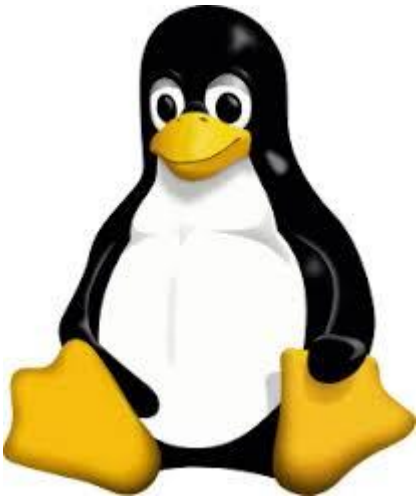
<RO:BIT 리눅스 & git 보고서>

19기 지능 팀 예비단원 박기홍

1. 리눅스란 무엇인가?



[리눅스 아이콘]

1) 리눅스 개념

: 리눅스는 유닉스(Unix) 계열의 오픈 소스 운영체제로, 다양한 장치 및 플랫폼에서 실행될 수 있는 범용 운영체제이다. 리눅스는 서버, 데스크톱, 모바일 장치, 임베디드 시스템 등에서 널리 사용된다. 리눅스는 커널(Kernel)과 그 위에 설치된 여러 가지 응용 프로그램들로 구성되어 있다. 리눅스의 커널은 운영체제의 핵심으로, 하드웨어와 소프트웨어 간의 중간 계층 역할을 하며, 프로세스 관리, 메모리 관리, 파일 시스템 관리, 네트워크 통신 등 시스템 자원을 관리하는 역할을 담당한다.

2) 리눅스 특징

A. 오픈 소스: 리눅스는 자유 소프트웨어 재단의 GPL 라이선스에 따라 누구나 자유롭게 소스 코드를 열람하고 수정할

수 있으며, 배포도 가능하다.

B. 안정성: 리눅스는 매우 안정적이고, 높은 보안성을 제공하며, 주로 서버 환경에서 널리 사용된다.

C. 다양한 배포판: 리눅스는 다양한 배포판(Distro)이 존재하며, 대표적으로 우분투(Ubuntu), 데비안(Debian), 페도라(Fedora), CentOS 등이 있다.

D. 명령줄 인터페이스(CLI): 리눅스는 주로 터미널을 통해 명령어 기반으로 작업을 수행하며, GUI도 사용할 수 있지만, 많은 작업을 효율적으로 명령어로 처리할 수 있다.

3) 리눅스 창시자

: 리눅스는 1991년 리누스 토르발스에 의해 개발되었다. 당시 핀란드의 대학생이었던 토르발스는 기존 유닉스 운영체제를 대체할 무료 운영체제를 만들기 위해 리눅스를 개발하게 되었다. 리누스 토르발스는 리눅스의 소스 코드를 공개하여 많은 개발자들이 이를 개선하고 확장할 수 있도록 했다. 리눅스 커널은 전 세계 개발자 커뮤니티에 의해 유지 관리 및 개선되고 있으며, 특히 서버와 클라우드 인프라에서 중요한 역할을 하고 있다. 리눅스는 GNU 프로젝트의 소프트웨어와 함께 제공되며, 이로 인해 리눅스는 GNU/Linux라고 불리기도 한다.

2. 우분투 보고서

1) 우분투란 무엇인가?

Ubuntu는 **리눅스(Linux) 운영체제** 중 하나로, 데스크탑, 서버, 클라우드, IoT 기기 등 다양한 환경에서 사용할 수 있는 범용적인 운

영체제이다. Canonical이라는 회사에 의해 개발 및 관리되며, 오픈 소스 소프트웨어로 누구나 자유롭게 사용하고 배포할 수 있다. Ubuntu는 유닉스(Unix) 기반의 운영체제이므로 안정성과 보안성이 뛰어나며, 사용자 친화적인 환경을 제공하는 것이 큰 장점이다. Ubuntu는 아프리카 철학에서 유래한 "우분투(Ubuntu)"에서 이름을 가져왔으며, 이는 "인간미", "나눔과 상호 협력"이라는 의미를 가지고 있다. 이러한 철학이 반영된 Ubuntu는 전 세계의 개발자와 사용자 커뮤니티가 협력하여 발전해오고 있다.

2) Ubuntu의 주요 특징

A. **오픈 소스(Open Source).**

Ubuntu는 완전히 오픈 소스 소프트웨어로 제공되며, 누구나 소스 코드를 자유롭게 열람하고 수정할 수 있다. 이는 커뮤니티와 개발자들이 운영체제의 발전에 기여할 수 있게 하며, 보안 문제를 빠르게 해결할 수 있게 만든다.

B. **사용자 친화적인 인터페이스.**

Ubuntu는 리눅스 배포판 중에서도 가장 사용자 친화적인 인터페이스를 제공한다. 그래서 Linux를 처음 접하는 사용자에게도 쉽게 접근할 수 있다. **GNOME** 데스크탑 환경을 기본으로 제공하며, 다른 데스크탑 환경으로도 쉽게 전환할 수 있다.

C. **안정성 및 보안.**

리눅스 기반의 Ubuntu는 매우 안정적이고 보안성이 뛰어나다. 서버 운영체제로 널리 사용되는 이유도 이러한 안정성과 보안성 때문이다. 기본적으로 사용자 계정에서 관리자 권한이 필요할 때만 루트(root) 계정으로 전환하도록 되어 있어, 악성 소프트웨어나 해킹으로부터 시스템을 보호할 수 있다.

D. **다양한 배포판 (Flavors).**

Ubuntu는 다양한 환경에 맞춰 여러 배포판을 제공한다. 데스크탑용 **Ubuntu Desktop**, 서버용 **Ubuntu Server**, 클라우드 환경에서 사용되는 **Ubuntu Cloud** 등이 있다. 또한, 가벼운 환경을 제공하는 **Xubuntu**, **Lubuntu** 같은 배포판도 있다.

E. **LTS (Long Term Support) 버전.**

Ubuntu는 매 2년마다 **LTS(Long Term Support)** 버전을 출시 하고 있다. 이 버전은 5년간 안정적인 보안 업데이트 및 기술 지원을 제공한다. 이를 통해 기업이나 서버 환경에서 안정적으로 Ubuntu를 사용할 수 있다.

F. **소프트웨어 저장소 및 패키지 관리.**

Ubuntu는 **APT**(Advanced Package Tool)라는 패키지 관리 시스템을 사용해 소프트웨어 설치와 관리를 쉽게 할 수 있다. 사용자는 apt 명령어를 통해 수천 가지의 소프트웨

어를 손쉽게 설치하고 관리할 수 있다.

Snap 패키지 시스템도 지원하며, 이를 통해 다양한 앱과 서비스를 쉽게 설치할 수 있다.

G. **광범위한 하드웨어 지원.**

Ubuntu는 다양한 하드웨어 플랫폼에서 동작할 수 있도록 설계되었다. 데스크탑 컴퓨터, 노트북, 서버, 라즈베리 파이와 같은 임베디드 장치에서도 쉽게 설치 및 운영할 수 있다.

H. **강력한 커뮤니티 지원.**

Ubuntu는 전 세계의 개발자와 사용자로 이루어진 강력한 커뮤니티가 있다. 문제가 발생했을 때 도움을 받을 수 있는 포럼, 위키, IRC 채널 등이 활성화되어 있으며, 커뮤니티에서 제공하는 다양한 튜토리얼과 가이드를 통해 Ubuntu를 쉽게 익힐 수 있다.

3. git 보고서

[git 아이콘]

1) git 개념

: git은 소스 코드 관리 및 버전 관리를 위한 분산형 버전 관리 시스템(DVCS, Distributed Version Control System)이다. Git은 주로 소프트웨어 개발에서 사용되며, 여러 개발자가 동시에 하나의 프로젝트에서 작업할 수 있도록 도와준다. Git은 로컬에서 모든 변경 이력을 관리할 수 있으며, 중앙 서버 없이도 독립적인 버전 관리를 수행할 수 있다.

2) git의 주요 특징

A. 분산형 버전 관리: Git은 중앙 서버에 의존하지 않고, 각 사용자가 로컬에 전체 프로젝트의 히스토리를 복사한다. 이를 통해 중앙 서버가 다운되거나 네트워크가 불안정해도 작업을 지속할 수 있다.

B. 브랜칭(Branching): Git의 브랜칭 모델은 매우 가볍고 빠르다. 다양한 브랜치를 생성하여 독립적인 기능을 개발하거나 실험한 후, 원하는 시점에 다시 병합할 수 있다. 브랜치 모델은 협업과 코드 관리를 더 쉽게 만든다.

C. 빠른 성능: Git은 효율적인 데이터 구조를 사용하여 로컬 저장소에서 작업하는 속도가 빠르다. 이는 개발자가 빠르게 커밋, 병합, 브랜치 전환 등의 작업을 수행할 수 있게 한다.

D. 강력한 병합 기능: Git은 여러 개발자의 변경 사항을 효율적으로 병합하는 데 매우 뛰어나다. 복잡한 충돌이 발생해도 이를 해결할 수 있는 도구와 방법을 제공한다.

E. 비용 효율성: Git은 무료로 사용할 수 있으며, 오픈 소스로 제공되기 때문에 누구나 수정하고 배포할 수 있다.

3) Git의 주요 개념

A. Repository(저장소): 프로젝트의 모든 파일과 변경 이력이 저장된 장소이다.

B. Commit(커밋): 파일의 변경 사항을 기록한 스냅샷입니다. Git은 각 커밋을 통해 프로젝트의 특정 시점으로 되돌아갈 수 있게 한다.

C. Branch(브랜치): 개발의 독립적인 라인이다. 여러 브랜치를 만들어 다양한 기능을 독립적으로 개발할 수 있다.

D. Merge(병합): 서로 다른 브랜치에서 작업한 내용을 하나의 브랜치로 합치는 작업이다.

E. Clone(복제): 원격 저장소의 전체 내용을 로컬로 복사하는 작업이다.

4) Git 설치 방법

A. 먼저 패키지 목록을 업데이트 한다.

```
$ sudo apt update
```

B. Git을 설치한다.

```
$ sudo apt install git
```

C. Git이 정상적으로 설치 되었는지 확인한다.

```
$ git --version
```

4. Github 관련 개념

[Github 아이콘]

1) Github 개념

GitHub는 Git을 기반으로 한 웹 기반의 버전 관리 및 협업 플랫폼이다. 주로 소프트웨어 개발자들이 코드를 관리하고 협업하는 데 사용된다. 다양한 프로젝트 관리에도 활용되고 있다. GitHub는 Git을 사용하여 소스 코드 관리 및 버전 관리를 제공하며, 추가적으로 프로젝트 관리와 협업을 위한 다양한 기능을 제공한다.

2) Github의 주요 기능과 특징

A. Git과의 통합.

: Git은 분산 버전 관리 시스템으로, 여러 개발자들이 동시에 코드를 작성하고 관리할 수 있게 한다. GitHub는 Git을 바탕으로 온라인에서 코드 저장소를 호스팅하고, 다른 개발자들과 쉽게 협력할 수 있도록 도와준준다. GitHub를

사용하면 개발자는 원격 저장소를 생성하고, 자신의 로컬 저장소와 동기화하여 버전 관리가 가능핟하다.

B. 공개 및 비공개 저장소.

GitHub는 사용자가 공개 저장소와 비공개 저장소를 만들 수 있게 해준다.

- 공개 저장소: 누구나 저장소에 접근하여 소스 코드를 볼 수 있으며, 오픈 소스 프로젝트가 주로 여기에 포함된다.

- 비공개 저장소: 특정 사용자만 접근할 수 있으며, 기업이나 개인 프로젝트에서 주로 사용된다.

C. 협업 도구.

- Pull Request (PR): 사용자가 저장소에 변경 사항을 제안할 때, PR을 통해 검토를 요청할 수 있다. 코드 리뷰를 거친 후 프로젝트 관리자는 변경 사항을 병합할지 결정할 수 있다.

- Issues: 프로젝트에서 발생하는 문제나 새로운 기능 요청을 기록할 수 있으며, 이를 통해 프로젝트 관리와 팀 간의 소통이 원활해진다.

- Projects: 칸반 보드 형식의 프로젝트 관리 도구로, 작업 흐름을 쉽게 시각화하고 관리할 수 있다.

D. 코드 호스팅과 CI/CD 통합

: GitHub는 코드 저장소뿐만 아니라 GitHub Actions를 통해 CI/CD(지속적 통합 및 배포) 프로세스를 자동화할 수 있는 도구도 제공한다. 이를 통해 코드를 푸시할 때마다 자동으로 빌드, 테스트, 배포를 할 수 있다.

E. 오픈 소스 커뮤니티.

: GitHub는 전 세계적으로 가장 큰 오픈 소스 커뮤니티 중 하나로, 수많은 오픈 소스 프로젝트가 호스팅되고 있다. 개발자들은 GitHub를 통해 오픈 소스 프로젝트에 기여하거나, 자신의 프로젝트를 공유할 수 있다.

F. 포크(Fork) 및 클론(Clone)

- 포크(Fork): 다른 사용자의 저장소를 복제하여 자신의 저장소로 가져와, 독립적인 개발을 할 수 있다. 포크한 저장소에서 작업한 후, 원본 저장소에 PR을 통해 변경 사항을 제안할 수 있다.
- 클론(Clone): 로컬로 저장소를 복제하여 Git을 통해 작업을 할 수 있다. 클론한 저장소는 로컬에서 작업한 후 원격 저장소로 푸시할 수 있다.

G. 버전 관리 및 이력 추적.

- 1GitHub는 각 커밋에 대한 상세한 기록을 저장하므로, 코드 변경 이력을 쉽게 추적할 수 있다. 이를 통해 코드의 변경 사항을 분석하고, 이전 버전으로 되돌아갈

수 있다.

3) Github 장점

- 글로벌 협업: 다양한 지역의 개발자들이 협업할 수 있는 강력한 도구이다.
    - 오픈 소스 지원: GitHub를 통해 수많은 오픈 소스 프로젝트가 개발되고 유지된다.
    - 광범위한 통합: Jenkins, Travis CI, Slack 등의 다양한 도구와 쉽게 통합할 수 있다.
    - 비주얼 프로젝트 관리: 이슈 트래킹, 프로젝트 보드 등을 통해 비주얼적으로 프로젝트를 관리할 수 있다.
    - 코드 리뷰 기능: 협업을 강화할 수 있는 코드 리뷰 도구가 내장되어 있어, 팀원들이 코드를 쉽게 검토하고 피드백을 줄 수 있다.

5. GitBucket 관련 개념

[Git Bucket 아이콘]

1) GitBucket 기본 개념

: GitBucket은 GitHub와 유사한 기능을 제공하는 오픈 소스 Git 플랫폼으로, Git 리포지토리 관리와 프로젝트 협업을 위한 웹 인터페이스를 제공한다. GitBucket은 Scala로 작성되었으며, 주

로 자체 서버에 설치해 Git 리포지토리를 관리하고자 하는 조직이나 개인이 사용한다. GitHub와 비슷한 UI와 기능을 제공하므로 사용자는 익숙한 환경에서 작업할 수 있다.

2) GitBucket의 주요 기능.

A. Git 리포지토리d 관리.

GitBucket은 Git을 기반으로 한 리포지토리 관리 기능을 제공한다. 이를 통해 사용자는 Git 저장소를 생성, 관리할 수 있으며, 버전 관리 및 협업이 가능하다.

B. Pull Request.
GitHub와 마찬가지로, GitBucket도 Pull Request 기능을 지원 가능하다. 이를 통해 사용자는 코드 변경 사항을 검토하고 병합할 수 있다.

C. 이슈 관리:
GitBucket에는 프로젝트에. 발생한 문제를 추적하고 해결할 수 있는 이슈 관리 기능이 포함되어 있다. 이를 통해 프로젝트 관리가 용이하며, 이슈를 통해 팀 간의 의사소통을 할 수 있다.

D. 위키(Wiki).

프로젝트에 대한 문서를 작성하고 관리할 수 있는 위키 기능이 제공된다. 이 기능을 통해 프로젝트의 사용 방법, 개발 가이드 등을 문서화할 수 있다.

E. 플러그인 시스템.

GitBucket은 플러그인을 통해 기능을 확장할 수 있는 시스템을 제공한다. 사용자는 플러그인을 통해 원하는 기능을 추가하거나, 기존 기능을 강화할 수 있다.
예를 들어, CI/CD 통합, Slack 알림 등과 같은 기능을 플러그인으로 추가할 수 있다.

F. 사용자 및 권한 관리.

GitBucket은 사용자 관리 기능을 제공하며, 각 프로젝트나 리포지토리에 대한 권한을 세부적으로 설정할 수 있다.
예를 들어, 읽기 전용 권한을 부여하거나 쓰기 권한을 특정 사용자에게만 허용할 수 있다.

G. Web UI를 통한 관리.

GitHub와 유사한 직관적인 웹 기반 UI를 통해 쉽게 리포지토리를 관리할 수 있다. 이 기능은 Git에 익숙하지 않은 사용자에게도 유용하다.

H. LDAP 및 Active Directory 통합.

GitBucket은 LDAP 및 Active Directory와의 통합을 지원하므로, 기업 내 기존 인증 시스템을 사용해 GitBucket에 접근할 수 있다.

I. Docker 지원.

GitBucket은 Docker 이미지를 통해 쉽게 배포할 수 있다. 이를 통해 Docker 환경에서 GitBucket을 설치하고 관리하는 것이 간편해진다.

J. 경량화된 설치:

GitBucket은 매우 가벼운 설치 요구 사항을 가지고 있다. 단일 WAR 파일로 제공되므로, Java Servlet 컨테이너(예: Tomcat, Jetty) 위에서 쉽게 배포할 수 있다.

6. git 명령어들

- git init: .git 하위 디렉토리 생성하기.

  (폴더를 만든 후, 그 안에서 명령을 실행 -> 새로운 git저장소가 생성됨)

  사용법: $ git init

- git clone <https:,, URL>: 기존 소스 코드를 다운로드 / 복제한다.

  사용법: $ git clone /로컬/저장소/경로

- git clone 사용자명@호스트:/원격/저장소/경로: 원격 서버 저장소를 복제하기.

  사용법: $ git clone 사용자명@호스트:/원격/저장소/경로

- git add <파일명>: 커밋에 단일 파일의 변경 사항을 포함한다.

  사용법: $ git add <파일명>

- git add *: 커밋에 단일 파일의 변경 사항을 포함한다.

  사용법: $ git add *

- git add -A: 커밋에 파일의 변경 사항을 한번에 모두 포함하기.

  사용법: $ git add –A

- git commit –m "커밋 메시지": 커밋 생성하기.

  사용법: $ git add <파일명>

- git stash: 작업 중인 변경 사항 임시 저장하기.

  현재 작업 중인 변경 사항을 임시로 저장하고, 작업 디렉토리를 클린 상태로 되돌린다. 나중에 다시 복원할 수 있다.

  사용법: $ git stash

- git stash pop: 저장된 변경 사항 복원하기.

  git stash로 저장한 임시 변경 사항을 다시 디렉토리에 적용한다.

  사용법: $ git stash pop

- git status: 현재 저장소의 상태 확인하기.

  현재 저장소의 상태, 즉 변경된 파일, 추가된 파일, 커밋할 파일 등을 확인할 수 있다.
  사용법: $ git status

- git log: 커밋 로그 확인하기.

  프로젝트의 커밋 이력을 확인한다. 각 커밋의 해시, 작성자, 날짜, 메시지가 출력된다.

  사용법: $ git log

- git diff: 변경 사항 비교하기.

  커밋되지 않은 파일의 변경 사항을 비교해 보여준다.

  사용법: $ git diff

- git reset: 이전 커밋으로 되돌리기.

  지정한 커밋으로 저장소를 되돌리고, 이후의 변경 사항은 모두 삭제 된다.

  사용법: $ git reset --hard <commit_hash>

- git revert: 특정 커밋을 취소하고 새로운 커밋 생성하기.

  지정한 커밋을 되돌리되, 새로운 커밋을 생성하여 되돌린 이력을 남긴다.

사용법: $ git revert <commit_hash>

- git revert: 특정 커밋을 취소하고 새로운 커밋 생성하기.

  지정한 커밋을 되돌리되, 새로운 커밋을 생성하여 되돌린 이력을 남긴다.

  사용법: $ git revert <commit_hash>

- git chmod: 명령어와 다양한 사용 옵션

  chmod 777: 모든 사용자에게 읽기, 쓰기, 실행 권한을 부여한다.

  사용법: $ chmod 777 filename

  chmod 700: 파일 소유자만 읽기, 쓰기, 실행이 가능한다.

  사용법: $ chmod 700 filename

- git nano: 터미널에서 파일 편집하기.

  터미널 기반의 텍스트 편집기 중 하나로, 파일을 빠르게 수정할 수 있다.

  사용법: $ nano filename

- git crontab: 주기적인 작업 예약하기.

  주기적으로 실행해야 하는 명령어를 예약하는데 사용된다.

예를 들어, 매일 자정에 스크립트를 실행하도록 설정할 수 있다.

사용법: $ nano filename

7. linux 명령어

- sudo: 관리자 권한으로 실행하기.

  사용법: $ sudo

  1) 관리자만 읽을 수 있는 파일을 읽는다.
  2) 새로운 프로그램(Ubuntu Linux)을 설치한다.
  3) 새로운 프로그램 설치시 Package Manager를 이용하는 것이 보편적이다.

- pwd: 현재 위치 출력하기.

  사용법: $ pwd

- ls: 현재 디렉토리 내의 파일과 디렉터리 출력하기.

  사용법: $ ls

- cd: 디렉토리 이동하기.

  사용법: $ cd /

  <$ls –al의 상세 내용(권한)>

  1) 접근 권한(읽기/쓰기/실행 가능) 여부
  2) 링크된 파일 갯수

3) 소유자

4) 소유 그룹

5) 파일 크기

6) 만든 날짜

7) 만든 시간

8) 파일 / 디렉토리 이름

- mkdir: 디렉토리 생성하기.

  사용법: $ mkdir [dir_name]

- cp: 파일 또는 디렉토리 복사하기.

  사용법: $ cp [file] [target_dir_name]

- mv: 파일 또는 디렉토리 이동하기.

  사용법: $ mv [file_name_or dir_name] [target_dir_name]

- rm: 파일 또는 디렉토리 삭제하기.

  사용법: $ rm [file_name]

- cat: 파일 내용 확인하기.

  사용법: $ cat [file_name]

- touch: 빈 파일 생성하기.

  사용법: $ touch[file_name]

  빈 파일을 생성하기.

- echo: 문자열 화면 표시하기.

  사용법: $ echo [-neE] [ARGUMENTS]

- ip addr / ifconfig: 정보 확인하기.

  사용법: $ ip addr add [address] dev [interface]

- ss: 네트워크 확인하기.

- nc: 서버의 포트 확인하기.

- which, whereis, locate: 명령어 위치 확인하기.

- tail: 파일의 마지막 부분 확인하기.

- find: 파일이나 디렉토리 찾기.
  - find /path -name filename

주어진 이름의 파일을 찾기.

- find /path -type d

  디렉토리만 찾기.

- find /path -type f -name '*.txt':

  특정 확장자를 가진 파일 찾기.

- find /path -size +10M

  10MB 이상 크기의 파일을 찾기.

- ps: 현재 실행 중인 프로세스 목록과 상태 확인하기.

- grep: 주어진 입력값에서 패턴에 맞는 값 출력하기.
  - rep 'search_term' file.txt

    파일 내에서 특정 문자열을 검색하기.

  - grep -r 'search_term' /path/to/directory/

    디렉토리 내 모든 파일에서 문자열을 검색하기.

  - grep -i 'search_term' file.txt

    대소문자를 무시하고 문자열 검색하기.

  - grep -v 'search_term' file.txt:

    당 문자열이 포함되지 않은 줄을 출력하기.

- kill: 프로세스 종료하기.

- alias: 명령어 별창 만들기.

- vi / vim: 편집기

- chmod: 파일 권한 변경하기.
    - chmod 755 file

      소유자는 읽기/쓰기/실행, 그룹 및 다른 사용자는 읽기/실행하하기.
    - chmod 644 file

      소유자는 읽기/쓰기, 그룹 및 다른 사용자는 읽기만 가능한 상태임.
    - chmod +x file

      파일에 실행 권한 추가하기.
- chown: 파일 소유자 변경하기.
    - chown user file

      파일의 소유자를 변경함.
    - chown user:group file

      파일의 소유자와 그룹을 변경함.

- more: 페이지 단위로 파일 내용 보기(앞으로만).

- less: 페이지 단위로 파일 내용 보기(앞뒤로).

- head: 파일의 첫 몇 줄 표시하기.

- df: 디스크 공간 사용량 표시하기.

- du: 파일 공간 사용량 추정하기.

- top: 실행중인 프로세스 표시하기.

- ps: 현재 실행중인 프로세스 표시하기.

- pkill: 이름으로 프로세스 종료하기

- man: 명령어의 메뉴얼 페이지 표시하기.

- history: 명령어 히스토리 표시하기.

- unalias: 별칭 제거하기.

- wget: 웹에서 파일 다운로드하기.

  tus

- curl: 여러 프로토콜을 사용하여 데이터 전송하기.

- ping: 연결 테스트를 위해 ICMP 에코 요청 보내기.

- ip: IP 라우팅, 장치 및 터널 수정/표시하기.

-

  netstat: 네트워크 연결 및 라우팅 테이블 표시하기.
- ssh: 원격 시스템에 안전하게 로그인하기.

- scp: 호스트 간의 파일을 안전하게 복사하기.

- rsync: 원격 파일 전송 및 동기화하기.
  - rsync -av source/ destination/

    파일을 동기화하면서 변경된 파일만 복사하기.
  - rsync -avz source/ destination/

    압축하여 전송하기.
  - rsync -av --delete source/ destination/

    원본에 없는 파일을 대상에서도 삭제하기.

- tar: 파일 압축하기.

- tar -cvf archive.tar directory/

  디렉토리를 tar형식으로 압축하기.

- tar -xvf archive.tar

  tar 파일 압축 해제하기.

- tar -cvzf archive.tar.gz directory/

  gzip을 사용하여 디렉토리를 압축하기.

- tar -xvzf archive.tar.gz

  gzip으로 압축된 tar 파일 압축 해제하기.

- gzip: gzip 파일 압축 해제하기.

- gunzip: zip 파일 압축하기.

- unzitp: zip 파일 압축 해제하기.

- date: 현재 사용자 이름 표시하기.

- explrorer: 현재 폴더를 window 파일 관리자에서 보기.

  사용법: $ explorer.

- open: 현재 폴더를 mcaOS finder에서 보기.

사용법: $ open.

- code: 현재 폴더를 VS Code 에디터로 열기.

  사용법: $ code.

- uptime: 시스템 가동 시간 및 로드 확인하기.

  시스템이 얼마나 오래 가동되었는지, 그리고 현재의 시스템 로드(활성 프로세스 수)를 보여준다.

  사용법: $ uptime

- htop: 시스템 성능 및 프로세스 실시간 모니터링.

  CPU, 메모리, 프로세스 상태 등 시스템 성능을 실시간으로 모니터링할 수 있는 도구이다.

  사용법: $ htop

- df -h: 디스크 사용량 확인하기

  디스크 공간 사용량을 사람이 읽기 쉬운 형식으로 출력한다.

  사용법: $ df -h

- du -sh: 디렉터리 크기 확인하기.

  지정한 디렉터리의 크기를 사람이 읽기 쉬운 형식으로 출력한다.

사용법: $ du -sh /home/user/

- ln -s: 심볼릭 링크 만들기.

  파일 또는 디렉터리에 대한 심볼릭 링크를 생성한다.

  사용법: $ ln -s /path/to/file link_name

- systemctl: 시스템 서비스 관리.

  시스템 서비스를 시작, 중지, 재시작하거나 상태를 확인하는 데 사용된다. 예를 들어, Apache 서비스를 시작하려면 아래 명령어를 사용한다.

  사용법: $ sudo systemctl start apache2

- ncdu: 디스크 사용량을 인터랙티브하게 분석하기.

  디스크 사용량을 시각적으로 분석하여 가장 많은 공간을 차지하는 파일이나 디렉터리를 쉽게 찾을 수 있다.

  사용법: $ sncdu

- netcat: 네트워크 연결을 읽고 쓰기 위한 유틸리티이다.

  사용법 예시: $ netcat -l -p 1234

- tcpdump: 네트워크 트래픽을 캡처하고 표시한다.

  사용법 예시: $ tcpdump -i eth0

- file: 파일의 종류를 결정한다.

  사용법 예시: $ file ROBIT.jpg

- stat: 파일이나 파일 시스템의 상태를 보여준다.

  사용법 예시: $ stat file.txt

- jobs: 백그라운드 작업의 목록을 보여준다.

  사용법 예시: $ jobs

- script: 터미널 세션의 활동을 기록한다.

  사용법 예시: $ script session.log

  - strace: 시스템 호출과 시그널을 추적한다.

  사용법 예시: $ strace -p 1234

  - ltrace: 라이브러리 호출을 추적한다.

    사용법 예시: $ ltrace -p 1234

  - Iotop: 디스크 I/O 사용량을 모니터링하는 도구이다.

    사용법 예시: $ iotop

- sar: 시스템 활동을 보고하는 도구이다.

  사용법 예시: $ sar -u 1 3