LINUX & GIT 보고서

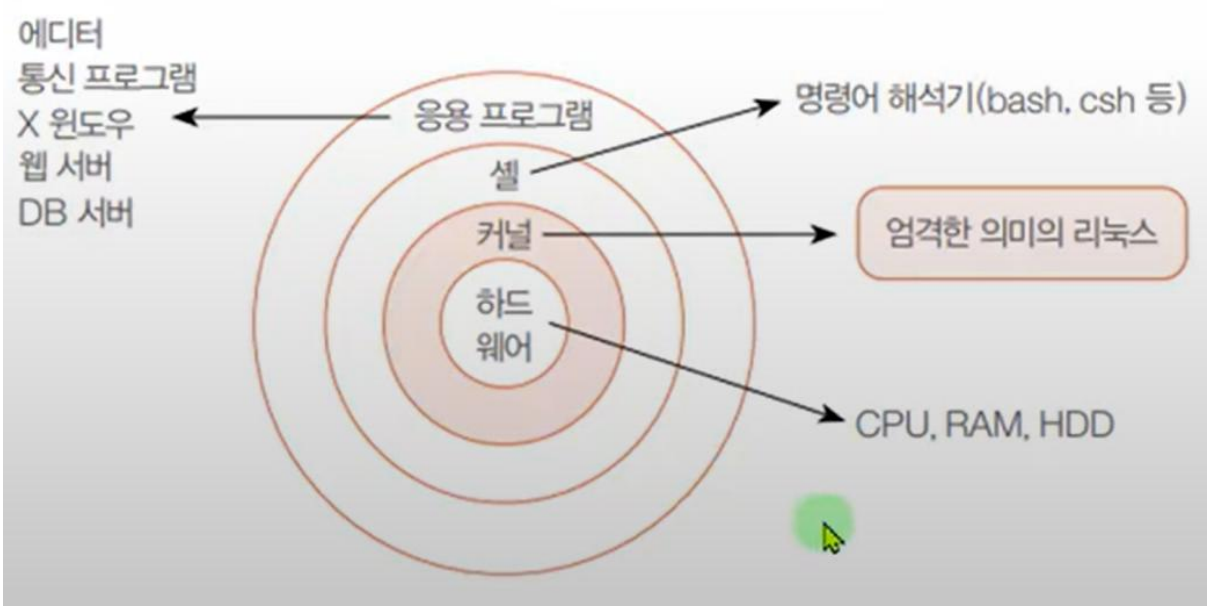
2023741024_로봇학부_박건후

목차

(1) LINUX

1-1. LINUX 배경 및 장점

리눅스는 무료 유닉스와 같습니다. 1991년 리누스 트로발스가 커널을 개발하였습니다. 하지만 이것은 리눅스가 아닌 커널만 개발한 것입니다. 커널이라는 것 위에 셸, 응용 프로그램 등이 추가로 개발되어 이것이 배포되는 것이 리눅스입니다.



위의 사진처럼 하드웨어 위에 리눅스의 엔진 개념인 커널이 있고 그 위에 셸, 응용 프로그램까지 세트로 배포되는 것입니다.

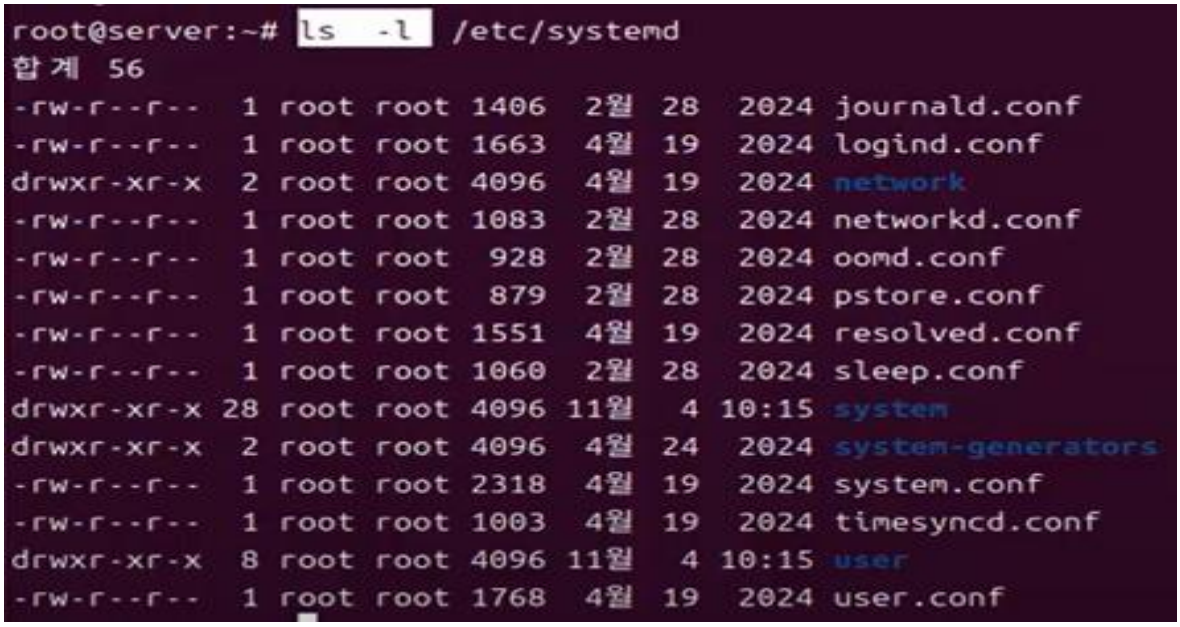
리처드 스톨만이 GNU 프로젝트를 시작하였고, 모두가 공유할 수 있는 소프트웨어를 목표로 만들었습니다. 컴퓨터 프로그램의 복제, 변경, 소스 코드의 사용에 대한 제한을 철폐하는 것을 목표로 했던 것입니다. GNU에서는 이런 목표를 실현시키기 위해 리눅스라는 커널을 채택하였고 GNU라는 프로젝트 덕분에 이 커널 위에 다양한 소프트웨어가 결합되어 다양한 배포판 리눅스 운영체제가 만들어지고 배포되었습니다. 그 중에는 저희가 사용하는 우분투 리눅스도 이에 해당합니다.

우분투 리눅스의 장점으로 강력한 개발 도구와 환경을 제공하여 소프트웨어 개발에 최적화되어 있습니다. 그리고 안정성과 보안이 뛰어나서 서버, 데이터 베이스 등 서버 운영에 적합합니다. 우분투 리눅스를 쓰게 되면, 터미널에 입력을 하여 파일과 폴더를 관리하는 경우가 많게 됩니다. 이를 모두 터미널에서 잘 알려진 명령어를 통해 간단하고 빠르게 처리할 수 있다는 것이 우분투 리눅스의 매력이자 장점인 것 같습니다.

1-2. LINUX 명령어

ls

ls는 현재 디렉터리, 위치에 있는 폴더나 파일 목록을 알려주는 명령어입니다. 그리고 여기서 'ls -l'을 쓰게 되면 파일명, 수정일, 파일 종류 등등의 정보를 상세히 알려줍니다.



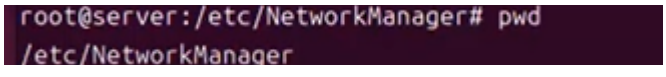
위의 사진처럼 파일의 날짜, 이름, 폴더인지 파일인지에 대한 정보들이 출력됨을 볼 수 있습니다.

cd

cd는 디렉터리를 이동하는 명령어입니다. cd "디렉터리명"을 쓰면 현재 디렉터리에 있는 폴더로 이동하게 되고 만약 그 폴더가 현재 디렉터리에 없을 시 이동할 수 없습니다. 이 방법은 상대 경로로 이동하는 것이며 하위 경로로 움직일 수 있습니다. 절대 경로로도 이동이 가능합니다. 이 때는 '/'라는 루트 기호를 사용하여 처음부터 원하는 경로까지 cd 라는 명령어 이후에 기입하면 됩니다. '.'은 현재 디렉터리를 의미합니다. '..'은 바로 이전 경로를 의미합니다. 이를 통해 "cd .."으로 그 이전 디렉터리로, "cd ../.."을 사용하면 이전의 이전 디렉터리로 이동할 수 있습니다.

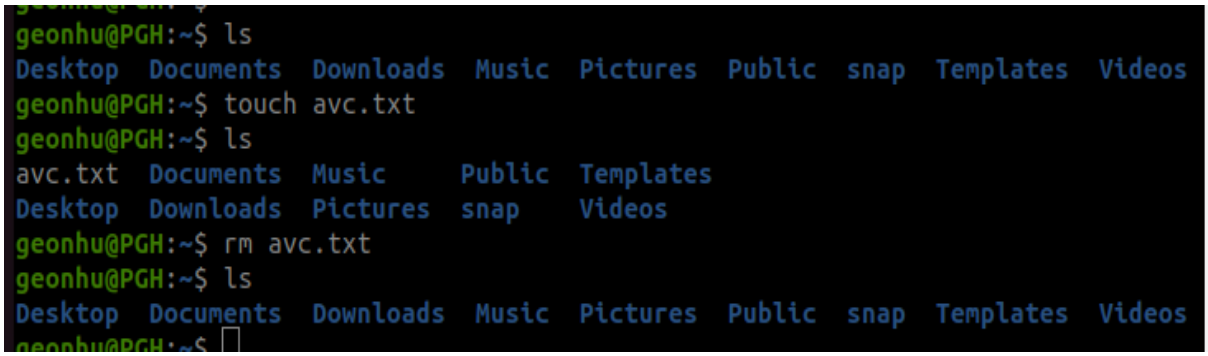
pwd

pwd는 현재 디렉터리의 전체 경로를 출력합니다. 아래의 사진과 같이 pwd 명령어를 사용하면 현재 디렉터리에 대한 절대 경로를 출력해줍니다.



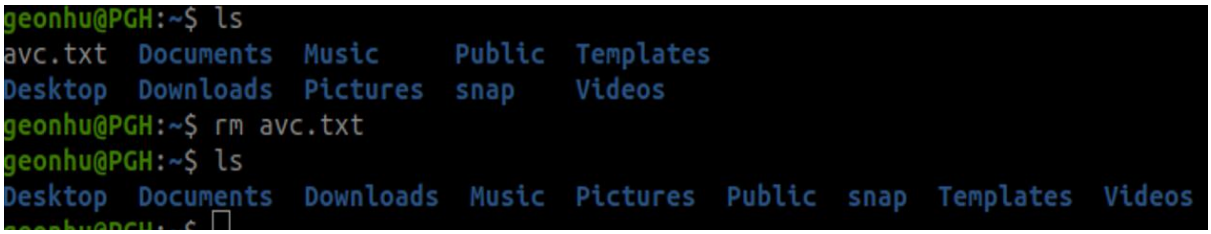
touch

touch 명령어는 파일을 생성하는 명령어입니다. "touch 파일명"을 쓰면 크기가 0인 새 파일을 현재 디렉터리에 생성하게 됩니다. 따라서 아래 사진처럼 avc.txt라는 텍스트 파일을 touch명령어로 생성하면 현재 디렉터리에 avc.txt파일이 생성된 것을 보실 수 있습니다. 물론 절대 경로를 기입하여 원하는 경로에 파일을 생성할 수도 있습니다.



rm

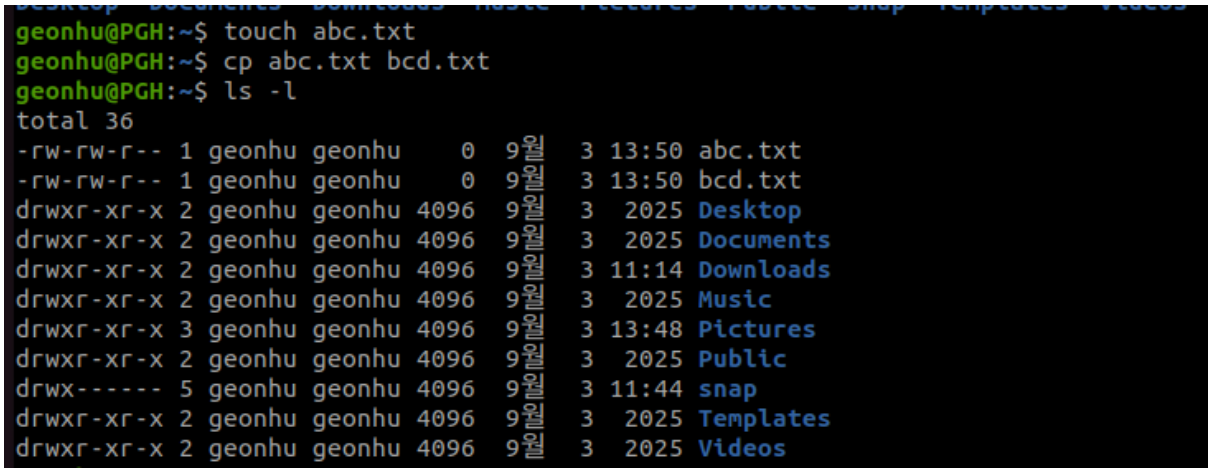
rm 명령어는 파일을 지우는 명령어입니다. "rm 파일명"을 사용하면 해당 디렉터리에 있는 파일을 지울 수 있습니다.



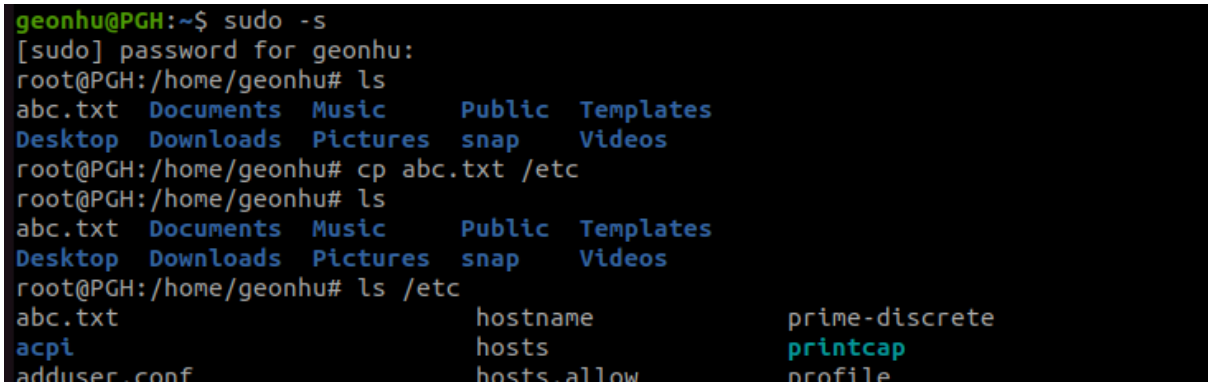
아래 사진과 같이 rm 명령어을 사용하니 avc.txt파일이 현재 경로에서 지워진 것을 볼 수 있습니다. 추가로 rm -rf로 디렉터리를 지울 수 있습니다.

cp

cp는 파일을 복사하는 명령어입니다. "cp 원본 파일명 복사 파일명"을 쓰면 현재 디렉터리에 복사된 파일이 만들어집니다. 아래의 사진과 같이 abc파일을 bcd 파일로 복사하여 복사된 bcd파일이 같은 경로에 존재하게 된 모습입니다.



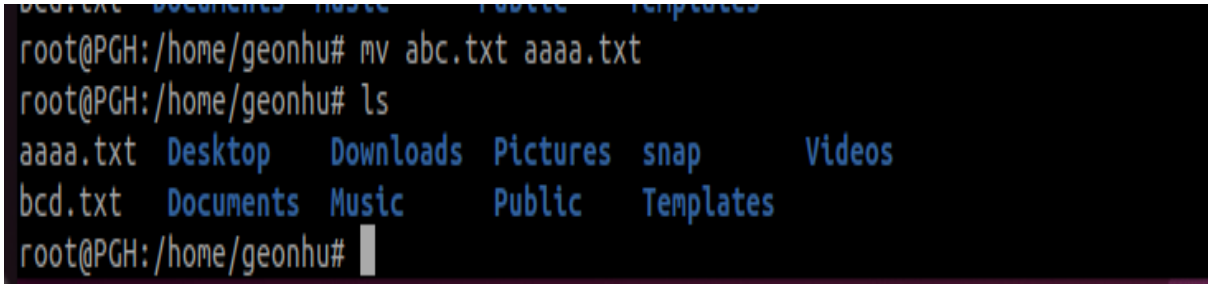
또한 복사할 때 다른 경로로 파일을 복사시킬 수도 있습니다. "cp 원본 파일명 복사시킬 경로"로 적으면 기입해준 경로로 파일이 복사됩니다. 아래의 사진을 보시면 cp명령어로 복사 이후에 현재 경로에도 abc파일이 있고 /etc경로에도 abc파일이 존재한다는 것을 확인할 수 있습니다.



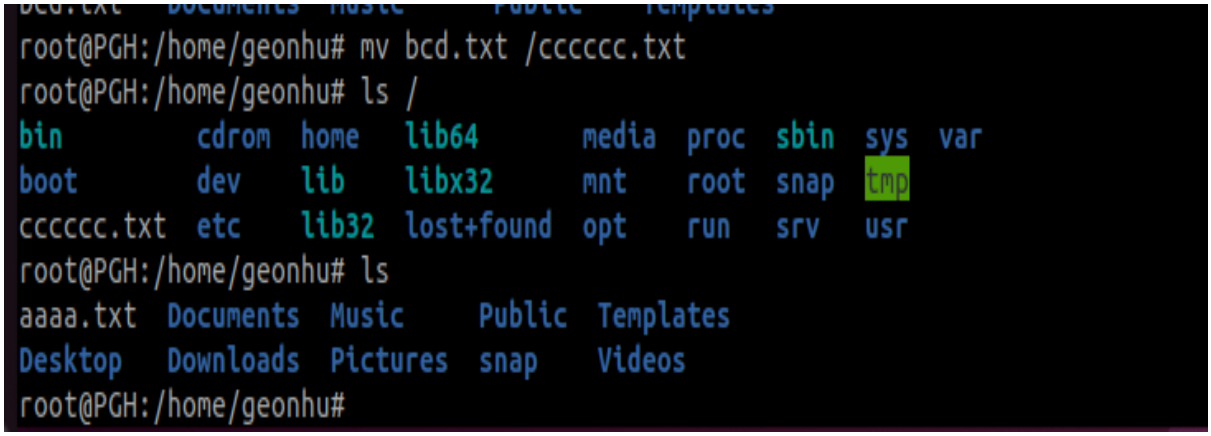
mv

mv는 파일을 이동시키는 명령어입니다. "mv 이동시킬 경로"를 적으면 해당 경로로 파일이 이동됩니다. 아래 사진에서는 '/'라는 루트 경로로 abc파일을 옮기는 모습입니다.

그뿐만 아니라, mv명령어를 통해 원본 파일의 파일명을 바꿀 수도 있습니다. "mv 원본 파일명 새로운 파일명"을 입력하면 해당 경로에 새로운 파일명으로 파일명이 바뀌게 됩니다. 아래 사진으로는 abc.txt파일이 aaaa.txt파일로 이름을 바꾸는 명령어를 쓴 것이고 실제로 abc.txt파일이 있던 자리에 aaaa.txt파일명으로 파일명이 바뀐 것을 알 수 있습니다.

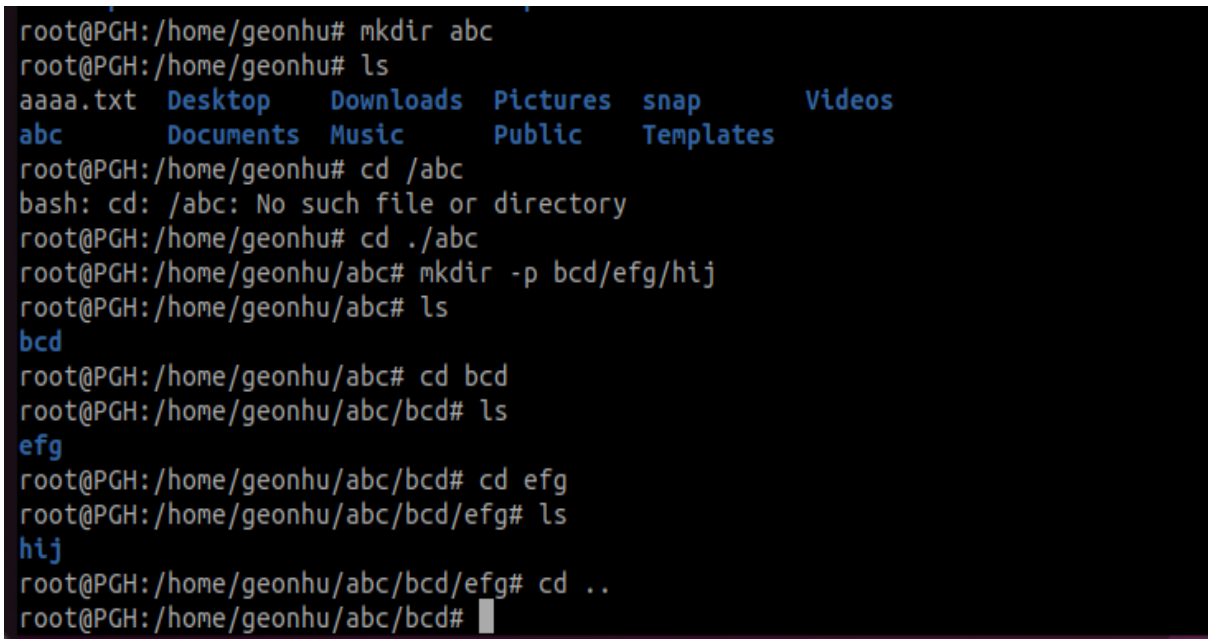


이를 활용하면 현재 디렉터리에 있던 파일을 파일명을 바꾸며 옮길 수도 있습니다. 그러려면 경로와 새로운 파일명까지 한꺼번에 적으면 됩니다. 경로와 새로운 파일명까지 적으면 그것 또한 경로가 되기에 해당 경로에 새로운 파일명으로 된 파일이 생기며 이동되는 것입니다. 아래의 사진을 보면, bcd.txt파일을 cccccc.txt파일로 파일을 이동시키며 파일명을 바꾸는 모습입니다. '/'디렉터리에 cccccc.txt파일이 생성되고 파일이 이동되었기에 현재 디렉터리에 bcd.txt파일이 없어졌음을 알 수 있습니다.



mkdir

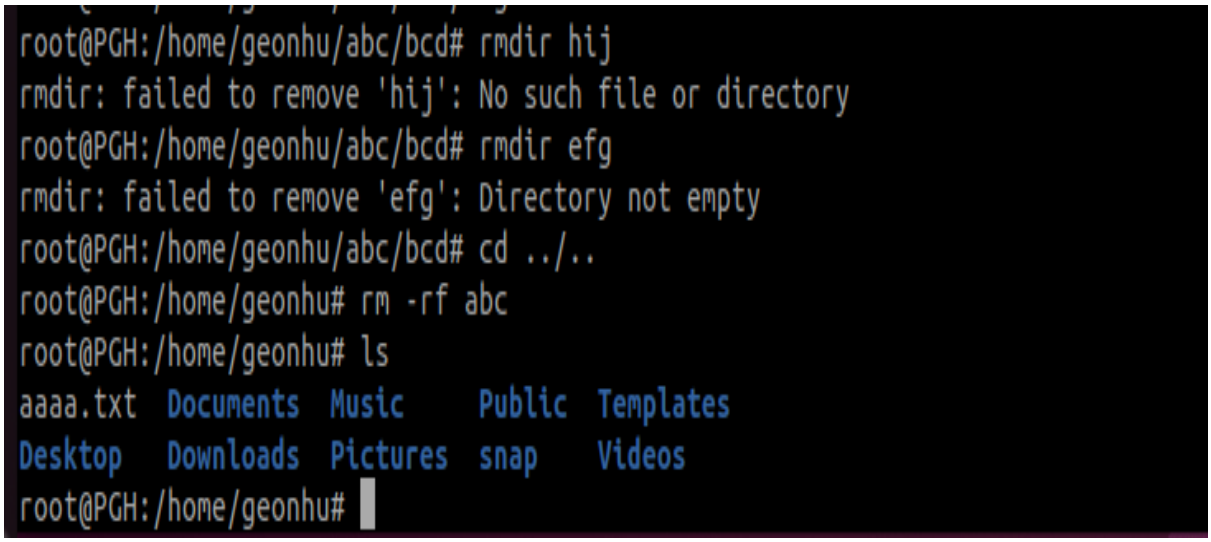
mkdir은 경로를 새로 만드는 명령어입니다. 즉 폴더를 새로 만드는 명령어입니다. "mkdir 폴더명"을 사용하면 현재 디렉터리에서 새로운 폴더를 생성합니다. 이 때 폴더명을 입력할 때 전체경로를 입력하며 폴더명을 입력하면 해당 경로에 폴더가 만들어지게 됩니다. 이렇게 폴더를 생성할 때 한 번에 하위 디렉터리를 여러 개 만들 수도 있습니다. 이럴 때는, "mkdir -p 만들 하위 디렉터리"를 입력하면 연속적인 하위 디렉터리를 만들 수 있게 됩니다. 아래 사진을 보면, mkdir -p 를 사용하여 없던 bcd, efg, hij라는 폴더를 하위 디렉터리로 연속으로 생성하는 모습입니다. abc라는 폴더 아래에 bcd가 있고 bcd 폴더 아래에 efg라는 폴더가 있는 것입니다.



rmdir

rmdir은 경로를 지워주는 명령어로 "rmdir 폴더명"을 쓰면 바로 하위 디렉터리가 지워지게 됩니다. 하지만 그렇게 지우기 위해서는 지우려는 폴더에 아무 내용이 없어야 합니다. 그래서 폴더의 내용을 모두 삭제하고 나서 지워야 합니다. 하지만 폴더 속 내용을 모두 지우지 않고도 폴더를 지울 수 있습니다. 그것은 rm 명령어를 사용하여 "rm -rf 폴더명" 을 사용하는 것입니다. 이러면 폴더 속에 있던 내용과 폴더 모두 지워버립니다.

아래의 사진을 보면 생성한 abc, bcd, efg, hij 폴더 중 efg 폴더를 지우고자 할 때 Directory not empty라는 문구로 efg폴더가 지워지지 않는 것을 볼 수 있습니다. 그 후에 geonhu경로로 이동하여 rm -rf로 4개의 생성된 폴더 중 제일 상위 디렉터리인 abc디렉터리를 지울 수 있었습니다. 그러면 abc의 하위 디렉터리 모두 소멸됩니다.



(2) GIT

2-1 GIT의 장점

git은 분산형 버전 관리 시스템으로 소스 코드와 파일의 변경 이력을 효율적으로 관리할 수 있는 오픈 소스 소프트웨어입니다. 그리고 이를 올릴 수 있는 웹 서비스로 github가 있는 것입니다. git을 통해 어떤 사람이 언제 어떻게 변경한 것인지 알 수 있기에 빠른 처리 속도와 협업이 가능해집니다. 그래서 여러 개발자가 동시에 작업할 때 발생하는 충돌을 최소화할 수 있습니다. 이를 통해 프로젝트의 안정성을 유지할 수 있게 됩니다. 또한 이전에 작성했던 코드를 다시 되살릴 수 있고 실수로 인한 데이터 손실을 방지할 수 있습니다.

git은 로컬 저장소와 원격 저장소에서 모두 작업이 가능하기 때문에 서버 없이도 로컬에서 작업 가능하며 서버가 손상되더라도 로컬에서 복구하는 것이 가능합니다. 이로 인해 데이터 손실을 최소화할 수 있습니다. 이 git은 텍스트나 소스 파일뿐만 아니라 이미지, 오디오 등등의 다양한 형식의 파일을 관리할 수 있기에 더욱 유연합니다.

이런 git을 올리는 곳이 github이고 github를 통해 자신의 git 저장소를 업로드 하고 다른 사람과 쉽게 변경 사항을 공유하고 협업할 수 있습니다.

2-2. GIT 명령어

git init

git으로 프로젝트를 관리하려면 저장소를 만들어야 합니다. 이 때 새 저장소를 만들기 위해 init이라는 명령어를 사용합니다. 이것을 사용하게 되면 현재 위치에서 저장소를 생성하게 됩니다. 또는 git init <URL>로 기존 github에 있는 저장소를 내 로컬로 복제할 수도 있습니다.

git clone

저장소를 복사하는 것도 가능합니다. 복사할 때에는 clone이라는 명령어를 사용합니다. 그래서 'git clone <URL>'을 명령어를 작성하면 github의 원격 저장소에 있던 내용을 로컬로 복사하여 가져올 수 있습니다.

```
git clone https://github.com/sessac-multi-docu/rag.git
```

git push

프로젝트 진행 시, 중간에 에러가 발생한다면 이를 고쳐야 합니다. 하지만 그와 동시에 지금 진행하고 있는 프로젝트에서 새로운 모듈을 개발해야 할 경우 위 두 코드는 독립적으로 수행되어야 합니다. 따라서 master (기본 브랜치) 외에 다른 브랜치가 하나더 필요합니다. 따라서 바뀐 브랜치에서 commit을 하면, 원래 브랜치에는 업데이트가 안되고, 새로운 브랜치에서만 업데이트가 됩니다. 그 후에 push를 하면 master 브랜치에 업데이트가 되는 것입니다. git push <저장소 이름> <브랜치> 명령어를 통해 원격저장소의 master 브랜치에 푸쉬합니다.

```
git push                          # 기본 브랜치로 푸시
git push origin 브랜치명   # 특정 브랜치로 푸시
```

git pull

다른 사람이 원격 저장소(Remote repository)에 업데이트한 파일이 있을 때, 원격저장소와 내 로컬저장소의 상태를 동일하게 만들기 위해 pull을 이용합니다.

```
git pull origin 브랜치명
```

git submodule

git submodule add 명령으로 외부 모듈을 현재 작업 디렉터리에 추가(clone)할 수 있습니다.