WHY SHOULD WE CARE ABOUT KERNELNEWBIES!

Vaishali Thakkar (@kernel_girl)
Who Am I?

Rails Girls
Summer of Code

OUTREACHY
Who is Kernelnewbie?

Type one:
- Open source newbie
- May or may not have formal computer science education
- Curious, enthusiastic, confused
Who is Kernelnewbie?

Type two:
- Belongs to open source world, new to Linux kernel
- Wants to contribute to Linux Kernel out of interest or job
- Curious, Excited, confused
Flow of Morty’s Journey in a Linux kernel world

1. Setting up an environment for the first contribution

2. First successful contribution

3. Find tasks/projects for more valued contribution

4. Quality contribution in the Linux kernel / full-time job
Why should you care about Morty’s journey?

- Fast kernel development process -> 7.8 changes per hour
  (Greg Kroh Hartman at Git Merge, 2016)
Why should you care about Morty’s journey?

- Fast kernel development process -> 7.8 changes per hour (Greg Kroh Hartman at Git Merge, 2016)
- Use of Linux Kernel is increasing in devices [e.g. Android and IoT]
Why should you care about Morty’s journey?

- Fast kernel development process -> 7.8 changes per hour (Greg Kroh Hartman at Git Merge, 2016)
- Use of Linux Kernel is increasing in devices [e.g. Android and IoT]
- Need of more subsystem maintainers/reviewers to share the load
Why should you care about Morty’s journey?

- Fast kernel development process -> 7.8 changes per hour (Greg Kroh Hartman at Git Merge, 2016)
- Use of Linux Kernel is increasing in devices [e.g. Android and IoT]
- Need of more subsystem maintainers/reviewers to share the load
- Need of more kernel developers
Why should you care about Morty’s journey?

- Fast kernel development process -> 7.8 changes per hour (Greg Kroh Hartman at Git Merge, 2016)
- Use of Linux Kernel is increasing in devices [e.g. Android and IoT]
- Need of more subsystem maintainers/reviewers to share the load
- Need of more kernel developers
- Kernel community works on trust
Obstacles in Morty’s journey

1: Setting up an environment

- First patch tutorial helps with setting up an environment but doesn’t really talk about how kernel development process works

- It suggests checkpatch.pl fixes but not all subsystem maintainers are fine with accepting these fixes

- Different subsystems follow different workflow
Obstacles in Morty’s journey

2: First successful contribution

- No specific time defined on when you will get reply or patch will get merged

- Emails == Chances of patches being lost

- Subsystem maintainers picks up the patches based on their work flow - no documentation on the same
Obstacles in Morty’s journey

2: First successful contribution

- Patch tags - some prefers RFC and some don’t

- Picky but not patient enough

- Rude behavior -> discourages Morty to continue their journey
Obstacles in Morty’s journey

3: Find tasks for quality contribution

- Not using version control development platform == no open issues
- TODO files in subsystems are not often updated
- Hard to find TODO tasks listed as comments in files
Obstacles in Morty’s journey

3: Find tasks for quality contribution

- Maintainers/developers often have few ideas but do not have time to implement them because of the busy schedule

- Newbies often look for more ideas but don’t know where to find them
Obstacles in Morty’s journey

4: Quality contribution in the kernel

- Many times new tasks often require discussions with the people before sending patches

- Mailing lists/IRCs work for some subsystems but having an information on whom to reach out can help when starting
Kernel Planet
Possible Solution
Possible Solution

Project Kernel Bridge
Idea: A home to address the concerns

- Website/Wiki
- Involve kernel developers/maintainers who are interested in building a bridge between kernel newbies and kernel developers / maintainers
Things to be implemented

- List of subsystems who are fine with receiving `checkpatch.pl` fixes and guiding kernelnewbies in their journey of first perfect kernel patch
- List of documentation available for each subsystem/kernel areas in a more organized manner
Things to be implemented

- Subsystem wise TODO list of tasks (Similar to github's help needed/beginner friendly tags)
Things to be implemented

- Subsystem wise assigned list of reviewers or developers who wants to help in reviewing patches
Things to be implemented

- List of more ambitious projects [Just a bunch of possible ideas from developers/maintainers]
[RFC] Solutions
I’m not sure about

- **Problem**: Behaviour

- **Solution**: Code of conduct, basically something that says you’ll not shout if the patch takes multiple revisions and bit of a time
[RFC] Solutions
I’m not sure about

- **Problem:** No docs on subsystem wise development process

- **Solution:** Document the subsystem wise process of sending patch to patch being merged [approx time?]
More ideas?
More ideas?
[https://github.com/nerdyvaishali/kernelbridge]
Resources

- Google.com
- Adult Swim - Rick and Morty