**Linux & git 보고서**

로빛 20기 인턴 김민조

**목차**

1. Linux

2. Linux 종류

3. Git

4. Git 종류

5. 참고문헌

**Linux**

운영체제란?

* CPU, 메모리, 스토리지 같은 시스템의 하드웨어와 리소스를 직접 관리하는 소프트웨어
* 즉, 작업을 수행하는 소프트웨어와 물리적 리소스를 연결

Linux란?

* 오픈소스 기반의 운영체제(OS)
* 서버, 클라우드, 임베디드, 개발 환경 등에 널리 사용됨
* 다중 사용자, 다중 작업을 지원하는 네트워크 운영체제(NOS)

Linux의 작동 원리

* Linux에는 GNU 툴과 같은 여러 공통 핵심 구성 요소들 포함
* 툴을 통해 사용자는 커널에서 제공되는 리소스 관리, 추가 소프트웨어 설치, 성능 및 보안설정 구성 등 여러 작업 수행 가능
* 모든 툴들이 번들로 묶여 기능적인 운영체제 구성

Linux 구성요소 및 기능

1. 커널

* OS의 기본 구성요소
* 하드웨어와 소프트웨어간의 통신 담당
* 시스템 자원을 효율적으로 관리

(2) 쉘

* 사용자와 시스템 간의 상호작용을 위한 명령어 인터페이스
* 사용자가 명령어 입력 & 실행 => 해석 & 실행
* 다양한 종류의 쉘 존재

(3) 파일 시스템

* 파일과 디렉토리를 계층적으로 구성하는 파일 시스템 사용
* “/”를 시작으로 모든 파일 및 디렉토리 조직

(4) 사용자 및 권한관리

* 다중 사용자 시스템으로 여러 사용자가 동시에 시스템 사용 가능
* 각 사용자는 고유한 사용자 ID와 Password를 가지며, 파일 및 리소스에 대한 권한 가짐
* 시스템 보안과 사용자 간의 격리 기능 구현

(5) 패키지 관리 및 네트워킹

* 패키지 관리자를 이용하여 손쉽게 소프트웨어 설치/업데이트/제거 가능
* 인터넷 연결/원격 서버 관리/파일 공유/네트워크 서비스 등 가능

(6) CLI(터미널, 콘솔, …), GUI

* Linux는 CLI를 통해 사용자가 명령어 입력 및 실행
* 동시에 GUI를 통해 직관적인 그래픽 환경 제공
* CLI는 시스템 관리 및 자동화 작업에 유용
* GUI는 이를 더 쉽고 직관적으로 볼 수 있게 만든 인터페이스

Linux의 특징

(1) 오픈소스

* 소스코드 공개 -> 누구나 자유롭게 사용, 수정, 배포 가능
* GNU GPL 라이선스 기반

(2) 멀티유저

* 하나의 시스템에 여러 사용자 동시 접속 가능
* 사용자마다 독립적인 환경 제공

(3) 멀티태스킹

* 여러 작업을 동시에 처리(백그라운드/포그라운드 작업 병행)
* 프로세스간 자원 충돌 방지

(4) 이식성

* 다양한 하드웨어 아키택처에서 구동 가능
* ARM, x86, x64, PowerPC 등 다양한 플랫폼 지원

(5) 모듈화

* 커널 및 시스템 구성 요소가 모듈 단위로 구성
* 필요한 기능만 로드/언로드 가능 -> 커스터마이징 용이

(6) 네트워크 기능 강화

* 강력한 네트워크 기능 내장
* 서버 운영체제로서 강력한 성능 발휘

(7) 보안성

* 사용자 권한 분리, 접근 제어
* SELiux, AppArmor 등의 보안 모듈
* 커널 보안 패치 빠름

Linux의 장점

* 유닉스와 완벽하게 호환 가능
* 공개 운영체제
* Pc용 os보다 안정적
* 무료로 제공
* 하드웨어의 기능을 효율적으로 사용
* 강력한 보안기능
* 인터넷의 모든 기능을 지원
* 개발 환경이 풍부

Linux의 단점

* 공개운영체제이기 때문에 문제발생 시 보상 X
* 한글입출력 하는데 어려움
* 기술지원 부족
* 특정 하드웨어에 대한 자원 부족
* 사용자의 숙련된 기술 요구

**Linux 종류**

Ubuntu(우분투)

(1) 설명

* 주로 데스크탑 사용자와 서버에 사용
* 사용자 친화 인터페이스 & 설치 용이
* 최신 소프트웨어 패키지와 다양한 하드웨어 지원

(2) 패키지 관리 시스템

* APT(Advanced Package Tool)

(3) 기본 데스크탑 환경

* GNOME

(4) 특징

* 쉽게 설치할 수 있는 GUI 프로그램
* 소프트웨어 센터를 통해 손쉽게 프로그램 설치
* LTS 버전이 제공되어 5년간 보안 업데이트 제공
* 커뮤니티 및 상업적 지원

(5) 용도

* 개인 사용자, 개발자, 서버 운영

(6) 버전

* LTS 버전: 5년간 안정적이고 장기적 지원받음

RHEL(레드햇 엔터프라이즈 리눅스)

(1) 설명

* 주로 기업 환경에서 사용되는 리눅스 배포판
* 안정성과 보안, 기술지원 제공 -> 기업용 서버 및 데이터 센터에서 주로 사용

(2) 패키지 관리 시스템

* RPM(Red Hat Package Manager) / YUM(Yellowdog Updater, Modified) / DNF(Dandified YUM)

(3) 기본 데스크탑 환경

* GNOME

(4) 특징

* 상용 소프트웨어로 기업에서 안정적인 시스템 운영을 지원
* 보안 강화, 성능 최적화, 엔터프라이즈 환경에서의 신뢰성이 특징
* 공식적인 상용 지원과 장기적인 업데이트 제공

(5) 용도

* 대규모 엔터프라이즈 환경, 서버 관리

(6) 버전

* 10년 이상의 긴 지원 주기를 제공, 특히 서버 환경에서 인기

CentOS(센트OS)

(1) 설명

* RHEL의 소스코드를 기반으로 한 커뮤니티 지원 리눅스 배포판
* RHEL의 장점인 안정성과 보안기능을 그대로 제공하면서, 상용 지원은 제외된 무료 리눅스

(2) 패키지 관리 시스템

* RPM(Red Hat Package Manager) / YUM(Yellowdog Updater, Modified)

(3) 기본 데스크탑 환경

* GNOME

(4) 특징

* RHEL의 무료버전으로, RHEL의 업데이트와 보안패치 동일하게 적용
* 서버환경에서 많이 사용, 엔터프라이즈급 성능 제공 but 상용지원 X

(5) 용도

* 기업 서버, 개발 및 테스트 서버

(6) 차이점

* CentOS는 RHEL에서 제공하는 상용지원 없이 커뮤니티 중심 무료지원만 제공

Fedora(페도라)

(1) 설명

* 레드햇에서 후원하는 리눅스 배포판으로 최신기술과 혁신적인 기능을 빠르게 통합하려는 목표로 개발
* 주로 최신 소프트웨어를 사용하는 개발자와 실험적인 환경에서 사용

(2) 패키지 관리 시스템

* RPM(Red Hat Package Manager) / DNF(Dandified YUM)

(3) 기본 데스크탑 환경

* GNOME

(4) 특징

* 최신기술과 새로운 소프트웨어를 가장 먼저 적용
* RHEL에서 지원하지 않는 최신 기능들을 실험적으로 제공

(5) 용도

* 개발, 테스트, 혁신적 환경

(6) 버전

* 비교적 빠르게 업데이트되고 새로운 기능을 시험적으로 제공

Debian(데비안)

1. 설명

* 매우 안정적이고 유연한 리눅스 배포판
* 많은 다른 배포판의 기반이 되며 우분투도 데비안을 기반으로 함

(2) 패키지 관리 시스템

* APT(Advanced Package Tool)

(3) 기본 데스크탑 환경

* GNOME

(4) 특징

* 안정성, 보안에 매우 집중
* 커뮤니티 기반으로 관리되고, 다양한 아키텍쳐를 지원
* 무료 소프트웨어만을 포함하며, 자유 소프트웨어 철학을 따름

(5) 용도

* 안정성 요구하는 서버 환경, 개인용 데스크탑

Arch Linux(아치 리눅스)

(1) 설명

* 경량화되고 유연한 리눅스 배포판으로, 단순함을 중요한 원칙으로 삼음
* 사용자가 시스템을 직접 설정 및 구성
* 패키지관리 시스템이 효율적이고 최신 소프트웨어 제공

(2) 패키지 관리 시스템

* Pacman

(3) 기본 데스크탑 환경

* 없음(사용자가 직접 설정)

(4) 특징

* 롤링 릴리스 모델을 채택하여 항상 최신 버전의 소프트웨어 사용
* 시스템을 최소화하여 사용자 맞춤형 환경을 구성 가능
* 커뮤니티 배포판으로 설치 및 관리가 상대적으로 어렵지만 매우 유연

(5) 용도

* 고급 사용자, 개발자, 시스템 맞춤화가 필요한 환경

Slackware(슬랙웨어)

(1) 설명

* 가장 오래된 리눅스 배포판 중 하나로, 간단하고 직관적인 시스템 제공
* 복잡한 설정없이 바로 사용 X => 사용자가 시스템을 더 깊이 이해하고 구성할 수 있도록 함

(2) 패키지 관리 시스템

* Slackpkg

(3) 기본 데스크탑 환경

* KDE(기본 제공)

(4) 특징

* 매우 안정적이고 빠르며, 시스템 경량화 되어있음
* 사용자 정의에 대한 자유도가 높으며, 수동으로 시스템 설정해야 할 때가 많음

(5) 용도

* 고급 사용자, 서버, 시스템 관리

Linux Mint(리눅스 민트)

(1) 설명

* 우분투를 기반으로 한 리눅스 배포판으로, 사용자 친화적인 인터페이스와 더 직관적인 환경 제공
* 기본적으로 Cinnamon, MATE, Xfce와 같은 데스크탑 환경 제공

(2) 패키지 관리 시스템

* APT(Advanced Package Tool)

(3) 기본 데스크탑 환경

* Cinnamon(기본), MATE, Xfce

(4) 특징

* 우분투와 유사하지만 더 사용자 친화적인 데스크탑 환경 제공
* 데스크탑 사용자가 빠르게 리눅스를 시작할 수 있도록 도와줌

(5) 용도

* 개인용 컴퓨터, 데스크탑 사용

openSUSE(오픈수세)

(1) 설명

* SUSE 리눅스의 커뮤니티 버전으로, 안정적이고 보안이 강화된 배포판
* 대기업 서버 환경에서도 사용될 수 있도록 설계

1. 패키지 관리 시스템

* Zypper

(3) 기본 데스크탑 환경

* KDE Plasma, GNOME

(4) 특징

* 매우 강력한 시스템 관리 도구인 YaST를 제공
* 안정성과 보안에 중점을 두고 있으며, 서버와 데스크탑 환경에서 모두 사용 가능

(5) 용도

* 서버, 기업 환경, 개발자

**Git**

Git이란?

* 분산형 버전 관리 시스템의 한 종류이며, 빠른 수행 속도에 중점을 둠

버전 관리란?

* 파일변화를 시간에 따라 기록했다가 나중에 특정시점의 버전을 다시 꺼내올 수 있는 시스템
* 각 파일을 이전 상태로 되돌리기 가능
* 프로젝트를 통째로 이전 상태로 되돌리기 가능
* 시간에 따라 수정 내용 비교 가능
* 누가 문제를 일으켰는지 추적 가능
* 파일을 잃어버리거나 잘못 고쳤을 때도 쉽게 복구 가능

Git의 장점

* 소스코드를 주고받을 필요 없이, 같은 파일을 여러 명이 동시에 작업하는 병렬 개발 가능
* 분산 버전 관리이기 때문에 인터넷이 연결되지 않은 곳에서도 개발을 진행할 수 있고, 중앙 저장소가 사라져도 원상복구 가능
* 팀 프로젝트가 아닌 개인 프로젝트여도 Git을 통해 버전 관리를 하면 체계적인 개발이 가능하고 프로그램이나 패치를 배포하는 과정도 간단해짐

Git 기본용어

(1) Repository

* 스테이지에서 대기하고 있던 파일들을 버전으로 만들어 저장하는 곳
* Git은 원격 저장소와 로컬 저장소 두 종류의 저장소 제공

(2) Remote Repository(원격 저장소)

* 파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 함께 공유하기 위한 저장소

(3) Local Repository(로컬 저장소)

* 내 PC에 파일이 저장되는 개인 전용 저장소
* 저장소를 새로 만들거나 이미 만들어져 있는 원격 저장소를 로컬 저장소로 복사 가능

(4) Working Tree (Working Directory)

* 저장소를 어느 한 시점에서 바라보는 작업자의 현재 시점
* 파일 수정, 저장, 등의 작업을 하는 디렉터리로, 작업 디렉토리라고도 함

(5) SnapShot

* 특정 시점에서 파일, 폴더 또는 워크스페이스의 상태
* SnapShot을 통해 특정 시점에 어떤 파일에 어떤 내용이 기록되어 있었는지, 폴더 구조는 어떠했는지, 어떤 파일이 존재했는지 등 저장소의 모든 정보 확인 가능
* Git에서는 새로운 버전을 기록하기 위한 명령인 commit을 실행하면 SnapShot 저장 가능

(6) Checkout

* 이전 버전 작업을 불러오는 것

(7) Staging Area

* 저장소에 commit하기 전에 commit을 준비하는 위치

(8) Commit

* 현재 변경된 작업 상태를 점검을 마치면 확정하고 저장소에 저장하는 작업

(9) Head

* 현재 작업중인 Branch를 가리킴

(10) Branch

* 가지 또는 분기점을 의미
* 작업을 할 때에 현재 상태를 복사하여 Branch에서 작업을 한 후에 Merge를 하여 작업 수행

(11) Merge

* 다른 Branch의 내용을 현재 Branch로 가져와 합치는 작업

Git 기본 흐름

* 로컬에서 Repository 만듬
* 작업 파일을 생성/수정
* 변경된 내용을 스테이징
* commit으로 저장
* 원격 저장소에 업로드

**Git 종류**

GitHub

1. 설명

* 가장 널리 사용되는 코드 저장소이자 협업 도구
* 오픈소스 프로젝트가 활발하게 진행되며, 개발자 포트폴리오를 관리하는 용도로 많이 활용됨

(2) 특징

* 광범위한 커뮤니티: 수많은 오픈소스 프로젝트가 등록 돼있어 협업기회 多
* GitHub Actions: CI/CD 자동화 도구 제공
* 무료 저장소 지원: public & private repository 무제한 제공
* 쉬운 코드 리뷰: 풀 리퀘스트 기반 협업이 직관적

(3) 단점

* 자체 호스팅 어려움: GitHub Enterprise 사용 시 비용 발생
* 고급 기능 제한: 무료 플랜에서는 일부 기능 제한

GitLab

1. 설명

* 자체적으로 호스팅할 수 있는 기능이 강력
* DevOps 파이프라인을 통합적으로 운영할 수 있음

(2) 특징

* 자체 호스팅 가능: private 서버에서 운영 가능
* GitLab CI/CD 기본 제공: 별도 설정 없이 강력한 CI/CD 파이프라인 구축 가능
* 무료 플랜에서도 강력한 기능 제공: 이슈 관리, 코드 리뷰 기능 포함

(3) 단점

* UI가 다소 복잡: GitHub에 비해 인터페이스가 직관적이지 않음
* 초기 설정 부담: 자체 호스팅 시 관리 부담 증가

Bitbucket

1. 설명

* Atlassian 제품군과 연동이 쉬워 기업 환경에서 유용하게 사용

(2) 특징

* Jira와 완벽한 연동: 이슈 관리와 코드 협업을 한 곳에서 가능
* 무료플랜에서도 강력한 지원: 최대 5명까지 무료로 사용 가능
* Mercurial 지원

(3) 단점

* 대중적이지 않음: GitHub 대비 개발자 커뮤니티 작음
* UI/UX 부족: 초보자가 사용하기 어려운 편

**참고문헌**

* [**Gitlab Github 차이 ? gitlab vs github vs Bitbucket - 도라가이드**](https://dora-guide.com/gitlab-vs-github/)
* [**GitHub vs Gitlab vs Bitbucket**](https://velog.io/@dm911/GitHub-vs-Gitlab-vs-Bitbucket)
* [**[GIT] ⚡️ git 개념 & 원리 (그림으로 알기쉽게 비유 😃)**](https://inpa.tistory.com/entry/GIT-%E2%9A%A1%EF%B8%8F-%EA%B0%9C%EB%85%90-%EC%9B%90%EB%A6%AC-%EC%89%BD%EA%B2%8C%EC%9D%B4%ED%95%B4#git_%EC%9D%B4%EB%9E%80_%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80?)
* [**[Linux] 리눅스 종류 및 차이점**](https://chkkim22.tistory.com/entry/Linux-%EB%A6%AC%EB%88%85%EC%8A%A4-%EC%A2%85%EB%A5%98-%EB%B0%8F-%EC%B0%A8%EC%9D%B4%EC%A0%90)
* [**Linux(리눅스)란? 오픈소스 운영체제의 핵심 구성 요소와 장점**](https://www.redhat.com/ko/topics/linux/what-is-linux)