Refactor strings – make everyone happy

Wolfram Sang

Consultant / Renesas Upstream Kernel Team

14.07.2016, LCJ16

The grand task[1]

Let's save memory!

...especially at run-time

[1] However, we are still in the fundamental research phase

Get the tools and metrics ready

How I measured

parse the diff

- good for measuring the magnitude mainly
- but saves all the compiling

bloat-o-meter

- doesn't look awfully much used
- minor fixes applied
- didn't show changes to constant strings
- two hacks for that floating around

How I measured

parse the diff

- good for measuring the magnitude mainly
- but saves all the compiling

bloat-o-meter

- doesn't look awfully much used
- minor fixes applied
- didn't show changes to constant strings
- two hacks for that floating around

Now three hacks are floating around.

coccinelle[2]

<3

[2] If you don't know it, introductory talk by Vaishali Thakkar tomorrow.

ninja-measure-bloat

```
for (commit_range)

  if not cached
    build objs before commit
    build objs after commit

  for (objs)
    bloat-o-meter before after

  print sum_commit

print sum_range
```

.config

To easily compare results, stay with one compiler and one config!

allyesconfig became pretty unusable

- can't be compiled for ARM
- builds lots of helpers first

a reasonable config for playing around

- `make defconfig` (x86-32)
- enable OLPC
- you now got `ACPI && OF`

Bad example from last year

```
/* UBIFS error messages */
#define ubifs_err(c, fmt, ...) \
\
    pr_err("UBIFS error (ubi%d:%d pid %d): %s: " fmt "\n", \
\
        (c)->vi.ubi_num, (c)->vi.vol_id, current->pid, \
        __func__, ##__VA_ARGS__)
```

Surprise!

Already fixed

- commit 3e7f2c5104a01f by Joe Perches
- saved more than 27K of text and data
- matches my numbers from last year
- still only half of what I wanted to do

Envisioning fs_err() and friends...

dev_err() and friends are awesome

- avoids the prefix hazzle from the example before
- consistent output, centralized
- works for every `struct device`
- I wanted something similar for filesystems

Quick tests showed that with fs_err() a few KB could be saved.

...but got side-tracked[3]

While evaluating various `dev_err()` usage patterns, I found one suspicious:

`dev_err()` combined with `__func__`?

- looks pretty useless once the error message is unique
- `__func__` is pretty expensive

This is an OK example (for now), but the rest can go:

```
dev_err(&client->dev, "%s: read error\n", __func__);
```

[3] or shall I say "main-tracked"?

A proof-of-concept for this

Let's apply this

- Removing all occurences removes easily >60K

  measured by parsing the >850K diff
- due to OK cases, needs to be manually reviewed
- did an example series for the RTC subsystem

  that's where I learned about OK cases
- 18 patches left, saved 156 byte/driver average, altogether saves >2.5K (5.1%)

Not send out yet, afraid of janitors.

Afraid of janitors?

Yes, a little...

- this work surely has janitorial parts
- *but only as a second step!*
- the first step is still refactoring

  what implies a lot of thinking, obviously
- but now, I can point people to this talk :)

Example: I saw many occasions where removing `__func__` was not enough. The whole error message should be removed.

For example: `usb_submit_urb()`

Promising candidate

- a lot of "returned %d" messages with little value
- gkh says all messages can go
- `usb_submit_urb()` prints a few messages, yet not on all exit paths
- this should be done first
- then, we can save $\approx$ 20K from $\approx$ 120 drivers

  Remember: 20K of just saying "failed" for this one function

Preparational changes in 4.9, removals in 4.10?

And pointing to more interesting issues

```
 exit:
-   retval = usb_submit_urb(urb, GFP_ATOMIC);
-   if (retval)
-     dev_err(dev, "%s - usb_submit_urb failed with result %d\n
-         __func__, retval);
+   usb_submit_urb(urb, GFP_ATOMIC);
 }
```

This isn't error handling, is it?

(Another side-path...)

Like pulling a thread from a ball of wool

Did I mention that some callers of `usb_alloc_urb()` print -ENOMEM messages which is totally unneeded and removing those saves 4K in 53 drivers?

Will be send out after v4.8-rc1 aiming for v4.9.

But why is `usb_submit_urb()` so often found?

But why is `usb_submit_urb()` so often found?

`usb-skeleton.c`

But why is `usb_submit_urb()` so often found?

`usb-skeleton.c`

Bummer!

- fixed it, too
- not much saved, but scales awesomely well

Apropros skeleton driver: Quiz Time!

```
static void skel_read_bulk_callback(struct urb *urb)

...

    dev_err(&dev->interface->dev,
        "%s - nonzero write bulk status received: %d\n",
        __func__, urb->status);
```

Who sees an issue here?

Apropros skeleton driver: Quiz Time!

```
static void skel_read_bulk_callback(struct urb *urb)

...

    dev_err(&dev->interface->dev,
        "%s - nonzero write bulk status received: %d\n",
        __func__, urb->status);
```

Who sees an issue here?

Since 2009. So much about error messages actually being read >:-)

Back to known land

Let's get one thing checkboxed

- I am not an USB expert, so hard to fix `usb_submit_urb()` on my own
- I consider myself an expert of the I2C subsystem, though
- let's fix something there
- something which can be a role model for other subsystems

Centralize *error when registering*

7 patches[4] to prepare `i2c_add_adapter`

- added 3 error messages
- improved 4 error messages

  some were really useless before
- 1 refactoring saving bytes already
- 1 resource leak bugfix(!)

  well, rarely used error paths are reviewed

Sent out, expected to land in v4.8.

[4] manually created, of course

Centralize *error when registering* II

Better consistency

- all drivers have common error messages
- more precise information provided than just -ERRNO

64 removals[5] across the tree

- saving now around 3K of variuos permutations of "registering failed"
- and more considering the future (i.e. scales well)
- $\approx$ 50 bytes/driver (half .rodata, half .text)

Will be send out after v4.8-rc1 aiming for v4.9. Branch is available.

[5] created with coccinelle, of course

Centralize *success when registering*

More difficult than expected

- drivers print various additional information
- some of them are not available to the I2C core
- some could be, but are not currently (e.g. bus speed)
- would need bigger refactoring, beyond the scope of this task
- for further activities, focus on error paths
- interesting viewpoint nonetheless

Summary

this time...

- about 30K of strings removed
- tools are ready and workflow is in place
- reference examples on the way to mainline

a *LOT* of work still to be done

- help is much appreciated
- it is not strictly janitorial, though
- focus on removing error strings first
- reviewing patches to improve coccinelle scripts takes most time

The End

Thank you for your attention!

Thanks to LF for funding!

Questions?

- Right here, right now...
- Later at the conference
- wsa@the-dreams.de

Branches can be found here:
git://git.kernel.org/pub/scm/linux/kernel/git/wsa/linux.git