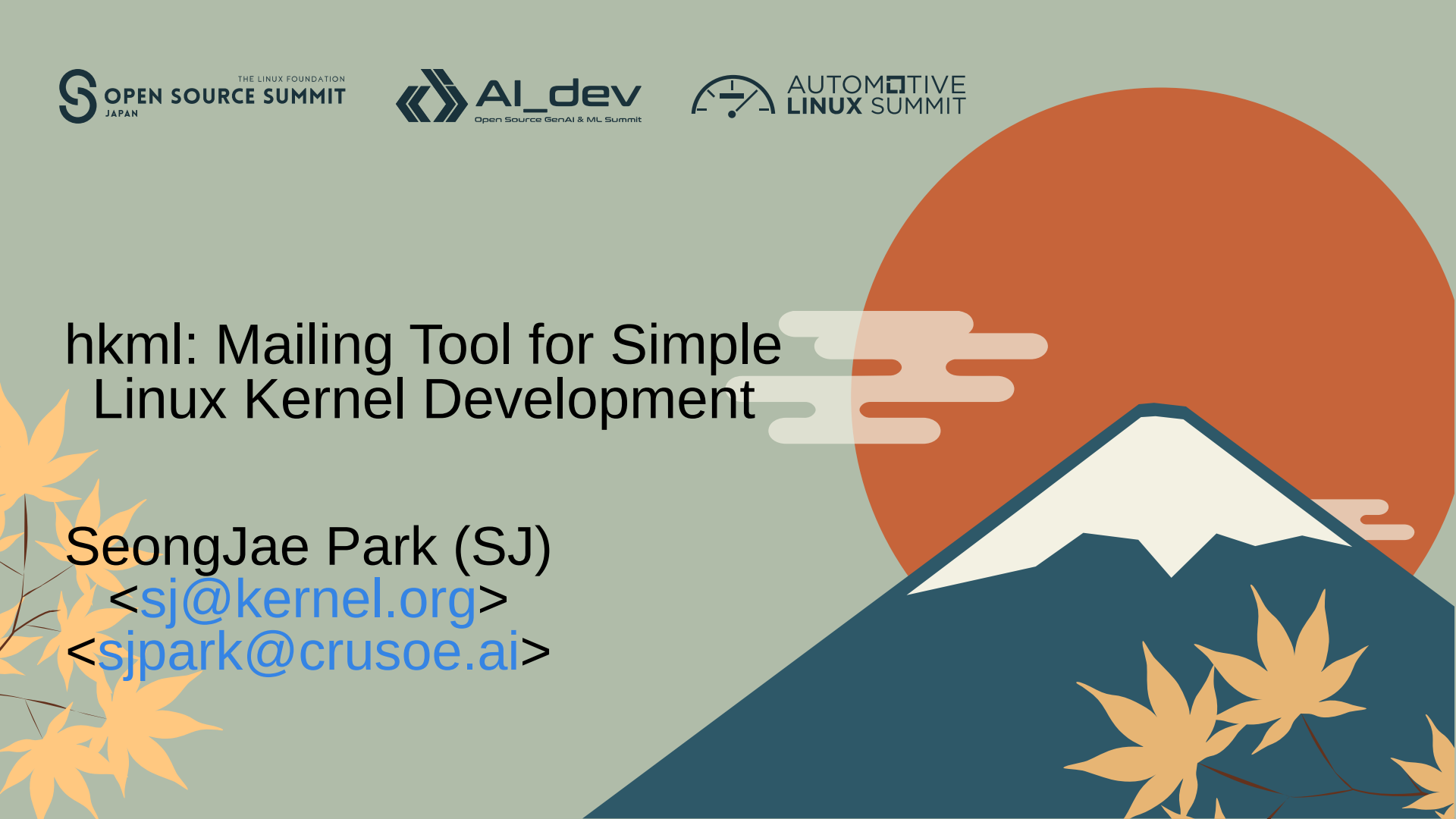


hkml: Mailing Tool for Simple Linux Kernel Development

SeongJae Park (SJ)
<sj@kernel.org>
<sjpark@crusoe.ai>

Table of Contents

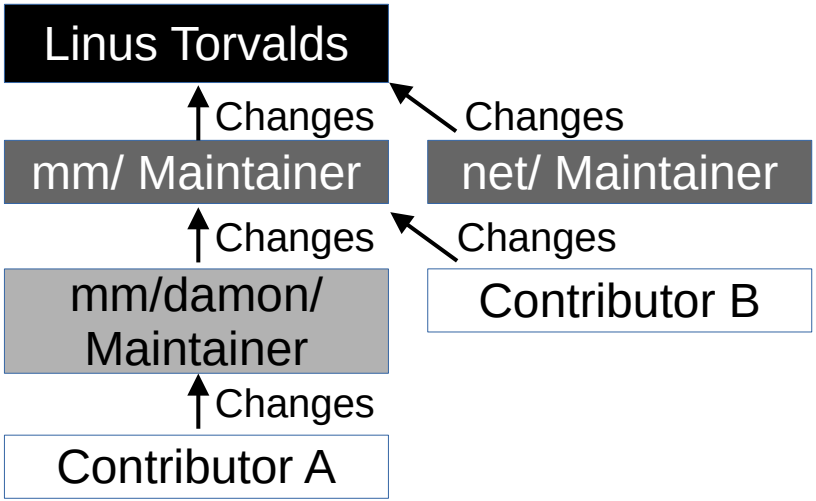
Linux Kernel Development Process and Its Pain Points

Linux Kernel Development Statistics

- Survived from the test of time, since 1991
- Being used nearly everywhere
- Keeping and even accelerating development speed
  - ~2,000 developers, ~15,000 commits, per ~9 weeks
- Most successful and active open source software

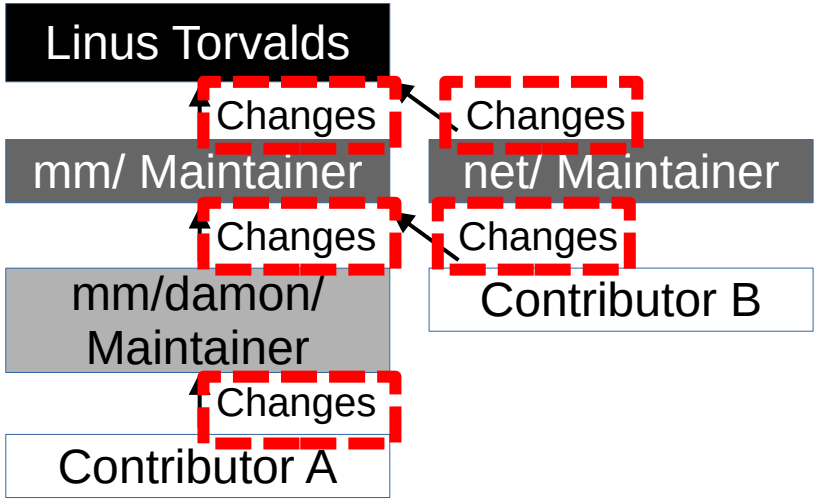
Kernel Development Process, 1000 ft View

- Kernel: A set of (recursive) subsystems
- One maintainer team per subsystem
- Contributors send changes to maintainers
- Maintainers send changes to upper level maintainer
- Distributed, scalable, simple



Individual Code Change Process, 100 ft View

- Make local changes
- Convert changes into human-readable form
- Send it to the relevant people
- Discuss about the proposal with the people
- Repeat until accept/reject
- Straightforward



Making Change Request, 10 ft View

- Standardized tool (‘git’) and resources available
- Make local changes
  - ‘git commit’
- Convert changes into human-readable form
  - ‘git format-patch’ or ‘git request-pull’

- Make local changes
- Convert changes into human-readable form
- Send it to the relevant people
- Discuss about the proposal with the people
- Repeat until accept/reject
- Straightforward

Sending Change Request, 10 ft View

- Send it as a plain text email
- Standardized tool (‘git’) makes this semi-automated and doable
- Finding relevant people
  - Maintainers, reviewers, and mailing list of the subsystem
  - MAINTAINERS file and get_maintainer.pl can be used
- Sending the patch or pull request
  - ‘git send-email’

- Make local changes
- Convert changes into human-readable form
- Send it to the relevant people
- Discuss about the proposal with the people
- Repeat until accept/reject
- Straightforward

Change Request Discussion, 10 ft View

- How to reply?
  - Bring your own mailing tool (Note: Gmail is sub-optimum)
- How to read others' mails?
  - Subscribe to mailing list, or
  - Search mailing list archives
- How to convince others?
  - Out of the scope of this talk

- Make local changes
- Convert changes into human-readable form
- Send it to the relevant people
- Discuss about the proposal with the people
- Repeat until accept/reject
- Straightforward

Change Request Discussion Pain Points

- Finding proper email client
  - There are more than two ways to do that
    - Kernel official documentation introduces 15 tools
- Reading other’s email
  - Subscribing doesn’t work for busy subsystems (hundreds of mails per day)
  - Modern mailing list archives (lore.kernel.org) are nice to lookup
    - Still, lacks ease of the private inbox and replying
- Many beginners forgive from this stage, or live with the pain

hkml: Hackers' Mails Management Tool

Evolution Story of a Mailing Tool

- Beginning: public-inbox based hack for mailing lists monitoring
  - Only for scratching the developer’s itch
- Middle age: Extended for DAMON maintenance mailing works
  - Still a hack for personal usages
- Now: Committed to support general Linux kernel contributors
  - For the developers friends and DAMON contributors
  - Listed on Linux kernel official document

hkml: Mailing tool for Mails-driven development

- Developed for minimum setup and resources
- Support public-inbox archives and mbox files
  - Not only for Linux kernel, not only for public-inbox
- Highly optimized for Linux kernel development
- Available at https://github.com/sjp38/hackermail

hkml Demo Time

Setup

- git clone https://github.com/sjp38/hackermail
- That’s it!

Reading Mails from Mailing Lists

- `hkml list <list name, e.g., damon>`
- Show the mails of the list with an interactive UI
- The interactive UI supports most works
  - Replying, Forwarding, Finding patches to review, Exporting mails as an mbox file, etc

Reading Mails Not Sent to Mailing Lists

- Save the mail[s] as an mbox file
- hkml list <mbox file>

Reading Mails of a Thread on Mailing List

- hkml list <msgid or lore.kernel.org link>
  - e.g.,

```
‘html list https://lore.kernel.org/damon/20251125015841.76180-1-sj@kernel.org’
‘hkml list 20251125015841.76180-1-sj@kernel.org’
```

Tagging Mails

- From the list, press ‘`m`’ and select ‘`manage tags`’
- To read mails of a tag,
  - `hkml list <tag name>`

Replying to Mails

- From the list, press ‘`m`’ and select ‘`reply`’
- The mail will be tagged as ‘`drafts`’ or ‘`sent`’ depend on your following action
- ‘`hkml`’ find previous draft and let you continue writing it

Writing and Sending a Mail

- `‘hkml write’` from the terminal
  - Works similar to replying feature

Testing Patches

- Press ‘`m`’, select ‘`handle as patches`’ → ‘`check patch[es]`’
  - Runs `checkpatch.pl` by default

Applying Patches

- ‘handle as patches’ → ‘apply patch[es]’

Downloading Patches

- ‘handle as patches’ → ‘export patch[es]’

Formatting and Sending Patches

- From terminal, ‘`hkml patch format <commits>`’
  - Setup CV with first commit’s parent’s message
  - Add recipients based on `get_maintainer.pl`
    - Only coverletter gets all recipients
  - Run `checkpatch.pl` for each patch
  - Give you a moment to review subjects and reviewers
  - Finally send the patches
  - The process can be aborted at any step

Remote tags synchronization

- `hkml sync --remote <your private git repo>`

Mails Monitoring

- hkml monitor add
- hkml monitor start

And more hidden features

- Flexible mails filtering
- Flexible mails display options
- Refer to USAGE.md
- Suggest/contribute your features

QnA

- Feel free to use
  - sj@kernel.org
  - https://github.com/sjp38/hackermail/issues

Backup Slides

Why not lei+b4+mutt ?

- Any tool is fine, if it works for you
- hkml developer didn’t take enough time on understanding the tools combination, so no strong opinion
- hkml developer just found hkml works for their workflow, so decided to use it