1-1. 지옥에서 온 관리자, 깃

깃이 제공하는 핵심 기능은 아래와 같다.

**(1) Version Control**

컴퓨터로 문서를 작성한 뒤 수정하는 모습을 상상해보자. 원래 내용도 남겨두고 수정한 내용을 저장해야 할 경우 ‘다른 이름으로 저장’하는 방법을 주로 사용한다. 그리고 다른 파일과 구별하기 위해 원래 파일 이름 뒤에 숫자를 붙이거나 문자를 추가한다. (Ex. ‘초안’, ‘수정’, ‘최종’…)

하지만 1,000개가 넘는 문서들을 수정할 때마다 이런 방식으로 저장한다면 나중에는 어떤 파일에서 어떤 내용을 수정했는지 기억할 수 없을 것이다. 반면 깃은 문서를 수정할 때마다 언제 수정했는지, 어떤 것을 변경했는지 편하고 구체적으로 기록하기 위한 버전 관리 시스템이다.

**(2) Backup**

우리의 컴퓨터는 ‘확실한 것’과 ‘불확실한 것’을 가지고 있다. 확실한 것은 **‘컴퓨터가 언젠가는 고장 난다.’**라는 것이고, 불확실한 것은 **‘내 컴퓨터가 언제 고장 날지 모른다’**는 것이다. 그러므로 자료를 컴퓨터에만 저장한다면 언젠가는 그 자료가 유실될 것이다. 따라서 우리는 Backup을 반드시 해야 한다.

백업 공간을 제공하는 인터넷 서비스 중에는 깃 파일을 위한 여러가지 시스템이 존재한다. 이를 깃의 **원격 저장소** 혹은 **온라인 저장소**라고 부른다. 이러한 서비스 중 가장 많이 쓰이는 것이 바로 **GitHub**이다.

**(3) Collaboration**

GitHub와 같은 온라인 서비스를 사용하면 여러 사람이 함께 일할 수 있다는 강력한 장점이 생긴다. 예를 들어, 팀원 둘이서 작업을 할 경우 팀원 A가 작업을 해서 원격 저장소에 올리면 원격 저장소에 있는 문서를 팀원 B가 내려 받아 작업하고, 작업이 끝나면 그 결과를 다시 원격 저장소에 올린다. 이처럼 깃을 사용하면 팀원들이 **파일을 편하게 주고받으면서 일할 수 있다.** 또한 누가 어느 부분을 어떻게 수정했는지 기록이 남기 때문에 나중에 **오류가 생겼을 때 파악하기 쉽다.**

온라인 저장소를 기준으로 여러 사람이 파일을 수정하고 저장하면 ‘두 사람이 같은 파일을 수정한다.’, ‘두 사람이 같은 파일의 같은 행을 수정한다.’등 여러 문제가 발생할 수 있다. 깃은 이렇게 **협업 과정에서 일어날 수 있는 여러 문제를 중간에서 정리해 주는 기능**도 가지고 있다.

1-2. 깃 설치하기

깃 설치 내용은 책을 참고하자.

깃을 사용하기 전에는 사용자 정보를 입력해야 한다. 깃은 버전을 저장할 때마다 그 버전을 만든 사용자 정보도 함께 저장하기 때문이다. 이를 통해 어떤 버전을 누가 언제 만들었는지 쉽게 파악할 수 있다. 깃에 사용자 정보를 입력하려면 아래와 같이 터미널에 입력하면 된다.

**\* git config --global user.name “GwonPyo”**

**\* git config --global user.email** [**ngp9441@gmail.com**](mailto:ngp9441@gmail.com)

깃에서 **사용자 정보를 설정하려면 git config 명령을 사용**한다. 여기에 **--global 옵션**을 추가하면 **현재 컴퓨터에 있는 모든 저장소에서 같은 사용자 정보를 사용하도록 설정**한다.

(참고) 사용자를 확인하려면 **‘git config –list’**를 입력하면 된다.

1-3. 리눅스 명령 연습하기

**(1) pwd**

**pwd 명령**을 입력하면 **현재 위치의 경로**가 나타난다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

**(2) ls**

현재 디렉터리에 **어떤 파일이나 디렉터리가 있는지 확인**하려면 **ls 명령**을 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

리눅스 명령에 **Option을 추가하려면 붙임표(-)와 원하는 옵션을 나타내는 글자를 함께 입력**한다. 예를 들어, **파일과 디렉터리의 상세 정보까지 표시하는 옵션**을 추가하려면 ls 명령 뒤에 **‘-l’**을 추가로 입력한다. **숨긴 파일과 디렉터리를 표시**하려면 **‘-a’**를 추가 입력한다. 두 옵션을 함께 사용하려면 **‘-la’또는 ‘-al’**처럼 **순서에 상관없이 글자를 붙여서 추가 입력**한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

(ls 명령 option)

-a: 숨김 파일과 디렉터리도 함께 표시

-l: 파일이나 디렉터리의 상세 정보를 함께 표시

-r: 파일의 정렬 순서를 거꾸로 표시

-t: 파일 작성 시간 순으로 (내림차순) 표시

**(3) cd**

현재 위치에서 **상위 디렉터리로 이동**하려면 **cd 명령 다음에 마침표 2개를 입력**한다. cd 명령을 실행한 후 $ 기호 위에 표시된 경로를 확인하면 이전 위치의 상위 디렉터리인 것을 확인할 수 있다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

**하위 디렉터리로 이동**할 때는 **cd 명령 다음에 이동할 하위 디렉터리 이름을 입력**한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**홈 디렉터리**로 돌아가려면 **cd 명령 다음에 ~를 입력**하면 된다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

**(4) mkdir**

**현재 디렉터리 안에 하위 디렉터리를 만들 때**는 **mkdir 명령**을 사용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**(5) rm**

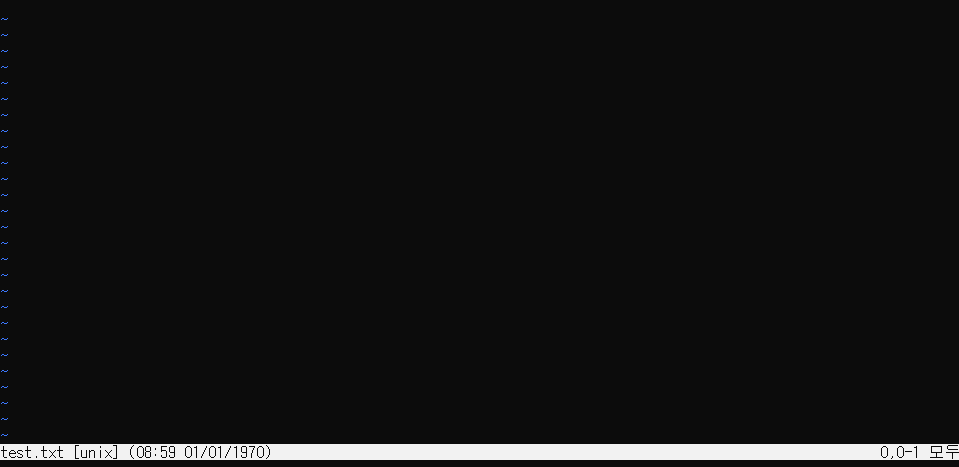
**디렉터리를 삭제**하려면 **rm 명령**을 사용한다. **-r 옵션**을 붙이면 **디렉터리 안에 있는 하위 디렉터리와 파일까지 함께 삭제**된다. 단, 삭제할 디렉터리의 상위 디렉터리에서 rm 명령을 입력해야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**(6) vim**

현재 디렉터리에 test.txt 파일을 만들려면 **vim 명령**을 사용하면 된다. vim 명령은 **뒤에 입력한 파일 이름과 같은 파일이 없다면 그 이름으로 새로운 텍스트 문서를 만들고, 파일이 있다면 그 파일을 연다.** vim 명령을 입력하면 아래와 같은 화면이 뜬다.



해당 화면에서 어떤 내용을 입력하려고 하면 잘 입력되지 않는다. vim 문서는 **문서를 작성하는 ‘입력 모드’**와 **문서를 저장하는 ‘ex 모드’**가 있기 때문이다. 빔은 처음에 ‘ex 모드’로 열려 키를 눌러도 아무런 반응이 없다.

**\* ex 모드 -> 입력모드: i, a를 누르면 변경할 수 있다.**

**\* 입력모드 -> ex 모드: esc를 누르면 변경할 수 있다.**

텍스트이(가) 표시된 사진

자동 생성된 설명

입력모드로 바뀌면 아래에 ‘--끼워넣기--‘라는 단어가 뜬다. 이때부터 텍스트를 입력할 수 있다.

텍스트를 입력하면 ex 모드로 돌아가야 한다. 그리고 콜론(:)을 입력하면 ‘끼워넣기’가 있었던 자리에 텍스트를 입력할 수 있다. ‘:wq’를 입력하면 입력했던 텍스트를 저장하고 종료한다. 해당 프로그램을 종료하면 빔을 실행했던 터미널 창으로 돌아간다.

(Vim 명령 모음)

:w, :write 편집 중이던 문서를 저장한다.

:q, :quit 편집기를 종료한다.

:wq 편집 중이던 문서를 저장하고 종료한다.

:q! 문서를 저장하지 않고 편집기를 종료한다.

**(7) cat**

터미널 창에서 간단히 **텍스트 문서의 내용을 확인**할 때는 **cat 명령**을 사용한다. cat 명령 다음에 텍스트 파일 이름을 입력하면 터미널 화면에 텍스트 파일의 내용을 보여준다.

텍스트이(가) 표시된 사진

자동 생성된 설명