ソフトウェア開発演習I
- 開発リポジトリのマイニングと分析 (2) -

情報科学研究科
ソフトウェア設計学研究室
助教　崔恩瀞(Eunjong choi)
E-mail: choi[at]is.naist.jp

講義概要

- 履修条件
  - ✓ C言語やJava言語の基礎的な文法知識を有すること。プログラミング未経験者はプログラミング基礎演習等を事前に履修することが望ましい。
  - ✓ また、Emacsエディタや UNIXのコマンドラインツール（grep,find等）についての基本的な知識を有することが望ましい。
- 成績評価 (Grading)
  - ✓ 各テーマごとに分析レポートの提出を課す。レポートの分析内容により成績の評価を行う。

講義スケジュール

Schedule of this lecture

	日時	内容
1.	１０月　５日（木）４，５限	開発リポジトリのマイニングと分析
2.	１０月１２日（木）４，５限	開発リポジトリのマイニングと分析
3.	１０月１９日（木）４，５限	開発リポジトリのマイニングと分析
4.	１０月２６日（木）４，５限	開発リポジトリのマイニングと分析
		毎週演習課題を提出してください.
5.	１１月　２日（木）４，５限	コードリーディング
6.	１１月　９日（木）４，５限	コードリーディング
7.	１１月１６日（木）４，５限	コードリーディング
8.	１１月３０日（木）４，５限	コードリーディング（予定）

欠席について

- 緊急の場合を除き、事前に連絡すること
  (softdev1-staff@is.naist.jp)
- 課題は出席者と同じ期限内に提出すること
  (必要があれば、授業のビデオが見られます)
- 2回を上限とすること

前回のGitコマンド

- git init
- git add <file/directry name>
- git commit –m “<comment>”
- git branch <branch name>
- git checkout <branch name>
- git log

前回のGitコマンド

```
> git init
```

- Gitリポジトリを作成するコマンド．当該ディレクトリ（ここではlocalフォルダ）でバージョン管理を始めることになる

```
> git add <file/directry name>
```

- ファイル（ディレクトリ）をインデックス（コミット候補）に登録

前回のGitコマンド

```
> git commit -m “<comment>”
```

- インデックスに追加されたファイルをコミット．コメントに何のためのコミットか，どのような変更を加えたか等を記述

```
> git branch <branch name>
```

- ブランチを作成する

前回のGitコマンド

```
> git checkout <branch name>
```

- ブランチを切り替える

```
> git log
```

- これまでどのようなコミットがされてきたかログを確認

前回の課題

- コンフリクトの解決
- git logのオプション

Gitの使い方に関する参考文献

- Webサイト(スライドシェア)
  - ✓ こわくないgit
    - ✓ http://ja.slideshare.net/kotas/git-15276118
  - ✓ いつやるの？Git入門
- 書籍もいろいろあるようです

- 前回は自分の作成したファイルを管理
- 今回は他のリポジトリをマイニング

作業フォルダの作成とリポジトリの準備

```
> mkdir local2
> cd local2
> git clone git://github.com/ruby/ruby.git
```

「git clone」とは？
Git 上のリポジトリからコピーを取得したい場合に使うコマンド.

git log（コミット履歴の閲覧）

```
>git log
Commit   12345de…                              #コミットID
Author:    Eunjong choi <choi@....>            #Author
Date  Thu Oct 4 15:01:42 2017 +0900            #日付

                    fix Bug                    #コミットメッセージ

Commit   abc12345…                             #コミットID
Author:    Hajimu Iida <iida@....>             #Author
Date  Thu Oct 1 ….
```

git log とは？
コミットを繰り返していると，または，新たにリポジトリ（コミット履歴付き）をクローンした場合，何が起こったのか振り返るためのコマンド

git log のオプションの例

```
>git log --oneline                          #1コミット1行で出力
        (--pretty=oneline)
> git log --date=short                      #日付整形
        (--relative-date)
        (--pretty=raw)
> git log --name-only                       #変更されたファイル名
        (--stat)
> git log --pretty=format:‘%cd %cn%n%s%n’ #コミットログ整形
        (--pretty=format: ……. --graph)
```

%cd: コミット日，%cn: コミッター名，%n: 改行，%s: メッセージ

> git help log
> https://www.kernel.org/pub/software/scm/git/docs/git-log.html

git shortlog

- ‘git log’ を要約して出力

git shortlog
コミットメッセージの一行目が著者ごとにグループ分けされて表示される

  - ✓ 主なオプション
    -n : 著者のコミットの数の降順で並び替え
    -s : コミットの数と著者だけ表示

- git help shortlog
- https://www.kernel.org/pub/software/scm/git/docs/git-shortlog.html

git tag

- タグの作成，削除，閲覧

> git tag
> 履歴上の重要なポイント（リリースバージョンなど）に印がつけられる.

- git help tag
  - ✓ https://www.kernel.org/pub/software/scm/git/docs/git-tag.html

git show

- タグ付けされたコミットに関連する内容を閲覧

```
> git show v2_0_0_0
```

  - ✓ git logと同様に –pretty=　のオプションが使える

- git help show
  - ✓ https://www.kernel.org/pub/software/scm/git/docs/git-show.html

シェルスクリプト

表示結果一覧に対して、
一つずつ繰り返し処理をする場合に便利

〇〇.shというファイルを生成

例えば

```
#!/bin/sh

for file in `ls`; do
        echo “$file”;
done | sort
```

コマンドlsで出力したファイル(ディレクトリ)名
1つずつを変数fileに入れ、
echoによってそのファイル(ディレクトリ)名を出力する
ことを繰り返して得られる出力結果に対して、
コマンドsortをした結果を最終的に出力する

./〇〇.sh　で実行

permission deniedが出た場合は、
chmodで権限を変更

シェルとパイプ

```
> cat test.txt                  #test.txtの中を表示
zzz
aaa
yyy
> cat test.txt | sort           #test.txtの中の表示結果を並び替え
aaa
yyy
zzz
> cat test.txt | sort –r > hoge.txt   #表示結果を逆に並び替え
> cat hoge.txt                        hoge.txtに書き出し
zzz
yyy
aaa
```

演習課題

Report Titleは「mining2_学生番号」
添付ファイルを利用する場合は，report titleと同じファイル名をつけること

締め切り：2017年10月19日（木）15：00

全学停電に伴う(10月14日(金)夕方から16日(月)午前まで講義サイト停止）

- ✓ 帰る際に大学PCの電源を落としてください。

演習課題（1/2）

- rubyリポジトリから以下の情報をマイニング
- 答えと答えを得た過程（コマンド）をレポート
  - ✓ 演習1-1: リポジトリ全体で何回コミットされたか
    - ✓ Hint: コミットを1行表示し，且つ，wcコマンドを使う
  - ✓ 演習1-2: リポジトリ全体で何人がコミットしているか
  - ✓ 演習1-3: 多くのコミットを行ったコミッター上位5人は誰か？
  - ✓ 演習1-4: v1_9_0_0のタグがついたコミットは何月何日に行われたか？

演習課題(2/2)

- rubyリポジトリから以下の情報をマイニング
- 答えと答えを得た過程(コマンド)をレポート
  - ✓ 演習2-1:頻繁にバグを修正したコミッター上位5人,
    ただし、コミットメッセージに“Bug” (大文字, 小文字を区別しない) を含むものをバグ修正と考える
  - ✓ 演習2-2:win32/win32.cのコミット回数
  - ✓ 演習2-3:リポジトリのファイル数
  - ✓ 演習2-4:win32ディレクトリの中のファイルのうち
    頻繁にコミットされたファイル上位5つ

GitPython（参考）

- Gitをプログラムから操作することが可能.
- 一例としてPythonから実行してみる

1. mkdir –p ~/Library/Python/2.7/lib/python/site-packages
2. easy_install –-install-dir=~/Library/Python/2.7/lib/python/site-packages gitpython

サンプル1

```
#!/usr/bin/env python

from git import *

repo = Repo(”/path/to/ruby”)

master = repo.commit(’trunk’)

master_author = master.author

for commit in repo.iter_commits():

        if commit.committer == master_author:

                print commit
```

サンプル２

```
#!/usr/bin/env python

from git import *

from time import gmtime, strftime

repo = Repo(“/path/to/ruby”)

for commit in repo.iter_commits():

        date = gmtime(commit.committed_date)

        print strftime(“%Y/%m/%d %a %H:%M:%S +0000”, date)
```

サンプル3

```
#!/usr/bin/env python

from git import *

repo = Repo(”/path/to/ruby”)

for commit in repo.iter_commits():

        if “fix” in commit.message:

                print commit.message
```

サンプル4

```
#!/usr/bin/env python

from git import *

git= Git(”/path/to/ruby”)

logs = git.log(“--pretty=format:‘COM%s’”,

                “--date=short”,

                “--name-only”).split(‘COM’)

for log in logs:

        print log
```

演習課題(1/2)

- rubyリポジトリから以下の情報をマイニング
- 答えと答えを得た過程(コマンド)をレポート
  - ✓ **演習1-1: リポジトリ全体で何回コミットされたか**
  - ✓ 演習1-2: リポジトリ全体で何人がコミットしているか
  - ✓ 演習1-3: 多くのコミットを行ったコミッター上位5人は誰か？
  - ✓ **演習1-4: 何月何日のコミットに次のタグが付けられているか**
    - ✓ **v1_9_0_0, v2_0_0_0**

演習課題1-1

- 演習1-1: リポジトリ全体で何回コミットされたか

演習課題1-1

- 演習1-1: リポジトリ全体で何回コミットされたか
  - ✓ Hint: 1コミットを1行で表示し，且つ，wcコマンドを使う

練習課題1-4

- 演習1-4: 何月何日のコミットに次のタグが付けられているか
  - ✓ v1_9_0_0, v2_0_0_0

練習課題1-4

- 演習1-4: 何月何日のコミットに次のタグが付けられているか
  - ✓ v1_9_0_0, v2_0_0_0
  - ✓ Hint: v1_9_0_0タグのコミット日を表示する