project
- Usually, primarily by email
  - Often centered around patches
- Wikis? Sometimes
- Bug trackers? Sometimes
- IRC (multi-user real-time chat)
- Web forums? Much less common

What Is A Patch

- A textual representation of a change you made to some software
- Delta, so much smaller than the original or modified version of the software
- Easy to send by email
- Perfectly computer readable
- Human readable
- Can be “applied” to the original software to end up with your modified version

Example Patch


```
diff --git a/drivers/regulator/palmas-regulator.c
           b/drivers/regulator/palmas-regulator.c
index 3c861d5f9245..93b4ad842901 100644
--- a/drivers/regulator/palmas-regulator.c
+++ b/drivers/regulator/palmas-regulator.c
@@ -970,7 +970,7 @@ static int palmas_regulators_probe(
                                                struct platform_device *pdev)
                PALMAS_SMPS12_CTRL_MODE_ACTIVE_MASK;

        pmic->desc[id].enable_reg =
-               PALMAS_BASE_TO_REG(PALMAS_LDO_BASE,
+               PALMAS_BASE_TO_REG(PALMAS_SMPS_BASE,
                        palmas_regs_info[id].ctrl_addr);
        pmic->desc[id].enable_mask =
                PALMAS_SMPS12_CTRL_MODE_ACTIVE_MASK;
```

Creating A Patch


```
cd some_git_checkout
vi path/to/file.c
git diff
```

or:

```
cp -r original edited
vi edited/path/to/file.c
diff -urN original edited
```

A Better Patch: Source Control


```
$ cd some_git_checkout
$ vi path/to/file.c
$ git commit -a -s
                (enter commit message in $EDITOR)
 1 file changed, 1 insertion(+), 1 deletion(-)
$ git format-patch -1
0001-regulator-palmas-fix-typo-in-enable_reg-
calc.patch
```

Process

- The next few slides detail the Linux Kernel process
- Other projects may well be lighter weight

Commit Messages

regulator: palmas: fix typo in enable_reg calculation

When setting up .enable_reg for an SMPS regulator, presumably we should call PALMAS_BASE_TO_REG(PALMAS_SMPS_BASE, ...) rather than using LDO_BASE. This change makes the LCD panel and HDMI work again on the NVIDIA Dalmore board anyway.

Fixes: 318dbb02b50c ("regulator: palmas: Fix SMPS enable/disable/is_enabled")

Fixes: dbabd624d4eec50b6 ("regulator: palmas: Reemove open coded functions with helper functions")

Signed-off-by: Stephen Warren <swarren@nvidia.com>

Signed-off-by

- S-o-b means something specific
- Understand it before you write it
- Essentially warrants that you have the right to contribute the code under the license
- http://developercertificate.org/

Patch Style

- Documentation/CodingStyle (Linux kernel)
- Check with (kernel or U-Boot):

```
$ ./scripts/checkpatch.pl 0001-foo.patch
total: 0 errors, 0 warnings, 8 lines checked


0001-regulator-palmas-fix-typo-in-enable_reg-
calc.patch has no obvious style problems and
is ready for submission.
```

Sending Patches


```
$ git send-email
  --to 'Liam Girdwood <lgirdwood@gmail.com>'
  --to 'Mark Brown <broonie@kernel.org>'
  --cc linux-kernel@vger.kernel.org
  *.patch

0001-...patch

…

Password for 'smtp://swarren@localhost:8587':

…

Result: 250 2.0.0 Ok: queued as 767D8E4103
```

Where To Send A Patch

- Different for each project
- Send to mailing lists for visibility
  and CC maintainers so notice the patch
- U-Boot: `doc/git-mailrc` (aliases for git send-email)

```
alias uboot  u-boot@lists.denx.de
alias u-boot uboot
alias arm            uboot, aaribaud ...
```

- Kernel:

```
$ ./scripts/get_maintainer.pl *.patch
Liam Girdwood <lgirdwood@gmail.com> (supporter:VOLTAGE AND CURRE...)
Mark Brown <broonie@kernel.org> (supporter:VOLTAGE AND CURRE...)
linux-kernel@vger.kernel.org (open list)
```

Patch Feedback

On 06/19/2014 10:08 AM, Stephen Warren wrote:

> On 06/19/2014 12:58 AM, Alexandre Courbot wrote:

…

>> > diff --git a/arch/arm/cpu/tegra-common/vpr.c
b/arch/arm/cpu/tegra-common/vpr.c

>> > +void config_vpr(void)

>> > + /* Turn off VPR */

>> > + writel(0x00000000, &mc->mc_video_protect_size_mb);

>> > + writel(0x00000001, &mc->mc_video_protect_reg_ctrl);

>

> Can we use a #define rather than "1" there, so we know what the bit

> means. Also "0" is as good as "0x00000000" and same for "1".

Patch Accepted

On 06/24/2014 05:01 AM, Mark Brown wrote:

> On Mon, Jun 23, 2014 at 02:53:25PM -0600, Stephen Warren wrote:

>> > From: Stephen Warren <swarren@nvidia.com>

>> >

>> > When setting up .enable_reg for an SMPS regulator, presumably we should

>> > call PALMAS_BASE_TO_REG(PALMAS_SMPS_BASE, ...) rather than using

>> > LDO_BASE. This change makes the LCD panel and HDMI work again on the

>> > NVIDIA Dalmore board anyway.

>

> Applied, thanks.

Where Are Patches Applied

- Simpler projects:
  - Patches checked directly into main branch
- Larger projects:
  - Each subsystem has a separate branch
  - Subsystem maintainer collects many patches
  - Periodically sends a “pull request” up the chain
  - These pull requests eventually make it into the main branch

Kernel Release Process

- Each release starts with a merge window
  - 2 weeks long
  - Pull requests merged during this time
  - “rc1” release made at end of merge window
- Followed by about 7 weeks of testing
  - Bug-fixes merged during this time
  - Each week a new “rc” release is created
- After “rc7” or “rc8”, the release is made

Questions?