Enviando patches com o Git

Lucas Seiki Oshiro

Contribuições usando Git

- Git é usado por 93% dos desenvolvedores (fonte)
- Usado não só para o **versionamento** em si, mas também para desenvolver de forma **colaborativa**
- Isso inclui **comunidades** FLOSS, **empresas**, **academia**
  - E qualquer outra situação que várias pessoas precisem lidar com o mesmo código!
- Serviços de hospedagem de repositórios ajudam
  - Ex: GitHub, GitLab, BitBucket
  - **Não é o caso do kernel**

Fluxo com GitHub / GitLab / etc

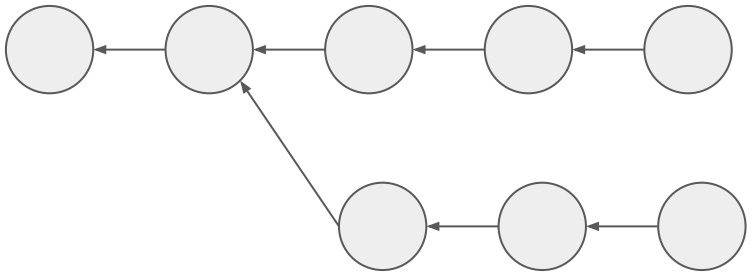
- Envio de Pull Requests (GitHub) ou Merge Requests (GitLab)
- Neles são feitas as contribuições
  - Código é enviado em uma **branch**
  - Além das mensagens dos commits, tem uma **descrição**
  - Os códigos são **revisados**
  - Caso esteja tudo ok, o mantenedor faz o **merge** da branch do PR/MR
- Exemplo:
  - Pull Request no GitHub
  - Revisão de código

E no Kernel?

- Tudo por **e-mail** (!!!)
  - Discussão
  - Código
  - Revisão
  - E funciona bem!
- Os nomes mudam um pouco:
  - O código é enviado nos chamados **patches**, cada um em uma mensagem de e-mail
  - Pequenas alterações vão em um único **patch**
  - Uma sequência de alterações vão em uma sequência de patches (e-mails) chamada **patchset**
  - A descrição é feita em um email separado chamado **cover letter**
- **Exemplos:**
  - Patch
  - Patchset

git format-patch

- Comando para formatar **commits** como **patches**
- Cada patch contém:
  - Um **assunto**, exemplo:
    `[PATCH 1/4 v2] titulo do commit`
  - Uma **mensagem**, correspondente à mensagem do commit
  - O diff do commit em relação ao commit anterior
- Os patches gerados são **arquivos** que podem ser abertos por clientes de email (Thunderbird, neomutt, Apple Mail, etc) para serem enviados
  - Eles são a própria mensagem, e não um anexo!



git format-patch

- Comando para formatar **commits** como **patches**
- Cada patch contém:
  - Um **assunto**, geralmente no formato `[PATCH 1/4 v2] titulo do commit`
  - Uma **mensagem**, correspondente à mensagem do commit
  - O diff do commit em relação ao commit anterior
- É possível enviar uma **descrição** adicional que não é parte do commit logo após o **---** (chamado de scissors)

```
[lso@eve (delta_width) patch-hub]$ git format-patch -2 --cover-letter -v2
v2-0000-cover-letter.patch
v2-0001-feat-increase-Delta-output-width.patch
v2-0002-feat-disable-gitconfig-reading-by-Delta.patch
[lso@eve (delta_width) patch-hub]$
[lso@eve (delta_width) patch-hub]$ cat v2-0001-feat-increase-Delta-output-width.patch
From c89fb650721991199ec5c184215814a83052c159 Mon Sep 17 00:00:00 2001
From: Lucas Oshiro <lucasseikioshiro@gmail.com>
Date: Tue, 25 Feb 2025 12:37:11 -0300
Subject: [PATCH v2 1/2] feat: increase Delta output width

Increase the Delta output to 130 columns. Ideally, this should be the
width of the "Preview" widget.

Signed-off-by: Lucas Oshiro <lucasseikioshiro@gmail.com>
---
 src/app/patch_renderer.rs | 2 ++
 1 file changed, 2 insertions(+)

diff --git a/src/app/patch_renderer.rs b/src/app/patch_renderer.rs
index e7a3bdd..4c93db9 100644
--- a/src/app/patch_renderer.rs
+++ b/src/app/patch_renderer.rs
@@ -100,6 +100,8 @@ fn delta_patch_renderer(patch: &str) -> color_eyre::Result<String> {
         .arg("less")
         .arg("--paging")
         .arg("never")
+        .arg("-w")
+        .arg("130")
         .stdin(Stdio::piped())
         .stdout(Stdio::piped())
         .spawn()
--
2.39.5 (Apple Git-154)
```

git send-email

- Abrir os patches no cliente de email é **chato**
  - Bem que o Git poderia enviar o email pra mim…
- E pode, com **git send-email** !

git send-email: configuração

- É necessário instalar o pacote antes:
  - Veja no link nas referências como!
- Configuração:
  - Nome, email
  - URL do servidor SMTP
  - Porta do servidor SMTP
  - Configuração de senha (opcional, ela **é armazenada sem criptografia!**)

```
## Configure with:
git config --global user.name "<your name>"
git config --global user.email "<your email>"
git config --global sendemail.smtpencryption tls
git config --global sendemail.smtpserver "<your_email_smtp_server_address>"
git config --global sendemail.smtpuser "<your_email>"
git config --global sendemail.smtpserverport "<your_email_smtp_server_port>"
```

```
git config --global sendemail.smtppass "<your password>"
```

git send-email: GMail

- Todo mundo na USP tem um **GMail**, o e-mail USP, por isso o foco nele
- O envio de e-mails via SMTP **continua ativo**, porém, precisa usar uma **App Password** em vez da senha normal
- É necessário habilitar antes a **autenticação em duas etapas**
- Uma App Password só pode ser vista **uma vez**, então cuide bem dela!
- Cuidado, são 16 caracteres e **não tem espaço** (isso engana…)
- Configurações:
  - smtpserver = smtp.gmail.com
  - smtpserverport = 587

git send-email: envio

- Exemplo de chamada:

```
    git send-email -3 --to='linux-iio@vger.kernel.org'
--cc='my@friend.com' --annotate --cover-letter
```

- As opções do **format-patch** funcionam nele
  - --cover-letter, -v3, -3, etc
- Outras flags:
  - --to: destinatário, no caso, a **lista de email** do subsistema
  - --cc: o cc do e-mail
  - --annotate: abre um **editor de texto** para editar a **cover letter** e os **patches**
    - Caso não tenha configurado o editor, configure com `git config --global core.editor <seu_editor>`
  - --dry-run: faz todo o processo mas **sem enviar o email**, para pessoas medrosas :-)

Referências

- Sending patches by email using git (FLUSP): https://flusp.ime.usp.br/git/sending-patches-by-email-with-git/
- App Passwords no GMail: https://support.google.com/mail/answer/185833
- Manpage do git format-patch: https://git-scm.com/docs/git-format-patch
- Manpage do git send-email: https://git-scm.com/docs/git-send-email