Kernel Internship Report (Outreachy)

Successor of the Outreach Program for Women (OPW)

Julia Lawall (Inria/Irill/LIP6)

Ebru Akagündüz, Roberta Dobrescu, Aya Mahfouz, Iulia Manda,
Cristina Georgiana Opriceana, Greg Kroah-Hartman, Laurent Pinchart

http://outreachy.org

October 7, 2015

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. (New!)
- 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 2015 (Round 10): Coccinelle, Media Controller Virtual Driver, Modernize Linux Wireless drivers, Full Dynamic Ticks, IIO dummy driver.
  - Winter 2014 (Round 9): Kernel tinification, Coccinelle, Surviving Year 2038, IIO staging drivers cleanup, Khugepaged swap readahead
- Other projects:
  - Debian, GNOME, Mozilla, OpenStack, Wikimedia, etc.

Internships are financed by the project's organization,
or by industry sponsors.

Round 11 is open now!

- Application period: September 29 – November 2.
- Accepted interns announced November 16.
- Internship period: 3 months, December 7 - March 7.
- 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

- 23 participants in the application period for summer 2015
- 17 submitted eligible applications
- 5 interns chosen
- 746 patches accepted overall from the 23 participants
  - 5523 lines added, 6324 lines removed
- 287 patches from the 5 accepted applicants during the application period
  - 2163 lines added, 2650 lines removed
- Intern Aya Mahfouz was the 5th most active contributor to Linux 4.1 in terms of patches and Outreachy was the 8th most active organization.

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

- Vaishali Thakkar: Coccinelle
- Vatika Harlalka: Full Dynticks
- Tapasweni Pathak: Faults in Linux
- Ebru Akagündüz: Improving THP Collapse Rate
- Roberta Dobrescu: IIO staging drivers cleanup
- Aya Mahfouz: Migration of Orinoco
- Iulia Manda: Linux Kernel Tinification
- Cristina Georgiana Opriceana: IIO Dummy Driver

Julia.Lawall@lip6.fr
http://outreachy.org
outreachy-admins@gnome.org

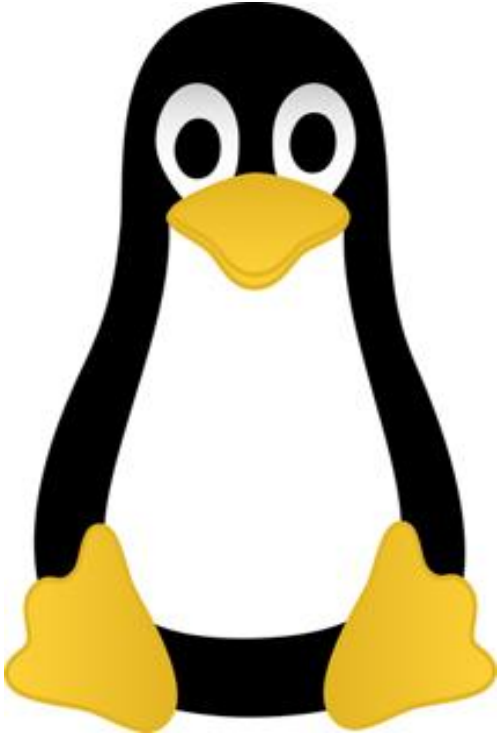
Thanks to Sarah Sharp!

Vaishali Thakkar

- **Project:** Coccinelle
- **Mentor:** Julia Lawall
- **Goal:** Replace out of date API uses with modern equivalents using Coccinelle
- **What I did:**
  - Worked on various API functions/macro s like timer API functions, ether device API functions, boilerplate code of module init/exit, resource managed functions (devm functions) etc
  - Wrote common semantic patches to detect/solve above cases
- **Total number of patches accepted:** 120 +
- **Future plans:** Looking for a full-time job in kernel development area
- **Contact:** vthakkar1994@gmail.com

1

HELLO!



Outreachy Presentation

Full Dynticks Project

Vatika Harlalka, Intern!

My mentors: Preeti Murthy

and Frédéric Weisbecker :D

2

BASIC IDEA

What are clock ticks?

To facilitate periodic functions like scheduling loads, expiring timers etc,
- *system timer or*
- ***programmable interrupt timer*** (PIT).

Every hit of this timer is known as a ***tick*.**

Ways of managing scheduling of ticks on a system:

- **CONFIG_HZ_PERIODIC**: Never omit ticks
- ***CONFIG_NO_HZ_FULL***: Omit on CPUs that have one/zero runnable task (***nohz_full CPUs***).
- **CONFIG_NO_HZ_IDLE**: Omit on idle CPUs.

3

THE PROJECT

How can we reduce OS jitter?

The idea is to delay ticks whenever possible!

On tick fire, expired timers are executed.

If a *non-pinned timer* is executed on a full dynticks CPU it will disturb the running process.

The project involved **affining these timers to appropriate online housekeeping CPUs.**

This is of interest in particular to real-time Linux users and also can be of great benefit to HPC workloads where there is only one task running.

THANKS!

Any questions?

You can email me at

- vatikaharlalka@gmail.com

or read my blog!

- vatikaharlalka.wordpress.com

Thank you Julia for presenting on my behalf :)

Faults In Linux

Tapasweni Pathak

Julia Lawall

Nicolas Palix

In 2001

- Chou et al. published a study of common faults in Linux versions 1.0 through 2.4.1

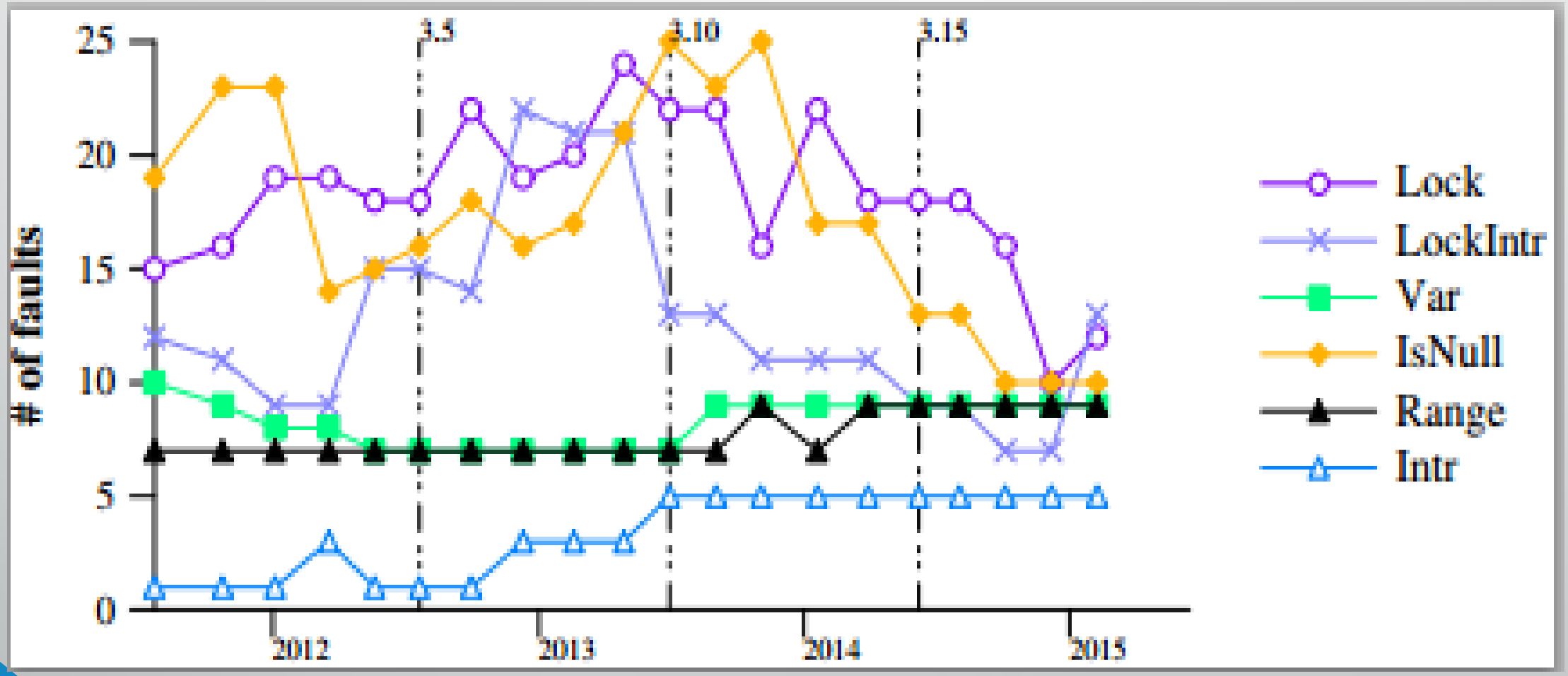
In 2012

- Palix et al. extended this work to cover the 2.6.x versions of the Linux Kernel

In 2014

- Tapasweni Pathak extended this by applying the same analysis to the Linux 3.x versions

Findings?


3.5
3.10
3.15
25
20
15
10
5
0
# of faults
2012
2013
2014
2015
Lock
LockIntr
Var
IsNull
Range
Intr

Improving THP Collapse Rate

Ebru Akagündüz
OPW, Linux Kernel Intern
ebru.akagunduz@gmail.com

Mentor: Rik van Riel

About me:

- Computer Engineer (graduation date: June, 2014)
- Junior System Admin
- OPW, Linux Kernel Intern - Round 9
- Contributor free software projects
- Likes to discover new things

Project aim:

- To improve THP collapse rate
  - ➔ What is THP?
  - ➔ What is benefit of it?
  - ➔ Why do we want to improve its collapse rate?

Before making changes ..:

- Start reading from do_page_fault()
- Detect function that -> collapses pages into a THP

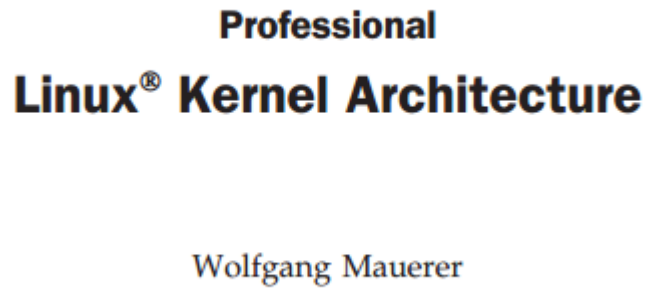
  -> splits pages to normal sized

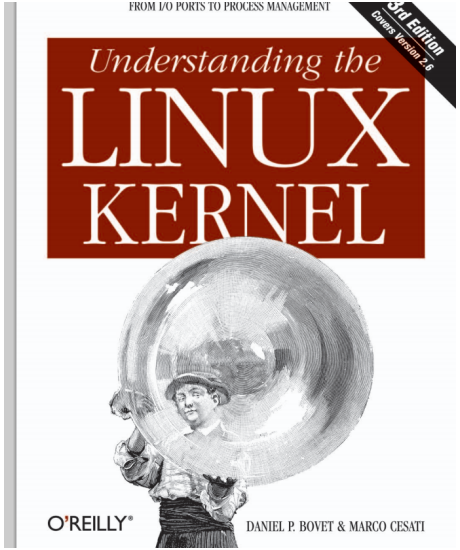
- How is a page swapped out?
- How is a page swapped in?

…. ….

…. And more questions needed to be answered :)

How to answer the questions?:





Follow call traces and keep:

```
do_page_fault()
  __do_page_fault()
    handle_mm_fault()
      __handle_mm_fault()
        __do_huge_pmd_anonymous_page()
```

```
do_swap_page()
        pte_to_swp_entry()
        lookup_swap_cache()
        if (!page)
                swapin_readahead()
```

```
add_to_swap()
        get_swap_page()
                scan_swap_map()
        if(PageTransHuge(page))
                if (split_huge_page_to_list(page, list))
                        swapcache_free()
        add_to_swap_cache()
```

How did I start coding?

- Detect where I should make changes?
- Be sure what changes are needed?
- Return the functions that I already have read before

Testing process?

- Examining oops messages
- Using gdb & perf
- Creating tracepoints

Contributions:

- Incorparating read-only pages & zero pages into THP
- khugepaged swapin readahead
- documentation

Achievements Through OPW:

- Learning basic functions of memory management
- Improving knowledges about the kernel
- Working from remote
- Working with most successful developers around world
- Getting accepted patches in upstream

Thanks!

IIO staging drivers cleanup

Roberta Dobrescu, Linux Kernel intern

<roberta.dobrescu@gmail.com>

FOSS Outreach Program for Women, Round 9

Mentors: Daniel Băluță, Octavian Purdilă

Why Linux Kernel internship?

- using Linux since my first year at the university
- having userspace experience with Linux
- some knowledge about Linux internals
- desire to learn how to hack into the kernel

Linux Kernel internship

- project: **IIO staging drivers cleanup**
- adapt the code to the Linux Kernel coding style
- change drivers to use proper IIO sysfs attributes
- main target: **Intersil ISL29018** digital ambient light and proximity sensor

First steps

- getting familiar with IIO Subsystem
- launched iiobits.wordpress.com blog

What is IIO?

- kernel subsystem for analog to digital or digital to analog converters and related hardware
- developed since 2009 by Jonathan Cameron and linux-iio community
- a great community

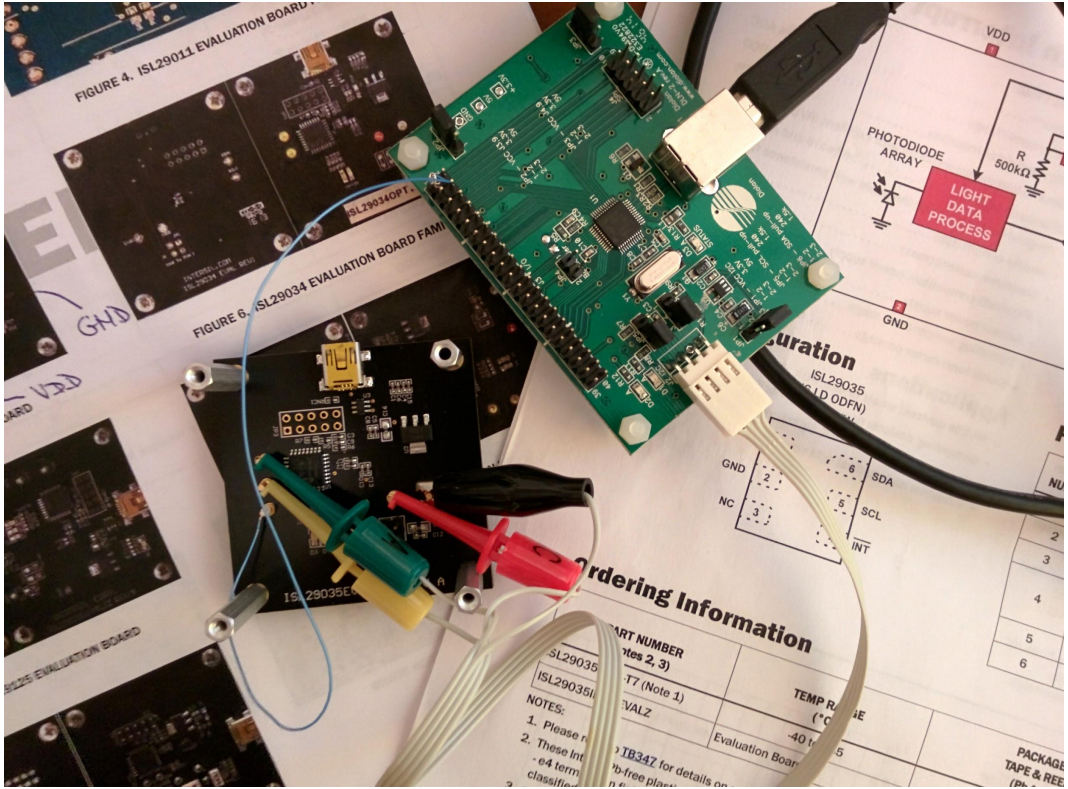
What I did
- first part of my internship -

- small fixes and cleanup patches
  - annotate Kconfig entries with the resulting module name
  - removing redundant warning messages
  - fix char unsigned order in ad8366 driver etc.
  - increased the sleep time for some not so time critical operations etc.

What I did
- second part of my internship -

- move IIO userspace applications out of staging
- getting familiar with isl29018 light sensor
- refactor isl29018 driver code to use standard IIO attributes in order to be moved out of staging

What I did
- second part of my internship



What I learned

- how to write code according to kernel coding style
- improve C and git skills
- how IIO Subsystem works
- being part of the great IIO community

After internship ended

- continued to contribute to Linux Kernel
- helped new Outreachy applicants to learn more about IIO
- looking forward to new opportunities

Many thanks to:

- **Daniel Băluță** and **Octavian Purdilă** for being great mentors
- **Jonathan Cameron** and **linux-iio** community
- **Greg KH** and all the other kernel developers that helped us during the application period
- **Sarah Sharp** and **Julia Lawall** for coordinating the Linux kernel internships
- **GNOME** and **Linux Foundation**

Modernize Orinoco

Aya Mahfouz

Outreachy Round 10

Sailboat on the Nile, Cairo, Egypt: Own work

Outreachy

- Applied to rounds 9 and 10
- Patches: cleanup and Y2038
- Accepted round 10
- Project: Modernize Linux wireless drivers
    - migrate Orinoco to cfg80211
    - remove WEXT from mainline kernel
- Mentors: Jes Sorensen and Kyle McMartin

WEXT

iwconfig	wpa_supplicant
Userspace	
ioctl	iw_handler
Wireless Extensions	
Linux Kernel	

cfg80211 Overview

- Revamp Wireless stack
- remove uses of ioctl
- userspace
  - nl80211 : netlink
  - stop micromanagement!
  - create classes of functions

cfg80211 Overview

iw	wpa_supplicant
nl80211	
Userspace	
cfg80211	
Linux Kernel	

cfg80211 Overview

```
struct wiphy {
u8 perm_addr[ETH_ALEN];
struct mac_address *addresses;
u8 max_scan_ssids;
struct ieee80211_supported_band *bands[IEEE80211_NUM_BANDS];
struct device dev;
char priv[0] __aligned(NETDEV_ALIGN);
}
```

cfg80211 Overview

```
01. struct cfg80211_ops {
02. int(*suspend)(struct wiphy *wiphy,
03. struct cfg80211_wowlan *wow);
04. int(*scan)(struct wiphy *wiphy,
05. struct cfg80211_scan_request *request);
06. int(*connect)(struct wiphy *wiphy, struct net_device *dev,
07.    struct cfg80211_connect_params *sme);
08. }
```

Orinoco in mainline kernel

WEXT ↔ Orinoco ↔ cfg80211

- WEXT code in Orinoco
  - iw_handler: wext.c
  - Notification functions w/ ioctl: main.c
- cfg80211 code in Orinoco
  - cfg.c

Orinoco W/ cfg80211

```
01. const struct cfg80211_ops orinoco_cfg_ops = {
02. .change_virtual_intf = orinoco_change_vif,
03. .connect = orinoco_cfg80211_connect,
04. .disconnect = orinoco_cfg80211_disconnect,
05. .join_ibss = orinoco_cfg80211_join_ibss,
06. .leave_ibss = orinoco_cfg80211_leave_ibss,
07. .set_monitor_channel = orinoco_set_monitor_channel,
08. .scan = orinoco_scan,
09. .set_wiphy_params = orinoco_set_wiphy_params,
10. };
```

Orinoco W/ cfg80211

```
01. const struct cfg80211_ops orinoco_cfg_ops = {
02. .change_virtual_intf = orinoco_change_vif,
03. .connect = orinoco_cfg80211_connect,
04. .disconnect = orinoco_cfg80211_disconnect,
05. .join_ibss = orinoco_cfg80211_join_ibss,
06. .leave_ibss = orinoco_cfg80211_leave_ibss,
07. .set_monitor_channel = orinoco_set_monitor_channel,
08. .scan = orinoco_scan,
09. .set_wiphy_params = orinoco_set_wiphy_params,
10. };
```

Concluding Remarks

- Orinoco supports many cards
- Simple hardware
- Only Agere-based hardware found
- Orinoco using cfg80211
  - new functions implemented for cfg80211_ops
  - some functions need more capable hardware
- Debug issues with calls to firmware

Thank you!

Arrival piazza, Sahl Hasheesh, Hurghada, Egypt: Source: http://communitytimes.me/wp-content/uploads/2013/12/piazza2.jpg

Linux Kernel Tinification

Iulia Mandă
Mentor: Josh Triplett

Context

- Size +10%/year for the last 10 years
- Embedded systems have limited resources and specific needs
- Too many functionalities in the kernel
- Boot time and power consumption are increasing

Patches

- Compile out multiuser support
- Kernel parameters at build time
- Syscall elimination
- Out of line inappropriate inline functions
- Detect changes in .rodata sections

Methods

- **Increase configurability**
- GCC size optimizations
- Automatic elimination of unused code

Methods

- Increase configurability
- **GCC size optimizations**
- Automatic elimination of unused code

Methods

- Increase configurability
- GCC size optimizations
- **Automatic elimination of unused code**

Patches

- **Compile out multiuser support**
- Kernel parameters at build time
- Syscall elimination
- Out of line inappropriate inline functions
- Detect changes in .rodata sections

Compile out non-root users

- Target application running directly out of init process
- Compile out non-root users, groups, capabilities and the corresponding syscalls
- Tested on Linux kernel 3.19, i386 arch
- Image size decreased by ~25KB

Patches

- Compile out multiuser support
- **Kernel parameters at build time**
- Syscall elimination
- Out of line inappropriate inline functions
- Detect changes in .rodata sections

Build time kernel parameters

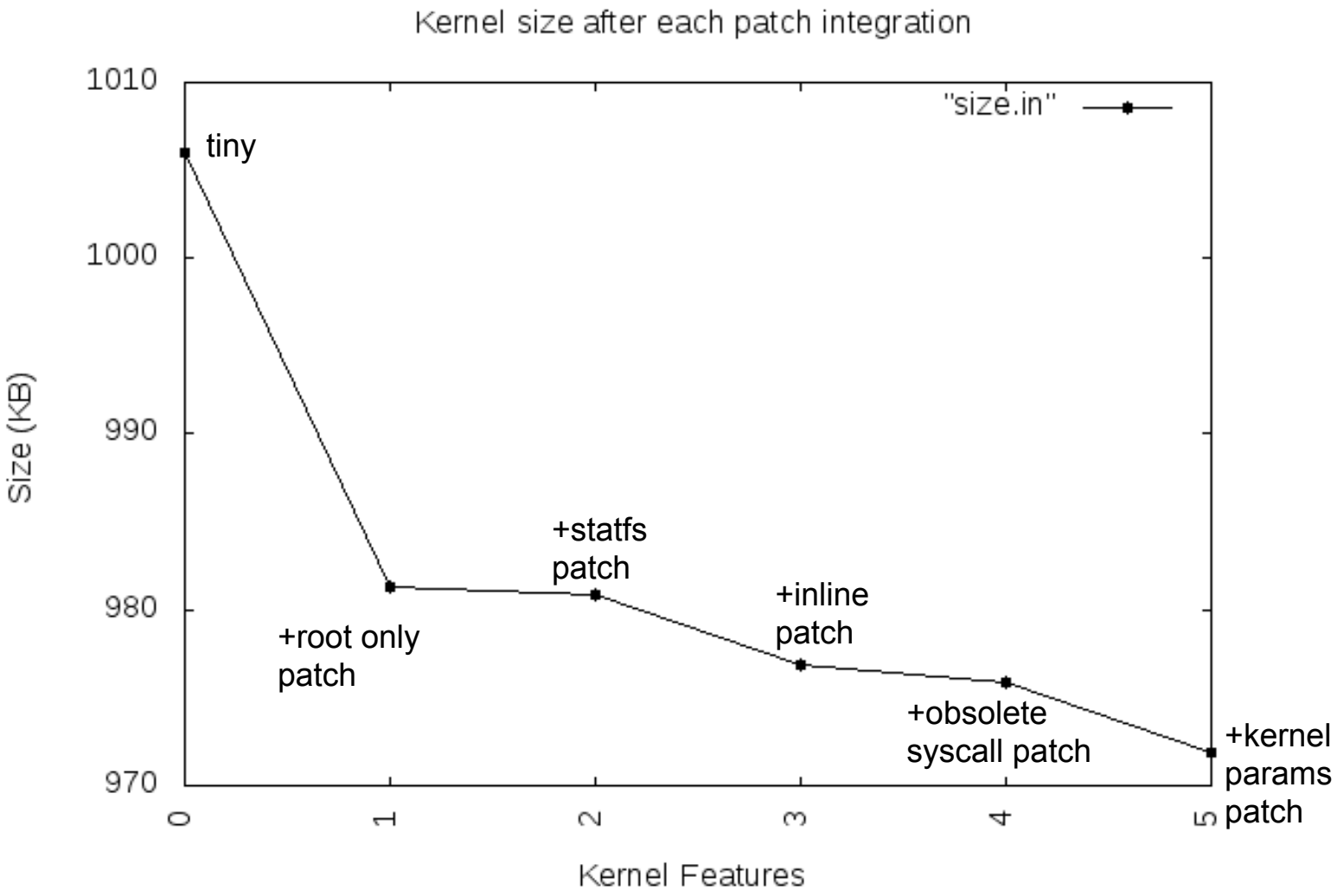
- We know exactly how to configured kernel
- Result: constant propagation and dead-code elimination
- No parsing of kernel arguments (~4KB)
- Set initcall_debug parameter to false → 400 bytes less

Patches

- Compile out multiuser support
- Kernel parameters at build time
- **Syscall elimination**
- Out of line inappropriate inline functions
- Detect changes in .rodata sections

Minimum number of syscalls

- Get system calls used in a specific user space application
- Compile in the kernel only the corresponding handlers


Kernel size after each patch integration
"size.in"
Size (KB)
1010
1000
990
980
970
0
1
2
3
4
5
Kernel Features
tiny
+root only patch
+statfs patch
+inline patch
+obsolete syscall patch
+kernel params patch

Results

- Decreased image size: 31KB total
- Faster boot time: 0.85s, 269 features (tiny)

  vs.

  1.48s, 532 features (default)
- Faster build time: 1.43m (tiny) vs. 13.23 (default)

Wishful Thoughts

- Test these features on different HW configs
- Dead code elimination
- Focus on **generalization** and **automation** of the reduction process
- https://tiny.wiki.kernel.org/projects

OPW experience

- First remote project
- Community feedback
- Great mentorship

IIO Dummy Driver

Linux Kernel Internship – May-Aug 2015

Cristina Georgiana Opriceană
Mentors: Daniel Băluță, Octavian Purdilă

University POLITEHNICA of Bucharest

Who am I?

Kernel Newbies

- Graduated CS in 2015
- Computer Network Security Master's Degree Student
- No Open Source contributions before

`Interests: Computer Networks, Operating Systems`

Outreachy Program

First Patches & Apply period

"Do you train hard in the off-season?
- No, it's all talent, I don't work, I just sit on the couch .. "

R. Federer, AO 2010

The IIO Dummy Driver

- Industrial I/O Subsystem

```
linux/drivers/staging/iio/iio_dummy_evgen.c
linux/drivers/staging/iio/iio_simple_dummy.c
linux/drivers/staging/iio/iio_simple_dummy_events.c
...
```

- reference/template IIO driver
- testing drivers in no real hardware environment

Dummy Driver Project Requirments

IIO Dummy Driver

"The goal of this project is to move IIO dummy driver out of staging. The major challenge in order to make this ready is **interrupt emulation** . The way evgen module generates interrupts greatly restricts their use and is far from clean. There is also work on the device creation part. Current approach where we use a module parameter controlling how many instances are created is pretty ugly."

Dummy Driver Project Requirments

IIO Dummy Driver

"The goal of this project is to move IIO dummy driver out of staging. The major challenge in order to make this ready is interrupt emulation. The way evgen module generates interrupts greatly restricts their use and is far from clean. There is also work on the **device creation part** . Current approach where we use a module parameter controlling how many instances are created is pretty ugly."

Rethink interrupt generation

- **GOAL**: Simulate real hardware interrupts

```
@@ -169,8 +192,9 @@ static ssize_t iio_evgen_poke(struct device *dev,
        iio_evgen->regs[this_attr->address].reg_id   = this_attr->address;
        iio_evgen->regs[this_attr->address].reg_data = event;

+       iio_evgen->handler.irq = iio_evgen->base + this_attr->address;
        if (iio_evgen->enabled[this_attr->address])
-               handle_nested_irq(iio_evgen->base + this_attr->address);
+               irq_work_queue(&iio_evgen->handler.work);
```

Calls only the threaded part

Rethink interrupt generation

- **GOAL**: Simulate real hardware interrupts

```
@@ -169,8 +192,9 @@ static ssize_t iio_evgen_poke(struct device *dev,
        iio_evgen->regs[this_attr->address].reg_id   = this_attr->address;
        iio_evgen->regs[this_attr->address].reg_data = event;

+       iio_evgen->handler.irq = iio_evgen->base + this_attr->address;
        if (iio_evgen->enabled[this_attr->address])
-               handle_nested_irq(iio_evgen->base + this_attr->address);
+               irq_work_queue(&iio_evgen->handler.work);
```

Calls only the threaded part

Replace by irq_work infrastructure

- **OUTCOME**: Call top & bottom handlers from hardware interrupt context

Configfs and device generation

- **GOAL**: Automatically register new device instances

- configfs is being introduced into iio
- allows creating and destroying objects in kernel
- extended to it the dummy driver

USERSPACE **KERNELSPACE**

What have I learnt?

- Integrate own code with existent code
- Apply operating systems concepts I've read about
- Get feedback from the community
- Invaluable mentorship experience
- Fun :)