SimplyClip

Anirudh Pande[1], **Lalit Bangad**[2], **and Pratyush Vaidya**[3]

[1]**apande@ncsu.edu**
[2]**lbangad.ncsu.edu**
[3]**pvaidya.ncsu.edu**

ABSTRACT

The following document presents linux kernel best practices and their connection to rubric items.

Keywords: linux kernel best practices, college group project

INTRODUCTION

Noted below are best practices and the lessons learnt while developing kernel of linux through the years according to Linux Kernel report publication 2017:

- `Short release cycles are important.`: New code is quickly made available in a stable release. Integrating new code on a nearly constant basis makes it possible to bring in even fundamental changes with minimal disruption.
- `Process scalability` : Process scalability requires a distributed, hierarchical development model.Spreading out the responsibility for code review and integration across 100 or more maintainers gives the project the resources to cope with tens of thousands of changes without sacrificing review or quality.
- `Tools matter.`
- `consensus-oriented model.`: No particular user community is able to make changes at the expense of other groups.
- `\no regressions" rule.`: This gives users assurance that upgrades will not break their systems.
- `Corporate participation in the process is crucial`: While corporate participation is important no company can drive development in directions that hurt the others or restrict what the kernel can do.
- `no internal boundaries.`: Any developer can make a change to any part of the kernel if the change can be justified.

no internal boundaries

- Number of commits
- Number of commits: by different people
- Issues reports: there are many
- issues are being closed
- Chat channel: exists
- evidence that the members of the team are working across multiple places in the code base

“no regressions” rule

- Issues reports: there are many
- issues are being closed
- Chat channel: exists

consensus-oriented model

- Issues reports: there are many
- issues are being closed
- Chat channel: exists
- Use of version control tools
- Use of style checkers
- Use of code formatters.
- Use of syntax checkers.
- Use of code coverage
- other automated analysis tools

Tools Matter

- Use of version control tools
- Use of style checkers
- Use of code formatters.
- Use of syntax checkers.
- Use of code coverage
- other automated analysis tools

Corporate participation in the process is crucial

- the files CONTRIBUTING.md lists coding standards and lots of tips on how to extend the system without screwing things up

Process scalability

- workload is spread over the whole team (one team member is often Xtimes more productive than the others...
- Number of commits
- Number of commits: by different people

REFERENCES

[1]Linux Kernel Development Report 2017 by Jonathan Corbet and Greg Kroah-Hartman