Linux & git 보고서

인턴 최세영

**목차**

**리눅스**

- 개요

1. 개발

2. 성장

3. 완성형

4. 오픈소스

5. 시스템

- 커널

1. 메모리 관리

2. 프로세스 관리

3. 장치 드라이버

4. 시스템 호출 및 보안

5. 요약

- 명령어

1. 디렉토리, 파일

2. 정보 표시

3. sudo

**GIT**

- 개요

- 구조

1. Working directory

2. Staging area

3. Repository

- 용어

- 장점

- 단점

**리눅스**

개요

개발

리눅스는 컴퓨터 운영체제 중 오픈 소스로 된 운영 체제이다. 리눅스는 1991년 리누스 토르발스에 의해 만들어졌다.

성장

리눅스가 유명해진 이유는 공개 당시의 운영체제 중 하나인 BSD가 소송에 휘말려 이에대한 대체품이 필요했는데 그 대체품으로 리눅스가 각광을 받게 되었다. 게다가 핵심 개발자 위주로 개발되던 기존 오픈 소스 모델들과 달리 누구든 패치노트를 작성할 수 있고 패치가 받아들여지면 해당 작성자가 리눅스 개발의 기여자가 되어 다음 릴리스에 이름이 올라가는 식으로 만들면서 개발자들의 개발욕구를 촉진시킨 것도 리눅스가 유명해진 이유 중 하나이다.

완성형

초기 리눅스는 기능이 완전치 않은 운영체제였다. 그러다 자체 커널 개발에 난항을 겪던 GNU 프로젝트가 리눅스의 커널에 관심을 가져 서로의 유틸리티를 결합하면서 완전한 운영체제가 되었다. 이후 다양한 기업들의 리눅스 개발 지원이 시작되면서 IT세계에서 빠른 영향력을 펼칠 수 있게 되었다.

오픈소스

이렇듯 리눅스가 오픈 소스로 된 운영체제이다 보니 해당 운영체제를 개량된 버전을 운영체제로 배포하는 리눅스 계열 OS들도 생기게 되었고 그로 인해 나온 운영 체제들이 우분투, 레드햇, 안드로이드와 같이 운영체제들이다.

시스템

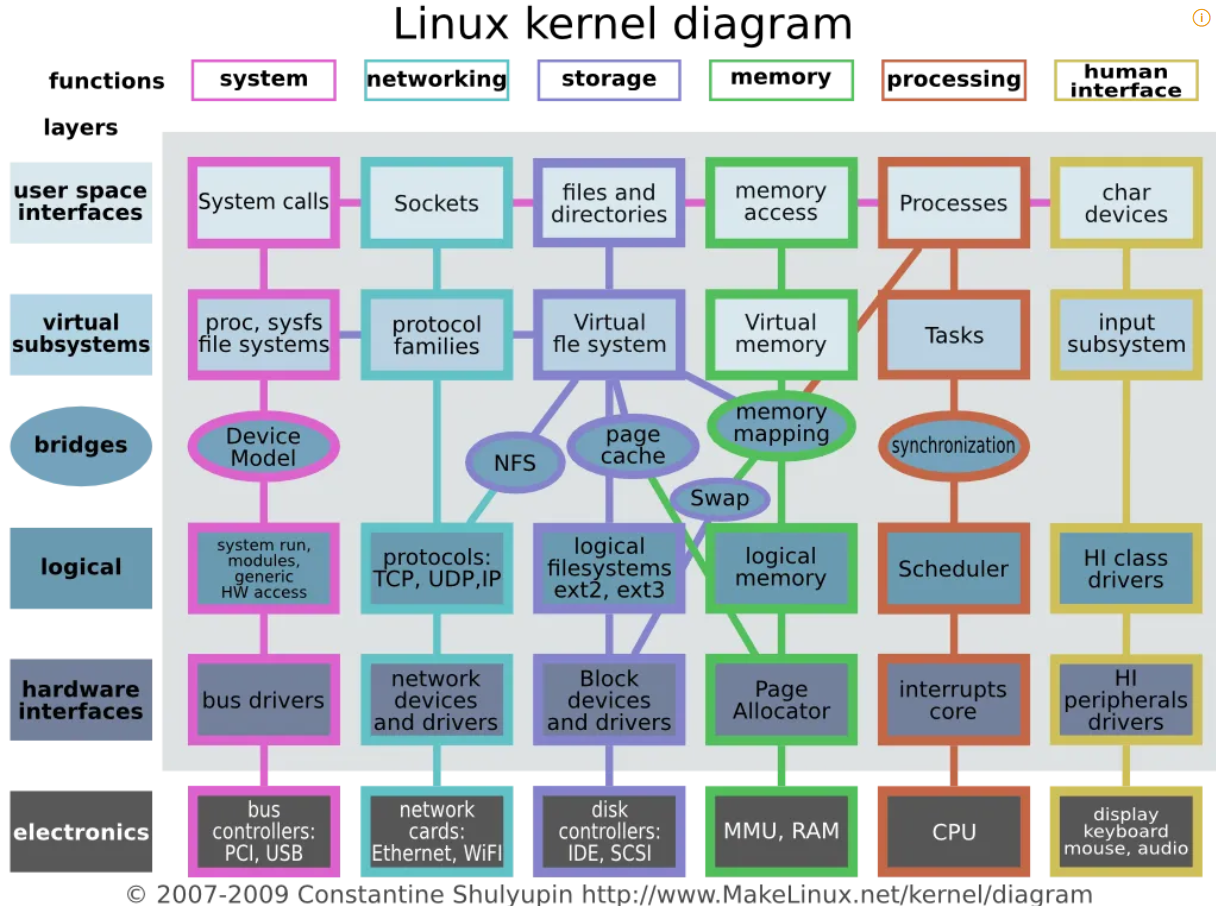
리눅스 시스템은 다음 3가지의 레이어를 가지고 있다.

- **하드웨어**: 시스템의 토대가 되는 물리적인 머신으로 메모리, 중앙 처리장치, 입출력 장치 등으로 구성된다.

- 리눅스 커널: OS의 핵심으로 메모리에 상주하며 CPU에 명령을 내리는 소프트웨어다.

- 사용자 프로세스: 실행 중인 프로그램, 커널이 관리하며 사용자 공간을 구성한다. 단순히 프로세스라 하며 커널이 프로세스와 서버간 통신이 원활할 수 있도록 도와준다.

커널



커널 구조도

커널은 컴퓨터 시스템의 모든 것을 완전히 제어하는 핵심 프로그램이다. 따라서 커널은 하드웨어와 프로세스를 다루면서 물리적, 추상화 자원과 보안을 관리해준다.

- 물리적 자원: CPU, 메모리, 디스크, 네트워크와 같은 하드웨어적인 것에 대한 자원

- 추상화 자원: 물리적 자원을 사용자가 직접 접근하여 다루는 것은 매우 복잡하며 어렵기에 사용자에게 추상적으로 자원을 보여줌으로써 보다 쉽게 접근할 수 있도록 만들어진 자원이다. 예시로 CPU -> Task, 메모리 -> page, segment, 디스크 ->파일 등이 있다.

- 보안: 외부(사용자 포함)에서 하드웨어와 같은 중요한 부분에 접근하여 시스템을 망가뜨리는 것을 막는다.

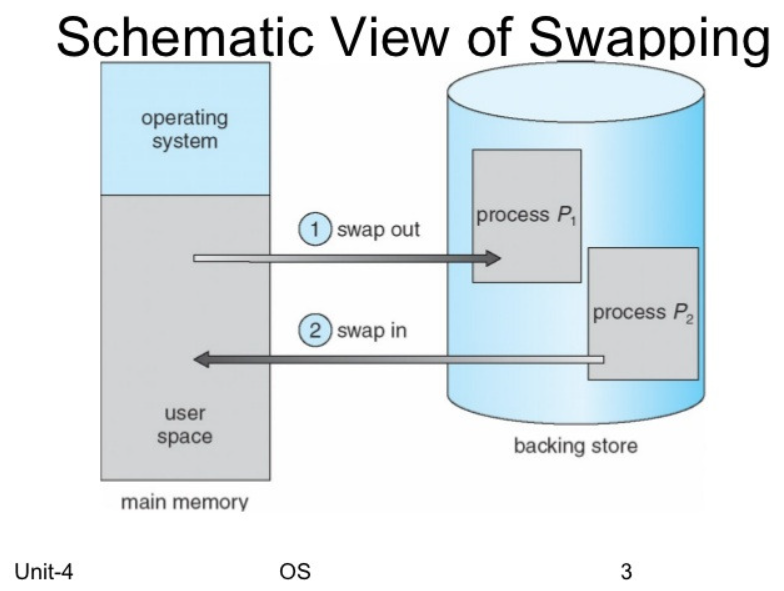
리눅스의 커널은 단일형 커널 구조를 가지고있고 모듈화가 되어 있어 컴파일 후 모듈을 로드할 수 있고 커널에 대한 개발적인 설계를 하여 커널에 포함해야할 기능을 최소화 할 수 있다. 그래서 필요 없는 장치의 커널을 배제할 수 있다. 리눅스 커널의 주 기능에는 메모리 관리, 프로세스 관리, 장치 드라이버, 시스템 호출 및 보안이 있다.

메모리 관리

사용자 프로그램의 요구에 따라 메모리의 영역을 분배하거나 이용이 끝나면 메모리 영역을 회수하는 등 기능을 수행하여 메모리를 관리한다.

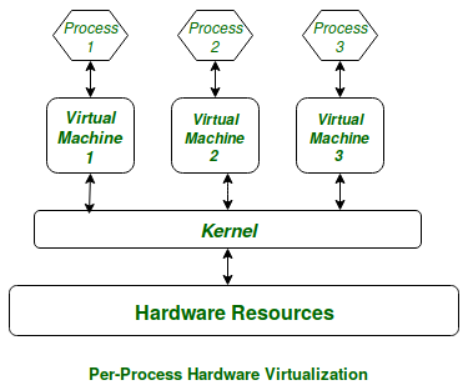
리눅스는 가상 메모리 또한 지원한다. 가상 메모리란 실제로는 존재하지 않지만 존재하는 것처럼 메모리를 이용할 수 있게 해주는 기능이다. 실제 메모리가 아닌 HDD와 같은 보조 기억 장치의 일부를 프로세스가 볼 때 메모리인 것처럼 보이게 해 실제 내장된 메모리보다 더 큰 용량의 메모리 사용을 할 수 있도록 한다.

이렇게 HDD에 할당된 가상 메모리 영역을 스왑이라고 한다. 메모리에 있던 프로그램 전체의 영역을 보조 기억장치로 보내는 것을 스왑 아웃, 그 반대는 스왑 인이라 한다.



프로세스 관리

프로그램을 실행할 때 해당 디렉토리에 있는 프로그램의 파일을 읽어 메모리에 적재한다. 이 프로그램이 메모리에서 실행되는 프로세스가 된다. 프로그램이 종료되면 프로세스또한 삭제된다. 하지만 사용자가 시스템에 로그인하면 100여개의 프로세스가 동시에 실행되는데 CPU는 1개이므로 여러 프로세스를 동시에 이용할 수 없다. 그래서 커널은 프로세스마다 CPU를 이용할 수 있는 시간을 분배하여 많은 프로세스를 동시에 처리하는 것처럼 만든다.



커널은 각 프로세스에 ID를 부여(PID)하고 이를 통해 프로세스를 관리하게 된다.

장치 드라이버

장치 드라이버라는 하드웨어 입출력을 제어하는 소프트웨어를 이용하여 장치를 관리한다.

시스템 호출 및 보안

시스템 호출은 표준 출력이나 파일을 쓰는 write, 읽는 read, 프로세스를 포크하는 기능 등을 가지고 있다. 이를 통해 응용프로그램이 커널에서 제공하는 시스템 호출만을 통해 커널의 기능을 잠시 사용할 수 있도록 도와주며 이러한 기능을 통해 외부로 인한 시스템 오류를 제한할 수 있다.

이렇게 커널은 하드웨어와 사용자간의 접근을 쉽게해 사용성을 높이는 매개역할을 한다.

명령어

Unix및 리눅스에서 사용할 수 있는 터미널 명령으로 대부분 POSIX 표준으로 관리되는 툴이며 대부분 GNU의 Core utilities에 포함되어있다.

디렉토리, 파일

- ls: list의 약자로 특정 디렉토리 내의 내용을 출력한다. 옵션에 따라 출력 내용이 바뀐다.

- pwd: 현재 작업중인 디렉토리의 경로를 출력한다.

- cd: 디렉토리 이동 명령어다.

- mkdir: 디렉토리를 추가한다.

- rmdir: 디렉토리를 삭제한다.

- rm: 파일, 디렉토리를 삭제한다.

- touch: 빈 파일 생성한다.

- cp: 파일을 복사한다.

- mv: 파일을 옮기거나 이름을 수정할 때 사용한다.

- cat: 파일의 내용을 화면에 텍스트로 출력해준다.

정보 표시

- history: 지금까지 사용한 명령어를 출력한다.

- man page: 시스템에 대한 전반적인 정보가 입력된 매뉴얼이다.

- top: 현재 실행되고 있는 프로세스에 대한 정보를 출력하는 명령어다.

- uname: 현재 사용하는 운영체제에 대한 정보를 표시한다.

- find: 옵션에 따라 파일을 찾는다.

- who: 로그인된 사용자의 정보를 출력한다.

- w: 현재 로그인된 사용자가 실행 중인 작업을 볼 수 있다.

- ps: 현재 실행 중인 프로세스를 보여준다.

- df: 연결된 디스크의 용량을 출력한다.

Sudo

Substitute User and Do의 줄임말로 ‘다른 사용자 권한으로 실행한다’는 뜻이다.

각종 명령어 앞에 sudo를 붙이면 해당 명령어는 최고 관리자 권한(root)으로 실행된다. Sudo를 붙인 명령어를 처음으로 사용한다면 사용자 확인으로 현재 로그인 된 사용자 계정의 비밀번호를 올바르게 기입해야 실행된다. 마치 윈도우의 관리자 권한으로 실행과 같은 역할을 한다.

이러한 기능을 가지고 있는 명령어이기에 사용할 때 조심히 다루어야한다.

**Git**

개요

Git은 소스관리 툴로 리눅스를 계발한 리누스 토르발스가 리눅스 커널을 관리하는 툴들이 마음에 안들어 만든 소스 관리 툴이다. Git의 특징은 매우 빠른 속도와 분산형 저장소 지원이다. 여기서 분산형 저장소가 Git의 장점인 쉬운 버전 관리에 중요한 역할을 한다.

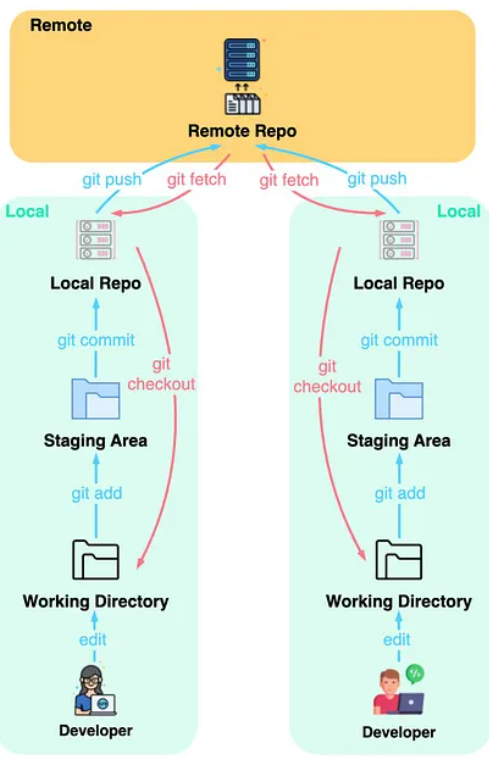
구조

Git은 각 개발자가 중앙 서버에 접속하지 않은 상태에서도 코드 작업을 할 수 있다. 각 개발자가 중앙 서버의 내용을 개인서버에 불러오고 각자의 부분에서 개발을 하다가 개발이 끝나면 Branch라는 것을 만들어 중앙 서버에 맞게 개량 후에 필요에 따라 심사를 하여 중앙 서버에 해당 업데이트를 적용하는 방식이다.

따라서 Git은 Working Directory, Staging Area, Repository로 나뉘어진다.

Working Directory

개인적인 작업이 이루어지는 디렉토리를 뜻한다.

Staging area

개인 서버와 작업폴더 중간에 위치한 중간 저장 영역으로 추출, 변환, 로드 프로세스를 위해 사용된다.

해당 단계에서 파일은 4가지의 단계로 구분된다.

- Untracked: 파일이 Working directory에 있지만 git이 관리를 안 하는 상태

- Staged: 파일이 git add를 통해 Staging area에 반영되어 git이 관리(Tracked)하는 상태

- Unmodified: Tracked하는 파일 중 수정되지 않은 상태

- Modified: Tracked하는 파일 중 수정이 이루어진 상태, Staging Area에 안 올라간 상태

repository

commit을 한 소스코드 파일들부터 프로젝트 변경 이력 또한 저장되어있는 저장소를 의미한다.

종류로는 로컬 저장소, 원격 저장소 2가지가 있다.

용어

- Repository: Stageing area에서 대기하던 파일을 버전으로 만들어 저장하는 저장소로 파일들이 저장되어 있는 물리적인 공간, 로컬 저장소, 원격 저장소 2가지 종류가 있다.

- Working Tree: 현재 파일 수정, 저장 등을 작업하는 디렉토리

- Staging area: Commit하기 전 저장소, Repoistory와 Working directory 사이 공간

- Head: 현재 작업 중인 브랜치

- Snapshot: 특정 시점의 파일, 폴더, 워크스페이스의 상태

- Commit: 작업 후 Repository에 저장하는 작업

- Checkout: 특정 시점이나 Branch 소스 코드로 이동하는 것, 대게 이전 작업을 불러오는데 사용

- Branch: Commit된 소스 코드 타임라인에서 분기점, 분기점에서 작업을 마치고 Merge 할 수 있음

- Merge: Branch간 내용을 병합하는 작업

장점

- 서로의 코드를 검증하는 피어 리뷰에 용이

- 오프라인 작업 가능

- 분산처리로 속도가 빠름

- 일시적인 서버 장애에도 개발을 지속할 수 있음

- 서버의 구성을 유연하게 구서어 가능

- Branch가 매우 쉬움

- Merge시 문제가 적음

- 상업용도로 사용이 가능

- 널리 사용된다.

단점

- 관리 도구가 직관적이지 않음

- 도구를 이해하고 사용하는데 시간이 걸림

- 작업 계층 구조가 자유로워 정확한 설명이 힘듦

- 변동 사항이 자동적으로 알려지지 않아 취합시 충돌이 생길 수 있다.

- 프로젝트 중 일부를 Branch 할 수가 없다

- 개인 서버에서 작업하기 위해서는 해당 프로젝트의 크기 만큼의 저장공간이 필요하다.