오픈소스 소프트웨어 개발 – 12

(Online) Git - basic 2

광운대학교 이윤상
2017년 2학기

이번 시간에 할 것

- 변경내역 확인하기
- 되돌리기
  - 완료한 Commit 수정, Unstaged 상태로 되돌리기, 파일의 수정 사항을 취소하기
- 파일 이름 변경 / 삭제
- 특정 파일을 추적 대상에서 제외 - .gitignore
- 도움말, 참고자료

[실습] 변경 내용 확인하기 | git log

- **git log** : repository에 저장된 commit history를 출력
- **git log –<n>** : 최근 n개의 commit history만 출력
- **git log –p** : commit history와 함께 commit간의 차이점을 출력

(Shell)

```
git log
git log -2
git log -p
```

변경 내용 확인하기 | git diff

- **git diff** : 현재 unstaged 상태(add하기 전)의 변경 사항 출력
- **git diff --staged** : 현재 staged 상태(commit 대기 상태)의 변경 사항 출력
- **git diff HEAD** : 마지막 commit과 현재 working directory의 차이점 출력
- **git diff <commit id1>..<commit id2>** : 두 개의 commit 간의 차이점 출력
- **--word-diff, --color-words** 추가하면 보기 편할 수도 있다.

[실습] 변경 내용 확인하기 | git diff

(Shell)

```
vim file1.txt    # 약간의 파일 수정 (수정1)
git add file1.txt
vim file1.txt    # 또 다른 파일 수정 (수정2)

git diff            # unstaged 상태의 변경 사항 출력 (수정2)
git diff --staged   # staged 상태의 변경 사항 출력 (수정1)
git diff --staged --word-diff # staged 상태의 변경 사항 출력
git diff HEAD    # 마지막 commit과 working dir 차이점 출력
```

- git log의 결과에서 commit id 2개를 복사해서 git diff로 비교해보자.

(Shell)

```
git log    # 비교할 commit id 확인
git diff 1df8e8.. a74fd3
```

[실습] 되돌리기 | 완료한 Commit 수정

- **git commit --amend** : 가장 최근 commit을 수정할 수 있다.
  - 가장 최근 commit의 commit message를 수정하는 용도로 많이 사용함.

(Shell)

```
git log -p     # 가장 최근 commit의 내용을 확인
vi file1.txt   # 파일 내용을 약간 수정
git add file1.txt   # add한 후
git commit –amend

git status     # 상태 확인
git log –p     # 마지막 commit의 내용이 바뀐 것을 확인하자
```

[실습] 되돌리기 | Staged된 파일을 unstaged 상태로

- 파일을 약간 수정하고 add 한 후 git status 해 보자.

(git status 출력 결과)

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

      modified:   file1.txt
```

- 설명에 나온 대로 한 후, git status로 확인.

(Shell)

```
git reset HEAD file1.txt
```

[실습] 되돌리기 | Unstaged 변경사항 취소하기

- 파일을 약간 수정하고 (add 하지 않고) git status 해보자.

(git status 출력 결과)

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

- 설명에 나온 대로 한 후, git status로 확인.

(Shell)

```
git checkout -- file1.txt
```

[실습] 파일 이름 변경, 삭제

- 파일 이름을 변경 혹은 삭제한 후 해당 파일들을 git add하면 알아서 처리된다.

(Shell)

```
mv file1.txt fileA.txt
git status    # 상태 확인
git add file1.txt fileA.txt (혹은 git add .)
git status    # 상태 확인
```

(Shell)

```
rm file1.txt
git status    # 상태 확인
git add .
git status    # 상태 확인
```

참고사항

- git add 등의 명령어에 <파일 이름>이 입력되는 부분에 디렉터리를 대신 입력할 수도 있다.

(Shell)

```
git add .  # 현재 디렉터리 파일을 모두 add
git add dirname  # dirname이라는 하위 디렉터리의 파일을 모두 add
```

- 이렇게 하는 것이 훨씬 편하므로 많이 사용된다.

- 다만 "git add ."과 같은 경우, 추적하고 싶지 않은 파일까지 모두 add될 수 있다는 점에 주의.
  - 이어서 나오는 .gitignore를 사용하면 편하다.

특정 파일을 Git 관리 대상에서 제외

- Git이 관리할 필요가 없는 파일들이 있다.
  - 로그 파일, 빌드할 때 자동으로 생성되는 파일들

- **.gitignore**라는 파일에 파일 패턴(표준 Glob 패턴)을 적으면, Git은 해당 패턴의 파일들을 마치 없는 파일처럼 무시한다.
  - git status에서 Untracked로 표시되지 않는다.

- .gitignore에서 사용가능한 패턴 -> 다음 슬라이드

```
# '#'으로 시작하는 줄은 주석

# 확장자가 .a인 파일 무시
*.a

# 윗 라인에서 확장자가 .a인 파일은 무시하게 했지만 lib.a는 무시하지 않음
!lib.a

# '/'로 시작하면 하위 디렉토리에 적용되지 않는다.
# 확장자가 .tmp인 파일 중 현재 디렉터리에 있는 파일만 무시함.
/*.tmp

# build/ 디렉토리에 있는 모든 파일을 무시
build/

# Bin/ 혹은 bin/ 디렉토리에 있는 모든 파일을 무시
[Bb]in/

# test0.txt, test1.txt, ... test5.txt를 무시
test[0-5].txt

# doc/notes.txt 파일은 무시하고 doc/server/arch.txt 파일은 무시하지 않음
doc/*.txt

# doc 아래의 모든 .pdf 파일을 무시 (하위 디렉토리의 .pdf 파일도 무시)
doc/**/*.pdf
```

[실습] .gitignore

- 10-(Online)CMake-practical 강의에서 작성했던 VectorTest 코드를 이용하자
  - 다음 슬라이드의 파일들을 적당한 디렉터리에 작성 후 빌드

(Shell)

```
mkdir build
cd build
cmake ../
make
./VectorTest
```

- git init 명령으로 저장소를 만들자.

(Shell)

```
git init
```

vector.h

```
class Vector
{
private:
    double x_, y_;
public:
    Vector(double x, double y):x_(x), y_(y) {}
    void print();
    Vector operator+(const Vector& v2);
};
```

vector.cpp

```
#include <iostream>
#include "vector.h"

void Vector::print()
{
    std::cout << "(" << this->x_ << ", " << this->y_ << ")" << std::endl;
}

Vector Vector::operator+(const Vector& v2)
{
    return Vector(this->x_+v2.x_, this->y_+v2.y_);
}
```

main.cpp

```
#include <iostream>
#include "vector.h"

int main()
{
    Vector v1(1.1, 2.2);
    Vector v2(4.2, 5.2);

    v1.print();
    v2.print();
    (v1+v2).print();

    return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.12)
project(VectorTest)

add_executable(${PROJECT_NAME} main.cpp vector.cpp)
```

[실습] .gitignore

(Shell)

```
git status
vi .gitignore  # 아래 내용으로 작성
git status   # build/ 디렉터리가 Untracked files에서 사라짐
```

.gitignore

```
build/
```

- .실제로 사용할 때는 빌드 결과물이 저장되는 build/ 디렉터리를 무시하는 것으로 충분하지만, 연습을 위해 앞의 여러 가지 .gitignore 패턴 예제를 적용하며 git status 확인해보자.

```
# .gitignore 패턴 예제
*.txt
vector.*
CMakeFiles/
```

[실습] .gitignore

- 연습을 다하면, 다시 아래의 .gitignore로 build/ 디렉터리만 무시한 상태에서 파일 전체를 add & commit 하자.

.gitignore

```
build/
```

(Shell)

```
git add .
git commit
```

[실습] 실수로 .gitignore 패턴 추가를 못한 채 commit한 경우

- **git rm --cached** <file> : 이미 commit된 파일을 Git의 관리대상에서 제외함.
  - .gitignore에 패턴을 추가하는 것을 빼먹고 이미 파일을 commit 한 경우 사용하면 됨

- 그 후 아래 내용 실행해보자.

(Shell)

```
git rm --cached *.cpp  # 실제로는 .cpp를 제외하면 안되나, 연습삼아 해보자
git status

# *.cpp를 다시 stage에 올림
git add .
```

다양한 .gitignore 예제

- https://github.com/github/gitignore
  - 다양한 프로그래밍 언어 및 도구에서 사용할 수 있는 .gitignore 파일이 정리되어 있다.

도움말, 참고자료

- git <명령어> --help
  - ex) git commit --help

- Pro Git 한글 버전
  - https://git-scm.com/book/ko/v2/