計算機科学第一

10 月 6 日

今日の目標

- バージョン管理システム Git を使えるようになる。
- Scala の 2 通りの実行方法を経験する。
- Scala のビルドツール sbt をある程度使えるようになる。

1 バージョン管理システム Git について

1.1 バージョン管理システムとは

バージョン管理システムはある程度の規模のプログラム (もっと一般にテキストファイル) を書く上では必須のツールです。今までプログラムを書いてきて、既にある程度動作するプログラムに大きな変更を加えたくなったことがあると思います。そのような場合に一旦別ファイルにコピーしてからプログラムを編集した経験があるかもしれません。バージョン管理システムを使えばそのようなことをする必要はありません。またチームでプログラムの開発をする場合には一つのファイルを複数の開発者が書き換えた場合に可能な限り自動的にマージをしてくれます。この授業では最近のバージョン管理システムの中でも最もよく使用されている Git を使います。

1.2 Git の使い方

1.2.1 初期設定

ホームディレクトリに `.gitconfig` というファイルを作って以下のように書いてください。

```
[user]
        name = #名前#
        email = #メールアドレス#
[push]
        default = simple
```

ここで `#` **名前** `#` はあなたの名前の英語表記で置き換えてください。また `#` **メールアドレス** `#` はあなたのメールアドレスで置き換えてください。

1.2.2 リポジトリを作る

Git はリポジトリという単位でファイル群を管理します。まず Git で管理したいディレクトリに移動します。既に作成したファイルがあっても、空のディレクトリであっても構いません。そして

```
git init
```

とコマンドを実行します。これで現在のディレクトリ以下を Git で管理できるようになりました。この Git で管理されているディレクトリのことをリポジトリと呼びます。このディレクトリには `.git` というディレクトリが作成されています (`ls -A` で確認できます)。このディレクトリは Git が情報を保存するためのディレクトリであり、設定ファイルである `.git/config` 以外を編集することはありません。

1.2.3 管理するファイルを追加する

リポジトリを作っただけでは Git はどのファイルを管理すればよいのか分かりません。

```
git add #ファイル名#
```

というコマンドで Git で管理したいファイルを指定します。ここで `# ファイル名 #` は Git で管理したいファイル名に置き換えてください。たとえば `hello.scala` というファイルを Git で管理したいときは

```
git add hello.scala
```

としてください。これで Git は「`hello.scala` というファイルの変更を管理すればよいのだ」ということを理解します。しかしまだ Git は `hello.scala` の内容を読んではいません。次に紹介する `git commit` でファイルの内容を記録する必要があります。また一つのリポジトリ内で複数のファイルを追加しても構いません。

1.2.4 管理しているファイルの変更を記録する

今現在のファイルの状態を Git で記録したいときは

```
git commit #ファイル名# -m "#コメント#"
```

とします。ここで `# コメント #` は「このファイルの変更に対するコメント」で置き換えてください。自分一人しか使用しないリポジトリの場合はこのコメントはなんでもよいです。

リポジトリ内で最終コミット以降に更新された全てのファイルの変更を記録したいときは

```
git commit -a -m "#コメント#"
```

としてください。

コミット直後に Git が記録している最新のファイルの状態をリビジョンと呼び、それぞれのコミットに「リビジョン名」が割り当てられます (後述)。

1.2.5 その他のコマンド

過去のコミットログを見たい場合は

```
git log
```

としてください。
　最終コミットと現在のファイルとの差分を見るには

```
git diff
```

としてください。特定のリビジョンと現在のファイルとの差分を見たい場合は

```
git diff #リビジョン名#
```

としてください。ここで # リビジョン名 # は比較したいリビジョンのリビジョン名で置き換えてください。ここでリビジョン名とは git log で該当するコミットに関するログ

```
commit 8ac50947b3f75158f457335c65f3ff93e687c045
Author: Ryuhei Mori <mori@is.titech.ac.jp>
Date:   Sat Sep 19 10:33:44 2015 +0900

    comment
```

の中で 8ac50947b3f75158f457335c65f3ff93e687c045 にあたる部分です。これはとても長いですが最初の数文字を指定すれば十分です。例えばこの例の場合は

```
git diff 8ac509
```

としてください。
　特定のリビジョン同士の差分を見るには

```
git diff #リビジョン名1# #リビジョン名2#
```

としてください。
　コマンドのヘルプを見たいときは

```
git help #gitコマンド名#
```

としてください。ここで #git コマンド名 # は git のコマンド名 (init, add, commit, log, diff など) で置き換えてください。また、ここで紹介したコマンド以外にもたくさんの git コマンドがあります。

1.3 リポジトリの clone

上の使用方法では自分でリポジトリを作りましたが、既に存在するリポジトリを自分の環境にコピーすることができます。

```
git clone #gitリポジトリパス#
```

とすると現在のディレクトリに新しいディレクトリが作成されます。この新しいディレクトリは `#git リポジトリパス #` で指定したリポジトリのコピーです。リポジトリの指定方法については後述します。

Clone で作成されたリポジトリの場合、`.git/config` に clone 元のリモートリポジトリのパスが書かれているはずです。Clone したリポジトリに対して書き込み権限があれば変更をリモートリポジトリに反映することができます。

```
git push
```

としてください。

リモートリポジトリが更新された場合にその変更を自分のリポジトリに取り込むには

```
git pull
```

とします。ただし、自分のリポジトリとリモートリポジトリとで同じファイルの同じ部分を編集していた場合にはコンフリクトを自力で解消する必要があります。

2 今日の課題

1. Git のリポジトリを作って一通りコマンドを使ってみる (clone, push, pull 以外)。一例としては、リポジトリを作る → なにかテキストファイルを作る → 追加する → コミットする → ファイルを編集する → 差分を見る → 変更をコミットする → ログを見る → ファイルを編集する → 最初のリビジョンとの差分を見る。
2. 以降は `https://titech-is-cs115.github.io/lecture/assignments/lx00a.html` を参照。