What can possibly go wrong :)

/by Konrad Zapalowicz

Purpose of this talk

Agenda

What can possibly go wrong :)
/purpose of this talk

Although being fun to do and quite easy to start the Linux kernel development is considered hard, time consuming and it is said that it requires access to hardware. With this in mind many people might hesitate to try it out. In this presentation Konrad is going to use himself, a newbie contributor, as an example of how easy it is to start what did he learn (during the Eudyptula challenge and so far) and how this knowledge can be used to submit patches and participate in the upstream kernel development.

What can possibly go wrong :)
/Agenda

WHOAMI

The Myths about Linux kernel programming

How to start (ways to contribute)

The things that I wish I knew before I started

WHOAMI

Konrad Zapalowicz

Software Engineer at Cybercom Poland

Linux Enthusiast

Recent Linux Kernel contributor

Eudyptula Challenge Finalist

Some say that
/myths about Linux Kernel programming

Opinions on Linux Kernel programming

Some say that
/myths about Linux Kernel programming

Linux Kernel programming is considered hard and requires speciall skills

Some say that
/myths about Linux Kernel programming

Linux Kernel programming is considered hard and requires speciall skills

- [PATCH v3 17/24] media: imx: Add CSI subdev driver
- [RFC] vmbus: lockless ring write

Some say that
/myths about Linux Kernel programming

Linux Kernel programming is considered hard and requires speciall skills

Linux Kernel programming requires access to special hardware

Some say that
/myths about Linux Kernel programming

Linux Kernel programming is considered hard and requires speciall skills

Linux Kernel programming requires access to special hardware

Linux Kernel programming is pointless because all drivers had been written

Some say that
/myths about Linux Kernel programming

Linux Kernel programming is considered hard and requires speciall skills

Linux Kernel programming requires access to special hardware

Linux Kernel programming is pointless because all drivers had been written

Linux Kernel programming is time consuming

Here comes the reality
/myths explained

Linux Kernel programming is considered hard and requires speciall skills

Linux Kernel programming requires access to special hardware

Linux Kernel programming is pointless because all drivers had been written

Linux Kernel programming is time consuming

Here comes the reality
/myths explained

Linux Kernel programming is considered hard and requires speciall skills

Linux Kernel programming requires access to special hardware

Linux Kernel programming is pointless because all drivers had been written

Linux Kernel programming is time consuming

Here comes the reality
/myths explained

Linux Kernel programming is considered hard and requires speciall skills

Linux Kernel programming requires access to special hardware

Linux Kernel programming is pointless because all drivers had been written

Linux Kernel programming is time consuming

Here comes the reality
/myths explained

Linux Kernel programming is considered hard and requires speciall skills

Linux Kernel programming requires access to special hardware

Linux Kernel programming is pointless because all drivers had been written

Linux Kernel programming is time consuming

Here comes the reality
/myths explained

Linux Kernel programming is about C language

Linux Kernel programming can be done on x86

Linux Kernel programming is not only about writing drivers

Linux Kernel programming is when you have time for it, no stress

Ways of contributing

/how to start the journey

So we know that this is doable,
but how to start the journey

Ways of contributing

/how to start the journey

Improve the code quality

Ways of contributing

/how to start the journey

Improve the code quality

Do missing parts (the TODOs)

Ways of contributing

/how to start the journey

Improve the code quality

Do missing parts (the TODOs)

Work in the drivers/staging area

Ways of contributing

/how to start the journey

Improve the code quality

Do missing parts (the TODOs)

Work in the drivers/staging area

Fix Kernel oops (if you find one)

Improve code quality
/the scripts/checkpatch.pl tool

```
konrad in linux-mainline on master
% scripts/checkpatch.pl --strict --terse -f ./drivers/dma/mmp_pdma.c

./drivers/dma/mmp_pdma.c:335: CHECK: Please don't use multiple (...)
./drivers/dma/mmp_pdma.c:439: WARNING: void function return (...)
./drivers/dma/mmp_pdma.c:815: WARNING: else is not generally (...)
./drivers/dma/mmp_pdma.c:898: WARNING: Missing a blank line (...)
./drivers/dma/mmp_pdma.c:1057: WARNING: line over 80 characters
total: 0 errors, 4 warnings, 1 checks, 1126 lines checked
```

Improve code quality
/the scripts/checkpatch.pl tool

```
konrad in linux-mainline on master
% scripts/checkpatch.pl --strict --terse -f ./drivers/dma/mmp_pdma.c

./drivers/dma/mmp_pdma.c:335: CHECK: Please don't use multiple (...)
./drivers/dma/mmp_pdma.c:439: WARNING: void function return (...)
./drivers/dma/mmp_pdma.c:815: WARNING: else is not generally (...)
./drivers/dma/mmp_pdma.c:898: WARNING: Missing a blank line (...)
./drivers/dma/mmp_pdma.c:1057: WARNING: line over 80 characters
total: 0 errors, 4 warnings, 1 checks, 1126 lines checked
```

Improve code quality
/the sparse tool

```
konrad in linux-mainline on master
% make C=1 M=drivers/staging/dgnc

(now the sparse tool is being run for each file that is compiled)

konrad in linux-mainline on master
%
```

Improve code quality
/the sparse tool

```
konrad in linux-mainline on master
% make C=1 M=drivers/staging/dgnc

(now the sparse tool is being run for each file that is compiled)

konrad in linux-mainline on master
%
```

Do the missing parts
/the TODOs

```
konrad in linux-mainline on master
% find . -name "TODO" | wc -l
53
konrad in linux-mainline on master
% find . -name "README" | wc -l
55
```

The drivers/staging

Perfect place to start

Read the driverdev.linuxdriverproject.org Mailing List

Fix Kernel OOPS

Track and fix OOPS on your machine

Fix Kernel OOPS

Track and fix OOPS on your machine

Find inspiration at bugzilla.kernel.org

Many Ways of Contributing

Work on the drivers/staging code

Get inspiration from the TODOs and READMEs

Know what is happening around you (read Mailing Lists)

Improve code quality thus learn the code

It helps when you know it

The stuff that helps if you know it

It helps when you know it
/Git

Working with branches

Dividing the work into small chunks

The patches are best when served via git send-email

The git rebase -i is life saver

It helps when you know it

/Mailing Lists

The archives contain good stuff

Old patches from similar code might be a good hint

It helps when you know it

/Noone owns a task

Someone else might be working on the same stuff

It helps when you know it

/The coding style

Yes, the people are serious about the coding standard
and how the patches shall be formatted.

Summary

It is fairly easy to do the Linux Kernel development

You can start with the drivers/staging area

Fix the coding style issues

When ready move on to the next challenges

Read The F Mailing Lists

kthxbye

The End