Git によるバージョン管理入門

田中 健策（株式会社 RAKUDO）

第四回

前回までのあらすじ

conflict が起こらないように、ブランチを分けて開発をすることにしました。しかし、それらを統合しないと完成しません。
そのためには

- 各ブランチで独立して開発ができて、
- 統合が楽で（人手を介さず、自動的にできる）
- 結果がわかりやすい

ような手法が必要です。
それを

- LaTeX のソース分割
- github のプルリクエスト
- github actions

によって構築してみましょう。

LaTeX のソース分割 1

複数人で LaTeX を書くときは、ファイル分割をする必要です。
一番簡単なファイル分割は、\input や\include を使うものです。
しかしこれには、単独のファイルでコンパイルできないという欠点があります。
LaTeX のコンパイルはそれなりの時間がかかるので、これは避けたいです（コンパイルして結果を見るためだけに数秒かかるのは、執筆時に多大なストレスになります）。

LaTeX のソース分割 2

そこで、自分でマクロを各方法や、パッケージを使う方法など、様々な方法が開発されています。

- 効率的な LaTeX ファイル分割術
  https://qiita.com/wtsnjp/items/6ba3b8e12514d8a3bd41
- 分割した LaTeX ファイルを subfiles を使ってコンパイルする
  https://qiita.com/sankichi92/items/1e113fcf6cc045eb64f7

今回は、二つ目の subfiles パッケージを使います。TeX Live に含まれているが、もし各自の環境に含まれていなければインストールしておいてください。

main.tex と introduction フォルダの中の introduction.tex のファイルが main ブランチにあります。この 2 つが書き方の例になっていますので、見ておいてください（次のスライドと Wiki にも書き方を書いておきます）。

subfilesパッケージの使い方

まず、統合する方の LaTeX ファイル（今回は main.tex）のプリアンプル

```
\usepackage{subfiles}
```

を書き、本文の挿入したい位置に

```
\subfile{自分のフォルダ/自分のファイル}
```

を加えます。
そして自分の LaTeX ファイルのプリアンプルは

```
\documentclass[../main]{subfiles}
```

だけにしておきます。
必要なプリアンプルは main.tex のプリアンプルに書いておいてください（独自設定を使いすぎると conflict が起こるので気をつけてください）。

gitignoreについての注意

gitリポジトリーには普通、生成されるファイルは登録しません。
登録しないファイルのパターンは「.gitignore」というファイルに書いておきます。今回から、pdfファイルを.gitignoreに加えました。すでに生成されたpdfファイルをリポジトリーに登録してしまった人は、「git rm」を使って消しておいてください。
図などのために pdfファイルをリポジトリーに登録する必要がある人は、その pdfファイル（例えば tanaka/diagram.pdf）
を.gitignore に
!tanaka/diagramp.pdf
などのように無視しないように設定してください。
詳しくはウェブで.gitignoreについて調べればわかります。

プルリクエスト

良い仕組みを作っても、もしメンバーの一部ががおかしなファイルをリポジトリーに加えてしまえば、他の人までコンパイルできなくなって迷惑を被ります。
そうならないようにする仕組みが**プルリクエスト**です。
これは git ではなく Github の機能です。
「レビューをして、問題ないことを確認したらマージ」という作業をタスク化して、やりとりを記録できるようにしてくれるのです。

プルリクエストの流れ

- Github のリポジトリの上部のボタンでプルリクエストを作成
- 担当者がレビューして、修正が不要ならマージされます
- 修正が必要なら修正を依頼し、修正を push 後またレビュー
- そもそもプルリクエスト自体が不要ならクローズされます

このような流れが関係者にオープンになり、また記録されます。またオープンなリポジトリなら、自由にプルリクエストが作成できるので、自由に開発に参加することができます（もちろん担当者がプルリクエストのレビューを捌けるならではありますが）

今回のプルリクエストの使い方

- 定期的に main ブランチを自分のブランチに merge しておいて、main の変更を取り入れてください。
- 自分のブランチの LaTeX 作成が終わったら、コンパイルできることを確認します。
- main.tex を編集して、自分の作った LaTeX ファイルが main.tex をコンパイルしたときに取り入れられるようにします。
- main ブランチへのプルリクエストを作成します。
- 講師の田中が確認して、main へ merge します。

github actions

github などの github ホスティングサイトでは、その性質上さまざまなイベントが起きます。
そのイベントを起点に、さまざまな処理を自動的に走らせるのが、github actions です（それ以外に webhooks も同様の使い方ができます）。
今回は github actions を使って、「main ブランチへの変更があったら、自動的にコンパイルして pdf をダウンロード可能にする」という仕組みを作ります。
その仕組み自体は講師の田中が作るが、興味がある方は.github フォルダ内に関連ファイルがあるので、設定方法を確認してみてください（docker などの知識が必要です）。

結果の見方

main ブランチに変更があると、docker という仕組みを使って LaTeX をコンパイルする仮想環境が作られ（結構時間がかかる）、main.tex から pdf が生成されます。
生成された pdf は `https://github.com/tannakaken/nugitlecture2020/releases/new` からダウンロードできます。
これによって、現在のプロジェクトの進捗が可視化されます。
またコンパイルに失敗すると、管理者（この場合講師の田中）にメールが届きます。これで失敗の可視化もされます。
プログラミングや文章の執筆など、形がなくて進捗が見えにくいプロジェクトにおいて、これは重要です。

今回の課題

- 自分の LaTeX ファイル（なんらかの問題の解答。何の問題に解答したかを記入）を完成させる。
- 自分のファイルを subfiles に対応させ、main.tex を編集して、自分のファイルだけでも、main.tex に統合してでも、どちらでもコンパイルできるようにする。
- main ブランチへのプルリクエストを作成する。