Kernel Internship Report (Outreachy)
Successor of the Outreach Program for Women (OPW)

Julia Lawall (Inria/Irill/LIP6)
Shraddha Barke, Ioana Ciornei, Cristina-Gabriela Moraru, Janani Ravichandran, Ksenija Stanojević, Daniel Baluta

http://outreachy.org

October 4, 2016
What is Outreachy?

• Organized by the Software Freedom Conservancy
  – Formerly OPW, organized by Gnome.

• Goal: Get more women and other underrepresented groups into open source.

• Internship:
  – 3 months
  – $5,500 stipend
  – Paired with mentor

• Timing: May – August, December – March.
Who can apply for an internship?

- Women (cis and trans), trans men, and genderqueer people.

- Additionally, Outreachy is open to residents and nationals of the United States of any gender who are Black/African American, Hispanic/Latin@, American Indian, Alaska Native, Native Hawaiian, or Pacific Islander.

- Must be able to work full time.

- Can work remotely.

- Don’t have to be a student.
Which projects are involved?

- Recent kernel projects:
  - Summer 2016 (Round 12): Workqueue modernization, Coccinelle, IIO drivers, nftables, memory management latency tracing.

- Other projects:
  - Debian, GNOME, Mozilla, OpenStack, Wikimedia, etc.

Internships are financed by the project’s organization, or by industry sponsors.
Round 13 is open now!

- Application period: September 12 – October 17.
- Accepted interns announced November 8.
- Internship period: 3 months, December 6 - March 6.
- https://www.gnome.org/outreachy/
How to apply

- Pick a project
- Contact a mentor
- Contribute to the project
- Fill out an application
How to apply

• Pick a project
• Contact a mentor
• Contribute to the project
• Fill out an application
How to apply for the Linux kernel

- First patch tutorial: http://kernelnewbies.org/OutreachyIntro
- Clean up staging drivers
  - Learn about patch structure, coding style, tools.
- Small tasks from kernel mentors.
Contributions from applicants and interns

- ~15 participants in the application period for summer 2016
- 5 interns chosen
- 746 patches accepted overall from the 15 active applicants
  - 465 patches from the 5 accepted applicants
- 742 patches accepted overall from the 23 active applicants in summer 2015
Contributions from applicants and interns

- **Shraddha Barke**: #6 for Linux 4.4 in terms of patches (147)
- **Deepa Dinamani**: #9 for Linux 4.4 in terms of changed lines (7797)
- **Amitoj Kaur Chawla**: #5 for Linux 4.6 in terms of patches (117)
- **Bhaktipriya Shridhar**: #19 for Linux 4.6 in terms of patches (80)
- **Outreachy** was #9 for organizations for Linux 4.4 and #6 for Linux 4.6.
How can I help?

- Companies and individuals can:
  - Donate funds to support interns
  - Contact: outreachy-admins@gnome.org

- Kernel developers can:
  - Review patches
  - Volunteer as mentors
  - Contact: Julia Lawall <Julia.Lawall@lip6.fr>
Presentations from recent interns

- Shraddha Barke: Staging Drivers Cleanup
- Ioana Ciornei:
- Cristina-Gabriela Moraru: Extending the IIO Subsystem
- Janani Ravichandran: Memory Allocation Latency Tracing
- Ksenija Stanojević: Staging Drivers Cleanup

Julia.Lawall@lip6.fr
http://outreachy.org
outreachy-admins@gnome.org
Who am I?

- Electronics Student at BITS Pilani, India currently in final year.
- Winter Intern for Outreachy December - March 2015 round.
- Worked on cleaning up code in the staging directory of Linux Kernel with Greg Kroah Hartman.
- One of the most active developers in Linux v4.4 version - https://lwn.net/Articles/668870/
- Fantastic experience sending patches to the kernel!
- Currently working at TIFR, Mumbai as research assistant on Verification of concurrent programs for weak memory models.
How did I get involved?

- Ubuntu user for 3 years before Outreachy.
- Introduced to the working of Linux Kernel through Operating Systems course at university.
- Motivation to contribute as Linux is open source and impacts thousands of users and developers everyday.
- Outreachy is a great opportunity for students to begin contributing to open source projects.
The drivers and filesystems present in staging directory are those which are not ready to be merged into the main Linux kernel tree.

- Merge drivers out of staging after fixing checkpatch and sparse warnings.
dgap and gdm72xx

- Removed dgap and gdm72xx out of staging.
- No support for these drivers.
- [http://www.networkworld.com/article/2220370/4g/wimax-is-dead.html](http://www.networkworld.com/article/2220370/4g/wimax-is-dead.html)
Goldfish Driver

- Add DMA support to the driver using dma_alloc_coherent.
- Added error checking using dma_mapping_error for memory allocation errors during DMA mapping.
- Used the managed version dmam_alloc_coherent in appropriate places.
Other changes made

- Declared local functions as static.
- Removed unused calls to header files.
- Removed boilerplate code and added devm functions.
- Updated the function API and use macros in the kernel.
- Replaced wrapper functions with direct calls.
- Coccinelle scripts were used extensively to find and fix these issues.
Learning process during Outreachy

- How to use git effectively for version control
- How to successfully compile and build the kernel.
- How to read the kernel source code and understand the patch management process.
- How to interact with open source community.
- How to use tools such as Coccinelle, Smatch and Sparse for kernel code cleanup.
Thanks to the community!

- Linux Foundation, Outreachy and the entire Linux community.
- Greg Kroah Hartman, the most patient maintainer for being my mentor during the internship and accepting patches regularly.
- Julia Lawall, the most awesome and helpful person whom I still trouble when in trouble.
- Outreachy mentors and developers who helped in patch reviewing process.
- Past Outreachy and OPW interns.
Future Contribution

- Contribute to the cleanup of staging directory. Bugs are never ending and omnipresent!
- Develop Coccinelle semantic patches to fix issues in kernel code.
- Talks for developers on beginning to contribute to Linux Kernel - Workshop proposal for “Diving into Open source with Linux Kernel” selected for GHC India 2016, with Bhaktipriya Sridhar.
Thanks for listening

- A list of my accepted patches can be found here -
  https://git.kernel.org/cgit/linux/kernel/git/priv/github/staging.git/log/?qt=grep&q=shraddha+barke

- My internship blog - http://shraddhabarke.github.io

- Looking for a software development internship! :)}
Extending Industrial I/O Subsystem of Linux Kernel

Cristina-Gabriela Moraru
University POLITEHNICA of Bucharest

cristina.moraru09@gmail.com
Blog: kernelsense.wordpress.com

Mentors: Octavian Purdila
Daniel Baluta

04/10/2016

LinuxCon Europe 2016
Contribution

I. Get out of staging driver for 3-Axis Digital Compass: Honeywell HMC5843

II. Develop driver for humidity and temperature sensor: HopeRF TH06

III. Develop driver for digital potentiometer: Maxim MAX5487
I. 3-Axis Digital Compass: HMC5843

- Get driver for HMC5843 out of staging
- staging: drivers and filesystems that are not ready to be merged
- Issues solved:
  - Add sysfs attribute for bias configuration
  - Swap suspend() and resume() implementations
  - Coding style
  - Incomplete documentation
II. Humidity and Temperature Sensor: TH06
TH06 - Hardware

- TH06 communicates on I2C interface:
  - VCC
  - Ground
  - SDA (data bus)
  - Clock

- Extra hardware: Diolan to perform the conversion between I2C <-> USB
TH06 - Driver Development

- 2 registers: humidity & temperature => 2 read-only channels

Key points:
  - id definition
  - probe() - initialize channels and create sysfs entry for device
  - read_raw() - uses I2C kernel API and retrieves raw values / scale / offset

Similarity with Silicon Labs Si7020 driver

Finally, extended Si7020 for TH06 and Si7005 for TH02
III.

Digital Potentiometer: MAX5487
MAX5487 - Hardware

3 terminals:
- HIGH (H) -> Power Supply
- WIPER (W)
- LOW (L) -> Ground

The rotative slide controls the wiper position.

Wiper positions are given by the internal wiper registers.

Wiper registers are set via SPI interface.
MAX5487 - Driver Development

- 2 registers for wiper positions => 2 write-only channels
- SPI kernel API for device interaction: spi_write()
- 16 bit SPI commands:

<table>
<thead>
<tr>
<th>8 bits</th>
<th>8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>register address</td>
<td>value to be set</td>
</tr>
</tbody>
</table>

- value = index where the wiper is placed between HIGH and LOW terminals
- Support for MAX5487 / MAX5488 / MAX5489
- Accepted driver
Summary

- Added Linux Kernel support for:
  - HMC5853 3-axis digital compass
  - TH06, TH02 humidity and temperature sensors
  - MAX5487, MAX5488, MAX5489 digital potentiometers

- Blog: kernelsense.wordpress.com

- Looking forward to new open source challenges!

- Thanks to my mentors: Octavian Purdila and Daniel Baluta and to our program coordinator: Julia Lawall
Outreachy Internship - Memory Allocation Latency Tracing

JANANI RAVICHANDRAN
janani.rvchndrn@gmail.com
Motivation

- Users suffer excessive latencies during memory allocation
- Developers resort to some ad hoc methods to figure out the problem on production systems

Goal:
Use the tracing infrastructure and a userspace script to obtain information on where time was spent when memory allocation takes time > threshold
Memory Management Basics

- **Nodes and Zones**
  - Each memory bank is a node
  - Each node is divided into blocks representing ranges within memory zones

- **Memory pages**
  - System memory divided into fixed-size chunks called page frames

- **Order**
  - Allocation is done is powers of 2.
    Eg: Zeroth order allocation gives $2^0 = 1$ page,
    first order $2^1 = 2$ pages and so on
Memory Allocation - A Simplified View

Allocation request

Can request be fulfilled?
Memory Allocation - A Simplified View

Allocation request

Can request be fulfilled?

Y

Pointer to the first page returned
Memory Allocation - A Simplified View

Allocation request

Can request be fulfilled?

Zone reclaim successful?

Pointer to the first page returned

Y

N

Zone reclaim frees up unmapped file backed pages and slab pages.
Memory Allocation - A Simplified View

- Allocation request
  - Can request be fulfilled?
    - Yes
      - Pointer to the first page returned
    - No
      - Zone reclaim successful?
        - Yes
          - Zone reclaim frees up unmapped file backed pages and slab pages.
        - No
Memory Allocation - A Simplified View

Zone reclaim frees up unmapped file backed pages and slab pages.

Direct reclaim is writing dirty pages in memory to disk.

Compaction is moving used pages to get contiguous free pages.
Memory Allocation - A Simplified View

- Allocation request
- Can request be fulfilled?
  - Y: Pointer to the first page returned
  - N: Zone reclaim successful?
    - Y: High latencies
    - N: Direct reclaim and compaction

Zone reclaim frees up unmapped file backed pages and slab pages.

Direct reclaim is writing dirty pages in memory to disk.

Compaction is moving used pages to get contiguous free pages.
**Function Graph Tracer**

Prerequisite - must have ftrace configured in the kernel. When ftrace is configured, a directory called tracing is created within debugfs. Typically, /sys/kernel/debug/tracing.

- Ftrace is an internal tracing infrastructure built into the kernel
- Comprises many tracers but we focus on the function graph tracer
- Function graph tracer probes functions on both entry and exit and draws a graph of function calls.

Sample output:

```plaintext
# tracer: function_graph
#
#     TIME        CPU  TASK/PID         DURATION                  FUNCTION CALLS
#      |          |     |    |           |   |                     |   |   |   |
# 7566.006147 |   0)    Xorg-852    |               |      shrink_zone() {
# 7566.006148 |   0)    Xorg-852    |               |        shrink_zone_memcg() {
# 7566.006149 |   0)    Xorg-852    | 3.672 us    |          shrink_inactive_list();
# 7566.006153 |   0)    Xorg-852    | 1.089 us    |          shrink_active_list();
# 7566.006156 |   0)    Xorg-852    | 7.805 us    |        }
# 7566.006156 |   0)    Xorg-852    | + 38.336 us  |      }
```
Tracepoints

- Lightweight hooks that pass an arbitrary number of parameters
- Can be placed in places where some context information is desired

Sample:

```
0) compiz-2542 |               | /* mm_shrink_slab_start: scan_shadow_nodes+0x0/0x50 ffffffff81e6cec0: nid: 0 objects to
shrink 126 gfp_flags GFP_HIGHUSER_MOVABLE|__GFP_COLD pgs_scanned 8 lru_pgs 1217 cache items 252 delta 3 total_scan 126 */

0) compiz-2542 |               | /* mm_shrink_slab_end: scan_shadow_nodes+0x0/0x50 ffffffff81e6cec0: nid: 0 unused scan
count 126 new scan count 126 total_scan 126 last shrinker return val 0 */
```

Why use tracepoints?

- Function graph tracer only gives the duration of function calls.
- Tracepoints help analyse latencies better by exposing some additional information
Using both Function Graph and Tracepoints to analyse latencies

- There is a way to have only functions exceeding a certain user-specified threshold displayed in the function graph output (look up tracing_threshold!)
- No such filter for tracepoints.
- Would be great to display context information from tracepoints only when the threshold is exceeded, to reduce disk I/O!
- My work:
  - A shell script to set up functions in the memory allocation path, to be traced by the function tracer and enable existing tracepoints of interest in the kernel.
  - Another script to parse the output of ftrace to write to disk only the information associated with high latencies.

<TO DO: Insert link to the scripts>
Future work

- Add more tracepoints to get more information.
- Get it merged with the kernel.
Experience as an Outreachy Intern

- Learnt about how Linux kernel development works
- Had great mentors who were always there. Thank you, Rik and Julia!
- Learnt more about operating systems, version control, etc
- Spoke to some really inspiring former interns
- Had a tonne of fun :D

Would 100% recommend Outreachy!
References

The tracing infrastructure:

http://lwn.net/Articles/370423/ - Secrets of the Ftrace function tracer

https://lwn.net/Articles/410200/ - Trace-cmd - A front end for Ftrace


https://lwn.net/Articles/365835/ - Debugging using Ftrace - 1

https://lwn.net/Articles/366796/

And of course, the kernel documentation!

More about my work at tuxmex@wordpress.com.
Danke! :)

About Me

- Ksenija Stanojević. I live in Belgrade (Serbia)

- Outreachy Intern for Linux Kernel from December 2015 – March 2016

- Worked on the Staging drivers cleanup project: split existing mxs-lradc driver.
i.MX28
LRADC hardware

- The sixteen-channel low-resolution analog-to-digital converter block is used for voltage measurement
- 12 bit of resolution
- 8 virtual channels can be mapped to any of the 16 channels
- 4-wire/5-wire touch-screen controller
- Keypad (button) detect and button-detection interrupt circuit are implemented in the controller
MXS-LRADC

MFD

ADC
- General purpose ADC readings
- Battery voltage measurements
- Die temperature measurement
  - IRQ handling
  - Trigger handling

TOUCHSCREEN
- Coordinate measurements
- Touch-detection
- Pressure detection
  - IRQ handling
MXS-LRADC

- MFD
  - drivers/mfd/mxs-lradc.c
  - include/linux/mfd/mxs-lradc.h
- Analog-to-Digital-Converter
  - drivers/iio/adc/mxs-lradc-adc.c
- Touchscreen
  - drivers/input/touchscreen/mxs-lradc-ts.c
MFD

• Register devices via MFD API:

```c
int devm_mfd_add_devices(struct device *dev, int id,
                         const struct mfd_cell *cells, int n_devs,
                         struct resource *mem_base,
                         int irq_base, struct irq_domain *domain)
```
MFD

- Device represented with struct mfd_cell

```c
static struct mfd_cell mx28_cells[] = {
    {
        .name = "mxs-lradc-adc",
        .resources = mx28_adc_resources,
        .num_resources = ARRAY_SIZE(mx28_adc_resources),
    },
    {
        .name = "mxs-lradc-ts",
        .resources = mx28_touchscreen_resources,
        .num_resources = ARRAY_SIZE(mx28_touchscreen_resources),
    }
};
```
• Define resources: I/O memory, IRQ

static struct resource mx28_adc_resources[] = {
    DEFINE_RES_MEM(MXS_LRADC_BASE, 0x1fff),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_TRESH0_IRQ, "mxs-lradc-thresh0"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_TRESH1_IRQ, "mxs-lradc-thresh1"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_CH0_IRQ, "mxs-lradc-channel0"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_CH1_IRQ, "mxs-lradc-channel1"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_CH2_IRQ, "mxs-lradc-channel2"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_CH3_IRQ, "mxs-lradc-channel3"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_CH4_IRQ, "mxs-lradc-channel4"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_CH5_IRQ, "mxs-lradc-channel5"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_BUTTON0_IRQ, "mxs-lradc-button0"),
    DEFINE_RES_IRQ_NAMED(MX28_LRADC_BUTTON1_IRQ, "mxs-lradc-button1"),
};
MXS-LRADC drivers probing

• Get platform data
  - struct mxs_lradc *lradc = dev_get_platdata(dev);

• Get resources
  - IRQs
    IRQsirq =
    platform_get_irq_bynname(pdev, irq_name[i]);:
  - I/O memory:
    platform_get_resource(pdev, IORESOURCE_MEM, 0);
Thank you