

리눅스란



리눅스는 유닉스(Unix)라는 운영체제를 기반으로 하고 있으며, 뛰어난 안정성과 보안성, 높은 신뢰성과 성능이 특징입니다. 시스템의 자원을 효율적으로 관리 및 사용할 수 있으며, 멀티 유저(multi-user)와 멀티 태스킹(multi-tasking)을 지원하고 있습니다.

또한, 대부분의 리눅스는 CLI(명령어창)와 GUI(그래픽)를 모두 지원하고 있으며, 다양하고 강력한 네트워킹 기능 덕분에 서버 OS로 적합합니다. PC 서버에서도 엔터프라이즈 급의 성능을 제공하고, 성능이 낮은 PC에서도 작동합니다. 앞서 언급한 것과 같이 오픈소스 프로젝트이기 때문에 커널 소스코드 및 모든 관련 자료가 공개되어 빠른 발전을 지원하고 있습니다. 다양한 업무 환경을 만족시키는 다양한 배포판이 존재하고 풍부한 응용프로그램을 제공하고 있습니다.

가장 유명하고 사용성이 좋은 우분투(Ubuntu)와 사용자 인터페이스가 잘 갖춰진 페도라(Fedora), 라즈베리파이에서 자주 쓰이는 라즈비안, 우리에게 친숙한 안드로이드까지 모두 리눅스의 한 종류입니다.

우분투란



우분투(Ubuntu, (/ʊˈbʊntu:/))는 영국 기업 캐노니컬이 개발, 배포하는 컴퓨터 운영 체제이다. 데비안 리눅스를 기반으로 개발되며, 데비안에 비해 '사용자 편의성'에 초점을 맞춘 리눅스 배포판이다.

일반적으로 매 6개월마다 새로운 판이 공개되며, 장기지원판(LTS)은 2년에 한 번씩 출시된다.

우분투라는 이름은 창업자 마크 셔틀워스의 고향 남아프리카 공화국의 건국 이념인 우분투 정신에서 유래했다. 남아프리카 성공회의 데즈먼드 투투 대주교에 따르면 반투어 우분투에는 일일이 옮기기 힘들 정도로 많은 뜻이 담겨있지만, 캐노니컬은 "타인을 향한 인간애(humanity to others)", 또는 "네가 있으니 내가 있다(I am what I am because of who we all are)"라는 의미로 사용한다.

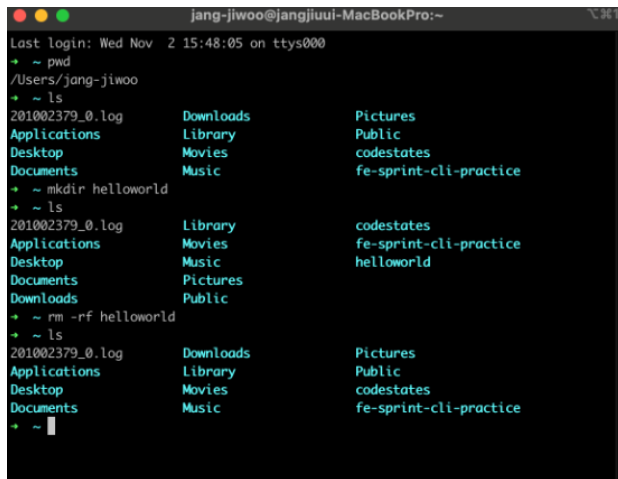
2012년 기준으로 온라인 설문 조사 결과에 따르면 우분투는 개인용 데스크톱과 노트북에서 가장 인기있는 리눅스 배포판이다.

우분투는 배포판을 수정하거나 수정한 것을 재배포할 수 있는 자유 소프트웨어로, 지금까지 수많은 변형 배포판이나 공식 지원하지 않는 창 관리자를 데스크톱으로 하는 배포판들이 나왔다. 우분투의 기본적인 철학, 즉 전 세계의 사람 누구나 어렵지 않게 리눅스를 사용하자는 표어에서 알 수 있듯이, 기본적으로 세계의 다양한 언어를 지원하고 그다지 높은 사양의 컴퓨터가 필요하지 않다.

우분투는 사용자가 손쉽게 운영 체제를 설치하고 사용할 수 있도록 설계되었다. 이를테면 7.10의 경우 CD를 넣고 시동한 다음, 일곱 단계만 거치면 바로 사용할 수 있도록 설치된다. Dapper(6.06) 버전부터 데스크톱 CD에는 유비쿼티가 들어 있어 컴퓨터의 재시동 없이 데스크톱 CD를 구동하는 동안 우분투를 시스템에 설치할 수 있는 기능이 있다. 또한 아직까지 그래픽 카드에 따라 데스크톱 관리자가 시동이 되지 않는 경우가 많기 때문에 텍스트 기반에서 설치를 할 수 있는 얼터너티브 CD를 따로 배포한다. 버전 11.04부터는 유니티가 기본 데스크탑으로 탑재된다. 그리고 20.04 현재 그놈이 기본 데스크탑이다.



Gui 코딩: 사용자가 편리하게 사용할 수 있도록 기능을 아이콘, 이미지 등의 그래픽으로 나타낸 인터페이스입니다. 마우스 클릭이나 드래그앤드롭이 가능하고 수시로 확인이 가능하여 사용이 쉽습니다. 흔히 사용하는 Windows 와 Mac 운영체제 모두 지원하고 있습니다.



```
jang-jiwoo@jangjiuui-MacBookPro:~
Last login: Wed Nov 2 15:48:05 on ttys000
+ ~ pwd
/Users/jang-jiwoo
+ ~ ls
201002379_0.log      Downloads      Pictures
Applications         Library       Public
Desktop              Movies        codestates
Documents            Music         fe-sprint-cli-practice
+ ~ mkdir helloworld
+ ~ ls
201002379_0.log      Library       codestates
Applications         Movies        fe-sprint-cli-practice
Desktop              Music         helloworld
Documents            Pictures
Downloads            Public
+ ~ rm -rf helloworld
+ ~ ls
201002379_0.log      Downloads      Pictures
Applications         Library       Public
Desktop              Movies        codestates
Documents            Music         fe-sprint-cli-practice
+ ~
```

Cli 코딩: 문자로 사용자와 컴퓨터가 상호작용하여 동작하는 인터페이스입니다. Windows 의 CMD, Mac 의 Terminal 에서 CLI 를 사용할 수 있습니다.

리눅스 명령어

ls : 현재 디렉토리의 모든 파일 및 폴더를 기본 형식으로 보여준다.

ls -l : 파일 및 폴더에 대한 자세한 정보와 함께 리스팅 한다.

ls -a : 숨겨진 파일을 포함하여 모든 파일을 보여준다.

ls : 현재 디렉토리의 모든 파일 및 폴더를 기본 형식으로 보여준다.

ls -l : 파일 및 폴더에 대한 자세한 정보와 함께 리스팅 한다.

ls -a : 숨겨진 파일을 포함하여 모든 파일을 보여준다.

ls : 현재 디렉토리의 모든 파일 및 폴더를 기본 형식으로 보여준다.

ls -l : 파일 및 폴더에 대한 자세한 정보와 함께 리스팅 한다.

ls -a : 숨겨진 파일을 포함하여 모든 파일을 보여준다.

ls : 현재 디렉토리의 모든 파일 및 폴더를 기본 형식으로 보여준다.

ls -l : 파일 및 폴더에 대한 자세한 정보와 함께 리스팅 한다.

ls -a : 숨겨진 파일을 포함하여 모든 파일을 보여준다.

cd는 디렉토리를 변경한다.

cd Documents : 현재 디렉토리에서 'Documents'라는 이름의 폴더로 이동한다.

`cd ..` : 현재 디렉토리의 상위 폴더로 이동한다. (즉, 되돌아간다.)

`pwd` : 현재 작업 중인 디렉토리의 경로를 표시한다.

`mkdir`은 새로운 디렉토리(폴더)를 생성한다.

`mkdir new_folder` : 현재 디렉토리에 'new_folder'라는 이름의 새 디렉토리를 만든다.

`mkdir -p folder1/folder2` : 'folder1'내에 'folder2'를 생성한다. -p 는 상위 디렉토리가 없는 경우, 그 상위 디렉토리를 생성하는 옵션이다.

`rmdir`은 디렉토리를 삭제한다.

`rmdir old_folder` : 'old_folder'라는 이름의 디렉토리를 삭제한다.

`rmdir`은 디렉토리가 비어있을 때만 작동한다. 내부에 파일이나 다른 디렉토리가 있으면 오류가 발생한다. 디렉토리 안의 파일과 함께 삭제하려면 `rm -r` 명령어를 사용해야 한다.

`touch`는 새로운 빈 파일을 생성하거나, 기존 파일의 타임스탬프(날짜 및 시간 정보)를 현재 시간으로 갱신한다.

`touch new_file.txt` : 'new_file.txt'라는 새 파일을 생성한다. 파일이 이미 존재한다면, 타임스탬프가 갱신된다.

간단하고 빠르게 파일을 생성할 수 있으며, 스크립트나 로그 파일을 초기화할 때 유용하다.

`cp`는 파일이나 디렉토리를 복사한다.

`Cp -i, --interactive` : 복사대상 파일이 있을 경우, 사용자에게 복사에 대한 실행 여부를 묻는다

`Cp -f, --force`: 복사대상 파일이 있을 경우, 사용자에게 확인없이 강제로 복사한다

`Cp -r, -R, --recursive` : 디렉토리를 복사할 경우 하위 디렉토리 및 파일을 모두 복사한다

`Cp -v, --verbose`: 복사진행 상태를 출력한다

`Cp -d, --no-dereference`: 복사대상 파일이 심볼릭 파일이면, 심볼릭 정보를 그대로 유지한 상태로 복사한다

`Cp -p, --preserve`: 원본 파일의 소유주, 그룹, 권한, 시간정보를 보존하여 복사한다

`Cp a, --archive (-dpr)`: 원본 파일의 속성, 링크정보들을 그대로 유지하면서 복사한다.

mv는 파일이나 디렉토리의 위치를 이동시키거나 이름을 변경한다.

mv old_name.txt new_name.txt : 'old_name.txt'의 'new_name.txt'로 변경한다.

mv file.txt /path/to/directory/ : 'file.txt'를 지정된 디렉토리로 이동한다.

mv 명령어는 파일을 이동시킬 때 복사 후 삭제하는 것이 아니라 파일의 위치정보만 변경하기 때문에 처리 속도가 빠르다.

이동하려는 대상 경로에 같은 이름의 파일이 이미 존재할 경우, 기존 파일은 덮어쓰여진다.

cat은 텍스트 파일의 내용을 화면에 출력하거나, 여러 파일의 내용을 연결하여 출력한다.

cat file.txt : 'file.txt' 파일의 내용을 화면에 표시한다.

cat file1.txt file2.txt > combined.txt : 'file1.txt'와 'file2.txt'의 내용을 합쳐 'combined.txt'에 저장한다.

chmod는 파일이나 디렉토리의 권한을 변경한다.

chmod 755 file.sh : 'file.sh'파일에 대해 소유자에게는 읽기, 쓰기, 실행

권한을 부여하고, 그룹과 기타 사용자에게는 읽기와 실행 권한만 부여한다.

`chmod u+x file.sh` : 'file.sh' 파일에 대해 현재 사용자에게 실행 권한을 추가한다.

권한 변경은 보안에 영향을 줄 수 있으므로 신중히 사용해야 한다.

Grep 파일 내용 중에서 지정된 패턴이나 문자열을 검색하여 그 결과를 출력한다.

`grep "text" file.txt` : 'file.txt'에서 "text"라는 문자열이 포함된 모든 줄을 표시한다.

`grep -r "text" .` : 현재 디렉토리와 하위 디렉토리에서 "text" 문자열을 재귀적으로 검색한다.

정규 표현식을 사용하여 복잡한 검색 패턴을 지정할 수 있으며, 로그 파일 분석이나 특정 데이터 추출에 유용하다.

Echo: 주어진 문자열을 터미널에 출력한다. 환경 변수의 값을 표시하거나, 파일에 텍스트를 쓰는 데에도 사용된다.

`echo "Hello World"` : 터미널에 "Hello World"라는 문구를 출력한다.

`echo $HOME` : 'HOME' 환경 변수의 값을 출력한다.

echo "some text" > file.txt : "Some text"라는 문구를 'file.txt'파일에 저장한다.

스크립트 작성 시 변수의 값을 확인하거나, 파일에 내용을 빠르게 추가할 때 유용하다.

sudo (SuperUser DO): 일반 사용자가 관리자(superuser) 권한을 가지고 명령어를 실행할 수 있게 한다. 시스템 설정 변경, 중요한 파일 수정, 관리자 권한을 필요로 하는 소프트웨어 설치 시 사용된다.

sudo apt-get update : 패키지 리스트를 업데이트 한다.

Find: 파일이나 디렉토리를 검색한다.

find . -name "file.txt" : 현재 디렉토리에서 'file.txt' 파일을 찾는다.

find / -type d -name "config" : 루트 디렉토리에서 'config'라는 이름의 디렉토리를 찾는다.

top 실시간으로 시스템의 프로세스와 리소스 사용량을 모니터링한다. CPU 사용량, 메모리 사용량, 실행 중인 프로세스 목록 등을 표시한다. 시스템 성능 분석과 문제 해결에 필수적인 도구이다.

df -h 디스크 공간 사용량을 사람이 읽기 쉬운 형태로 표시한다. -h

옵션은 용량을 GB, MB 단위로 표시하여 가독성을 높인다. 시스템의 저장 공간 관리에 유용하다.

tar 파일과 디렉토리를 압축하거나 해제하는 데 사용된다. tar -czf [아카이브명].tar.gz [디렉토리명] : 디렉토리를 압축한다. tar -xzf [아카이브명].tar.gz : 압축을 해제한다.

ps aux 현재 실행 중인 모든 프로세스의 상세 정보를 표시한다. CPU와 메모리 사용량, 프로세스 상태 등을 확인할 수 있다. 시스템에서 실행 중인 프로세스를 모니터링하고 관리하는 데 사용된다.

netstat 네트워크 연결, 라우팅 테이블, 인터페이스 통계 등을 표시한다. netstat -tuln : 활성화된 모든 TCP, UDP 포트를 표시한다. 네트워크 문제 진단과 모니터링에 필수적인 도구이다.

history 이전에 실행했던 명령어들의 히스토리를 표시한다. !숫자 : history에서 해당 번호의 명령어를 다시 실행한다. 자주 사용하는 명령어를 다시 입력하지 않고 실행할 수 있어 효율적이다.

ln -s [대상경로] [링크경로] 심볼릭 링크(바로가기)를 생성한다. 파일이나 디렉토리에 대한 참조를 만들 때 유용하다. 긴 경로를 짧게 만들거나, 여러 위치에서 같은 파일을 참조할 때 사용된다.

curl은 다양한 프로토콜을 사용하여 데이터를 전송한다.

curl -O : 지정된 URL에서 파일을 다운로드한다. API 테스트나 웹 리소스 다운로드에 자주 사용된다.

awk는 텍스트 처리와 패턴 검색을 위한 강력한 도구이다.

awk '{print \$1}' : 각 줄의 첫 번째 필드를 출력한다. 로그 파일 분석이나 텍스트 데이터 처리에 유용하다.

Sed: 스트림 편집기로, 텍스트 변환과 대체를 수행한다.

sed 's/old/new/g': 파일 내의 모든 'old'를 'new'로 대체한다. 대량의 텍스트 파일을 빠르게 수정할 때 사용된다.

Crontab: 주기적으로 실행될 작업을 예약한다.

crontab -e : 크론 작업 편집 crontab -l : 현재 설정된 크론 작업 목록 표시 백업 스크립트나 정기적인 작업 자동화에 활용된다.

Htop: top의 향상된 버전으로, 더 직관적인 인터페이스를 제공한다. 프로세스를 필터링하고 정렬할 수 있으며, 마우스 조작도 지원한다. 시스템 리소스 모니터링에 탁월하다.

Lsof: 열린 파일 목록을 표시한다.

lsof -i : 네트워크 연결 정보 표시

lsof -u: 특정 사용자가 사용 중인 파일 표시 시스템 문제 해결과 보안 감사에 유용하다.

Watch: 명령어를 주기적으로 실행하고 출력을 표시한다.

watch -n 1: 1초마다 명령어를 실행하고 결과를 갱신한다. 시스템 상태나 프로세스를 실시간으로 모니터링할 때 사용된다.

journalctl 시스템 로그를 조회하고 관리한다.

journalctl -u: 특정 서비스의 로그만 표시

journalctl -b : 현재 부팅 이후의 로그만 표시 시스템 문제 진단과 모니터링에 필수적이다.

git 과 Github이란?

git (깃) 은 분산 버전 관리 시스템(DVCS)으로, 소프트웨어 개발 과정에서 소스 코드의 변경 사항을 추적하고 관리하는 데 사용됩니다. git을 사용하면 여러 개발자가 동시에 작업할 수 있으며, 변경 내용을 효과적으로 병합하고 관리할 수 있습니다.

GitHub (깃헙) 은 git 을 기반으로 한 웹 호스팅 서비스로, 개발자들이 협업하고 소스 코드를 공유할 수 있는 플랫폼을 제공합니다. Github을 통해 프로젝트를 공개적으로 또는 비공개로 관리할 수 있으며, 이슈 트래킹, 풀 리퀘스트 등의 기능을 활용할 수 있습니다.

Github 기본 용어 정리

1)Local/Remote

- Local : 우리가 사용하고 있는 컴퓨터
- Remot : 원격 저장소

2) Repository (repo, 저장소)

- 프로젝트가 존재하는 저장 공간

3)Branch (브랜치)

- Repository 의 공간에서 독립적으로 어떤 작업을 하기 위한 공간

4)Commit

- 소스코드의 업데이트를 확정. 확정된 순간의 코드 상태를 메시지와 함께 Git Repo에 저장

* 로컬 저장소에는 변경이 반영되지만, 원격 저장소에는 아직 반영되지 않은 상태 (Push를 해주어야 반영된다)

5)Pull/Push

- Pull : 원격저장소의 내용을 로컬저장소에 끌어오는 것
- Push : Commit한 내용을 원격 저장소에 업로드

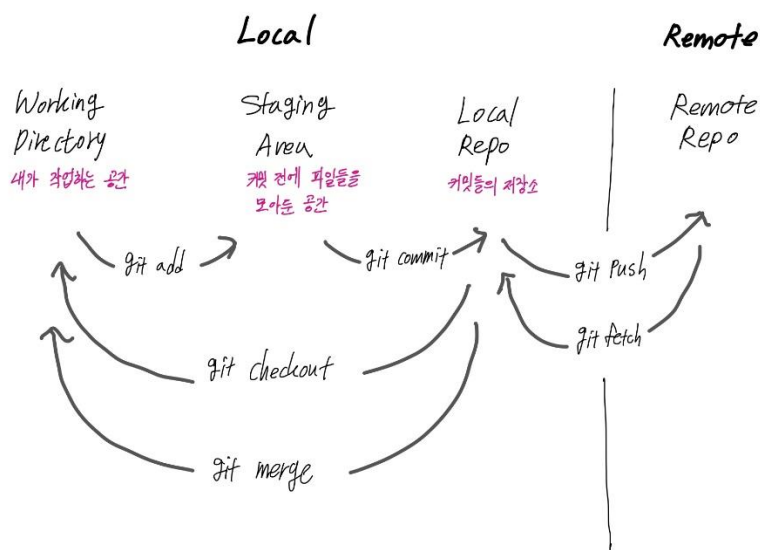
Github 기본 프로세스

Github의 프로세스는 다음과 같다.

자신의 수정 내역을 원격 저장소에 내보내려면 `git add -> git commit -> git push`의 과정을 거쳐야함

수정 내역을 받아올 때는 git fetch 수행

이 과정에서, 내가 수정한 내역이 원격 저장소에 있는 내역과 다를 수 있기 때문에 git merge를 수행하여 자신의 컴퓨터에 있는 소스코드를 원격지 저장소와 맞추는 것



Git 명령어

git init 깃 초기화, 저장소 생성

git remote add origin [깃허브 원격 저장소(레파지토리)

주소] 레파지토리와 로컬 저장소 연결

git config --global user.name [이름] 사용자 이름 설정

git config --global user.email [이메일주소] 사용자 이메일 설정

git clone [깃허브 원격 저장소(레파지토리) 주소] 레파지토리에서 로컬 폴더로 복제

git pull 레파지토리에서 변경된 사항 로컬 폴더로 복사
git add . 변경사항 스토리지에 올리기
git status 현재 상태 확인 (변경사항, 커밋내용, 현재 위치)
git commit -m [변경사항 메모] 변경사항에 대한 정보 메모
git push 원격 깃허브 저장소로 전송
git merge [브랜치명] 메인브랜치로 해당 브랜치 병합
git branch 현재 깃 공간(로컬)에서의 브랜치 조회
-- git branch -a 로컬, 원격 공간 전체 브랜치 조회
-- git branch -v 브랜치 상세정보, 마지막 커밋메세지 조회
-- git branch [브랜치명] 해당 브랜치명으로 브랜치 생성
-- git branch -d [브랜치명] 해당 브랜치 삭제
git checkout [브랜치명] 해당 브랜치로 이동

commit 내역 삭제하기

1. git log 커밋 내역 조회
2. git reset HEAD~[숫자] 최근 커밋내역에서 숫자 이전의 상태로 돌아감
3. git push -f origin [브랜치명] 원격저장소로 강제 push

pull 로 원격파일 강제로 덮어쓰기

1. git fetch --all git pull 받을 목록을 레파지토리에서 업데이트

2. `git reset --hard HEAD` 원격 저장소의 마지막 커밋 상태로 돌아감(원격저장소에 커밋된 내역 이후의 로컬에서 저장된 모든 커밋내역 삭제)
3. `git pull`

커밋하지않고 pull 받아오기

1. `git stash` 스테이지에 있는 내용과 아직 스테이지에 들어가지 않은 변경사항을 모두 저장
-- `git stash -m '메세지'` 커밋과 같이 변경사항에 메세지를 붙일 수 있음
-- `git stash list` 임시저장된 목록, 인덱스값 확인 가능
2. `git status` 워킹트리가 비어있는지 확인
3. `git pull` 변경사항 받아오기
4. `git stash pop` stash 로 저장한 변경사항 워킹트리에 반영(스테이지상태 미반영, pop 으로 꺼내오는순간 stash 에 저장됐던 내용은 삭제됨)
-- `git stash pop --index` 스테이지 상태까지 같이 반영
-- `git stash pop stash@{인덱스값}` 해당 인덱스값의 stash 로 저장한 변경사항 반영

원격 브랜치 가져오기

1. `git remote update` 원격 저장소 갱신
2. `git branch -a` 원격 저장소 포함 git 에 저장된 브랜치 리스트 조회

3. `git checkout -t origin/브랜치명` 원격 저장소의 브랜치와 동일한 이름과 내용을 가진 로컬 브랜치가 생성되고 해당 브랜치로 이동된다.

`git diff` 는 작업 디렉토리와 스테이징 영역 간의 차이를 표시한다. `git diff --staged` : 스테이징된 변경사항과 마지막 커밋 간의 차이를 표시한다.

`git log --graph` 브랜치와 병합 히스토리를 시각적 그래프 형태로 표시한다. `git log --oneline` : 각 커밋을 한 줄로 간단하게 표시한다. `git log` 를 통해 프로젝트의 변경 이력을 효과적으로 확인할 수 있다.

`git revert` 특정 커밋의 변경사항을 취소하는 새로운 커밋을 생성한다. 협업 시 히스토리를 유지하면서 변경사항을 되돌릴 때 사용된다.

`git cherry-pick` 다른 브랜치의 특정 커밋을 현재 브랜치에 적용한다. 특정 기능이나 버그 수정을 다른 브랜치에서 가져올 때 유용하다.

`git tag -a` 특정 커밋에 태그를 추가한다. 주로 릴리스 버전을 표시할 때 사용되며, 중요한 이정표를 표시하는 데 활용된다.

`git bisect` 는 이진 탐색을 통해 버그가 발생한 커밋을 찾는다.

`git bisect start` : 이진 탐색 시작

`git bisect good`: 정상 작동하는 커밋 지정

`git bisect bad`: 오류가 있는 커밋 지정

git worktree 는 동일한 저장소의 여러 브랜치를 동시에 작업할 수 있게 한다.

git worktree add [경로] [브랜치명] : 새로운 작업 트리 생성 특히 여러 기능을 동시에 개발할 때 유용하다.