内核文档贡献指南

如何向 Linux 内核文档提交你的补丁

华中科技大学开放原子开源俱乐部

2024 年 4 月 28 日

⚠ 警告!

由于 CTEX 引擎可能产生的各种字符替换问题，请不要复制该文件中的任何命令到你的终端中。

具体命令请查看HCTT Wiki

提前准备

环境配置

必备工具：

- 文本编辑器（VsCode, Sublime Text, [Neo]Vim, ...）
- Git [1]
- 一个搜索/翻译引擎
- 电子邮件客户端 (git-sendmail [2])

[1] 一个交互式的 Git 学习沙盒：
https://oschina.gitee.io/learn-git-branching/
[2] 使用 git-sendmail 发送补丁邮件参见
setting-up-git-send-email-with-gmail-to-send-linux-kernel-patch

提交到 HCTT

获取翻译项目

首先在 github 上 fork 我们的 TranslateProject

然后将自己的仓库 clone 到本地

```
git clone https://github.com/your-github-id/TranslateProject.git
```

选题

- 在HCTT 内核选题中选择 HCTT 维护者收集的选题

翻译注意事项

- 对于 HCTT 中译文，我们只需要在原文件上进行改动，同时在 Markdown 元数据中填入一些信息
- 遵循HCTT 翻译流程和中文排版指北

一般来说 HCTT 文档的翻译只需要遵循这两项规则即可。翻译完成后，译者需要自己先 review 一遍译文，然后创建 pr，可以使用我们提供的脚本一键提交

```
cd </path/to/your/translate_project>
submit2hctt.sh <path/to/your/markdown>
```

当然也可以使用浏览器创建 pr 进行提交

Anything you like

提交到内核

获取内核文档源代码

从 lwn 克隆整个 Git 仓库，并切换到文档的开发分支 docs-next

```
git clone https://mirrors.hust.edu.cn/git/lwn.git
or
git clone git://git.lwn.net/linux.git
cd lwn
git checkout docs-next
```

获取源代码后，可以在源代码目录下的
Documentation/doc-guide/sphinx.rst 查看内核文档子系统构建流程。

选题

- 在 Documentation/translations/zh_CN 中选择自己感兴趣的部分，并且查看其对应的 index.rst 中的 TODO List，从其中选择你想翻译的文章进行翻译

开始翻译

在开始翻译内核代码前，强烈建议阅读内核文档翻译指南

★ 其中最需要注意的是

- **一致性**：翻译后的原文，除了语言和开头的元数据不同，其他格式都应该与原文相同。
- **最大宽度**：中文文档要求每行长度不超过 40 个中文字符，或者不超过 80 个英文字符，其中一个中文字符的长度 $=$ 两个英文字符的长度，无论具体是多长，请尽量保持每行长度一样，要整整齐齐。

编译文档子系统

在提交之前，可以通过编译文档子系统判断翻译是否正确。

1. 安装 sphinx

```
./scripts/sphinx-pre-install
```

2. 开始编译

```
make cleandocs
make htmldocs
```

3. 查看编译信息，检查是否有和自己新添加的文章有关的警告或报错

Commit!

在完成修改后，可以采用 diff 工具检查你的修改。

```
git diff
```

检查没有问题后，开始 commit！[3]

```
git commit -asev
```

规范的 Commit message

docs/zh_CN: summary phrase

详细描述，如需分段，段与段之间需要一个空行

Signed-off-by: Author

[3] 当第二次 commit 时，请使用 `git commit -av --amend`

生成并提交补丁

华中科技大学邮箱配置指南

在使用 git-send-mail 发送邮件前，需要配置好你的 SMTP 设置。

1. 设置邮箱密码：通过统一身份认证进入学校邮箱，并设置一个邮箱账户密码。
2. 修改你的.gitconfig [4]

```
[sendemail]
    smtpEncryption = ssl
    smtpServer = mail.hust.edu.cn
    smtpUser = 替换为你的学校邮箱地址 (xxxx@hust.edu.cn)
    smtpServerPort = 465
```

[4] 当然，在.gitconfig 中不进行任何配置并在发送时指定命令行参数也是可行的。

建立你的 patch 文件

通过 git format-patch 建立你的 patch 文件，其默认 subject 前缀为 [PATCH]。

```
git format-patch -1
```

如果是第二次提交 Patch，需要自定义你的 subject 前缀和 changelog。

```
git format-patch --subject-prefix="PATCH v2" -1
```

Changelog 的格式可参考12页中的样例。

正确的 patch 格式 [5]

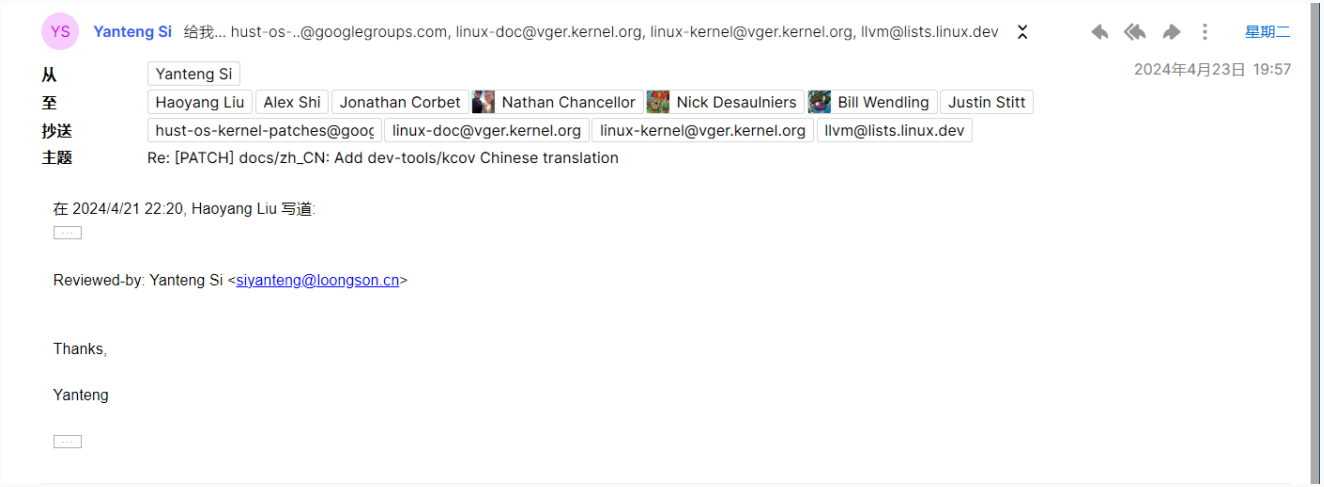
```
<commit message>
...
Signed-off-by: Author <author@mail>
---
V2 -> V3: Removed redundant helper function
V1 -> V2: Cleaned up coding style and addressed review comments

path/to/file | 5++++--
...
```

[5] submitting-patches.rst:704

patch 常见的签名标签及用途

- Signed-off-by: 表示签名者参与了补丁的开发，或者在补丁的交付路径中。
- Reviewd-by：表示一种意见声明，即补丁是对内核的适当修改，没有任何严重技术问题。此标签用于向审阅者表示信任，并告知维护人员对补丁所做的审阅程度。当维护者回复如下图所示的邮件后，说明你的翻译没有什么问题，代表已经被接收了，你也不需要继续发送邮件了。如果需要发送 V2 等版本的 PATCH，可以在 PATCH 上加上这一行。



内部 Review 邮件列表

邮件列表链接：

https://groups.google.com/g/hust-os-kernel-patches

在发送邮件时，在 Cc 中添加下面邮件地址

hust-os-kernel-patches@googlegroups.com

检查并发送

在发送 Patch 之前，检查你的 patch 文件

```
make cleandocs
make htmldocs
./scripts/checkpatch.pl ${PATCH_FILE}
```

根据编译信息和脚本提示消除所有错误和警告

可以通过下面的脚本得到相关 maintainer 信息：

```
./scripts/get_maintainer.pl ${PATCH_FILE}
```

发送你的 Patch

在给 Linux 邮件列表（LKML）发邮件之前，请先发送到内部邮件列表。

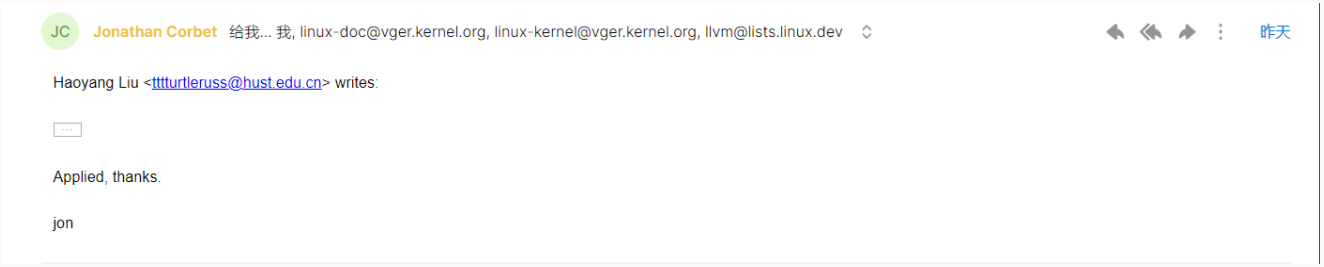
```
git send-email --to="hust-os-kernel-patches@googlegroups.com" ${
    PATCH_FILE}
```

内部审核通过之后，通过*get_maintainer.pl* 获取 maintainer，并发送你的 patch 给 maintainer，当然，也要记得 cc 我们的内部邮件列表以方便内部跟踪 Patch 进度。

```
git send-email --to-cmd="$(pwd)/scripts/get_maintainer.pl --nogit --nogit
    -fallback --norolestats --nol" --cc-cmd="$(pwd)/scripts/
    get_maintainer.pl --nogit --nogit-fallback --norolestats --nom" --cc
    ="hust-os-kernel-patches@googlegroups.com" ${PATCH_FILE}
```

如何得知自己的 PATCH 被接受

当收到来自 Jonathan Corbet <corbet@lwn.net> 的回复后，说明你的 PATCH 已经进入文档子系统了。



完结撒花!!

疑问？


HCTT
翻
HCTT 翻译组
华中科技大学Security Pride慕冬亮...
扫描群二维码，立刻加入该群
该二维码 7 天内 (5/3前)有效