LINUX & git 보고서

로빛 20기 인턴 2025406049 현창석

목차

1. LINUX 란?  
  
2. LINUX 명령어 정리

3. git 이란?

4. git 명령어 정리

■LINUX 란?  
  
 LINUX 란 windows와 같이 컴퓨터 운영체제 중 하나이다. 1991년 핀란드 출신 소프트웨어 개발자 리누스 토르발스가 리눅스 커널에 기반을 둔 오픈 소스 유닉스 계열 운영체제인 LINUX를 개발하였다.

다른 운영체제는 하나의 회사나 조직 내에서 개방적이지 않고 폐쇄적으로 개발되는 반면에 LINUX는 인터넷으로 연결되어 전세계 사람들이 개발에 참여할 수 있도록 만들어졌다. 이러한 방식으로 LINUX는 다른 운영체제에 비해 개발된 역사가 길지 않지만 성능과 안정성에 대한 부분에서 다른 운영체제를 능가한다.

개발을 할 때 다른 운영체제를 사용하는 것 보다 LINUX를 사용하는 이유가 있다.

첫 번째는 다중 처리 시스템이다. LINUX는 하나의 시스템을 다수의 이용자들이 접속하여 사용할 수 있다. 또한 접속한 각 이용자는 다수의 작업을 CPU 자원을 나누어서 응용프로그램을 각각 동작시킬 수 있다.

두 번째는 오픈 시스템이다. 운영체제의 핵심이라고 불리우는 커널 이외에도 위에서 언급했듯이 운영체제 내부 모든 응용 프로그램에 대하여 소스가 오픈되어 누구나 개발에 참여할 수 있다.

세 번째는 뛰어난 이식성이다. LINUX는 대부분 어셈블리어와 C언어의 구조를 가지고 있다. 따라서 어셈블리어 구조를 수정하고 C 언어 구조를 다시 컴파일하여 다른 시스템에 쉽게 이식하여 사용할 수 있다.

네 번째는 뛰어난 안정성 및 보안성이다. 앞선 설명에서 LINUX가 운영체제 내부프로그램의 소스가 모두 공개되어 누구나 개발에 참여할 수 있기에 안정성과 보완성에 대한 의문점이 들 수 있다. 하지만 이러한 부분을 미리 해결하기 위해 LINUX를 개발한 전세계 수많은 프로그래머들이 빠르게 오류 수정과 보완 패치를 진행하고 있다.

마지막으로 뛰어난 네트워크 환경 지원이다. 대부분 이용자들이 쓰는 이더넷 뿐만 아니라 SLIP, PPP, ATM 등 다양한 네트워크 환경을 지원하기 때문에 TCP/IP, IPX, AppleTalk 등 네트워크 프로그램의 대부분을 사용할 수 있다.

이러한 장점으로 LINUX 운영체제는 수많은 개발자들에 의해 사랑을 받아왔다

다만 LINUX의 한계도 분명히 존재한다. 첫 번째로 지원 부족이다. LINUX에서 사용하는 응용 프로그램의 대부분은 상업 프로그램이 아니기에 개발자들이 하나하나 기술 지원을 하는 것이 사실상 불가능에 가깝다.

또한 앞서 설명한 LINUX의 장점들 중 다른 운영체제에 비해 뛰어난 이식성과 확장성이 있지만 특정 하드웨어의 경우 설치에 문제가 있기 때문에 모든 플랫폼에 대하여 완벽히 작동하지 않는다는 단점이 존재한다.

두 번째로 사용자의 숙련도이다. 사용자가 중요한 설정을 할 때에 명령어를 하나하나 입력해야 하거나 편집기를 이용해 관련된 환경 설정파일을 수정해야 하기 때문에 사용자에게 일정 그 이상의 지식이 요구된다.

LINUX의 특징에 대해 알아보면 우선 계층적 파일 구조이다. LINUX는 ‘/’를 기준으로 하여 하위 디렉토리에 usr, var, bin 등이 포함되어 있다. 또한 이 하위 디렉토리의 하위 디렉토리에는 local, src 등이 포함되어 있다. 이러한 구조에 대해서 계층적 파일 구조라고 부르고 다른 단어로 트리 Tree 라고도 부른다.

다음으로 모든 장치가 파일로 작동한다는 것이다. 하드디스크, 키보드 또는 프린트, 화면 출력 등 시스템에 설치가 되어있는 하드웨어를 보통 장치라고 부른다.

LINUX는 이러한 장치들을 모두 파일화하여 특정 하드웨어가 가진 장치 파일에 명령을 내림으로써 해당 하드웨어에게 명령을 수행한다.

LINUX는 가상 메모리를 사용한다. 보통 가상 메모리란 하드디스크의 일부를 사용해 메모리로 기능하게 만드는 것을 뜻한다. 여기서 하드디스크는 데이터가 저장되는 공간이고 메모리, 다른 단어로 RAM은 컴퓨터가 작업을 수행하는 작업공간이라고 할 수 있다. 프로그램을 실행시킬 때 이 작업 공간으로 이동하여 수행하게 된다. 다만 이때 충분한 메모리의 공간이 없다면 큰 메모리를 사용하는 프로그램을 실행시킬 수 없게 된다. 이러한 상황을 방지하기 위해 사용하는 것이 가상 메모리이고 LINUX에서는 이를 스왑 SWAP이라고 부른다.

LINUX는 동적 라이브러리를 지원한다. 라이브러리란 프로그램에서 특정한 기능 실행시키기 위한 명령어, 다른 단어로 루틴Routine들을 모아 놓은 것이다.

LINUX는 프로그램 개발 시에 라이브러리 중 개발에 필요한 routine들을 모아 연결 Link 시킨다. 이렇게 같은 루틴을 공유하는 것을 공유 라이브러리라고 한다. 이를 실행 파일 내부에 저장하지 않고 프로그램 실행 시에 바로 사용하기 때문에 동적 공유 라이브러리라고 부른다.

LINUX는 가상 콘솔 기능을 지원하는데 가상 콘솔이란 하나의 모니터를 가진 시스템에 여러 개의 가상화면을 사용하는 것이다. LINUX는 보통 6개 정도의 가상 콘솔을 제공한다. 컨트롤 키 + 알트키 + F1~6 단축키를 사용해 총 6개의 가상 화면을 사용할 수 있다.

LINUX는 누구나 다 개발을 할 수 있기 때문에 여러가지의 종류가 존재한다.

먼저 우분투 리눅스이다. 우분투는 로빛 인턴 기간에 사용할 주요 LINUX 운영체제이다. 우분투는 초보자도 쉽게 설치할 수 있도록 간단히 배포를 한다는 특징이 있다. 또한 Unity 환경을 적용해 초보자들이 쉽게 적응할 수 있는 GUI 또한 제공한다. 보완 관련한 설정에 있어서도 전문지식이 없이 충분히 기본적인 보안 레벨을 유지하는 것이 가능하다. 우분투 이외에 대부분의 LINUX 운영체제 프로그램들이 서버용으로 사용되지만 우분투는 개인 사용자 또는 데스크톱 환경에 최적화가 되어 있으므로 사용자의 편의성이 뛰어나다는 특징 및 장점이 있다. 이처럼 LINUX를 처음 사용해보는 초보자들을 위한 운영체제 프로그램인 우분투는 초보자들을 위한 LINUX라는 특징을 가지고 있다. 다만 좋은 점만 있는 것은 아니다. LTS 버전을 사용하지 않는다면 6개월마다 대대적인 업그레이드를 필요로 하는 점과 Unity 환경이기에 GUI 와 관련한 부분에서 사용자마다 선호를 할 수도 그렇지 않을 수도 있다.

두 번째로 페도라 리눅스이다. 페도라 리눅스는 페도라 프로젝트에서 개발한 운영체제이다. 전문가를 위한 LINUX라고 불리우기도 한다. 페도라는 이후에 설명할 레드햇 리눅스에서 개발되는 기술이 가장 먼저 적용되기 때문에 최신의 기술을 먼저 사용해볼 수 있다는 특징이 있다. 또한 개발에 필요한 도구 등을 기본적으로 제공해준다. 이러한 전문가에게 이로운 특징이 있지만 업데이트를 할 시 에러가 발생하는 경우가 잦다는 점과 기본적으로 에러가 많다는 것이 문제점으로 꼽힌다.

세 번째로 칼리 리눅스이다. 칼리는 백트랙과 같이 해킹과 관련한 수많은 도구와 설명서가 존재한다. 따라서 칼리는 해킹, 정보보안을 테스트하기 위한 목적으로 만들어졌으며 누구나 무료로 다운로드하여 사용할 수 있다.

칼리는 지금까지 배포되고 있는 LINUX 운영체제 프로그램들 중 가장 안정성이 뛰어나며 포함하고 있는 패키지의 종류도 굉장히 다양하다는 특징이 있다.

다만 일정이상의 전문지식이 요구되고 널리 사용되는 유명한 소프트웨어와 사용할 수 없다는 점이 아쉬운 점으로 꼽힌다.

마지막으로 레드햇 리눅스이다. 레드햇 리눅스는 미국에 있는 레드햇 기업이 개발한 LINUX 배포판이다. 유료로 배포한다는 단점이 있지만 그만큼 서버의 안정성과 하드웨어에 대해 폭넓게 지원한다는 특징이 있다. 이뿐만이 아니다. 레드햇은 대규모의 데이터베이스 환경에서도 안정적으로 동작하며 지속적 관리, 최신 기술 , 보안 업데이트 등을 지원한다.

■LINUX 명령어 정리

먼저 touch 명령어이다. Touch는 파일의 날짜 및 시간에 대한 정보를 수정하는 명령어이다. ‘touch 옵션 파일명’ 순으로 입력하여 수정할 수 있다. 옵션이 여러개가 있다. –a 는 최근에 사용한 파일의 사용시간만 변경하는 것이고 –t 는 현재 시간 대신 직접 시간을 지정하여 수정한다. –m은 최근에 사용한 파일의 변경 시간만 수정하도록 하는 것이고 –r은 지정 파일의 접근 및 변경시간으로 다른 파일의 시간을 변경할 수 있다. 마지막으로 –c는 현재 시간으로 정보를 수정한다. 만약 어떠한 옵션 없이 사용하게 되다면 선택한 파일의 최근 사용한 시간과 최근에 변경한 시간이 현재 시간으로 바뀌게 된다.

두 번째로 cp 명령어이다. Cp명령어는 파일이나 디렉토리를 복사할 때 사용한다. ‘cp + 옵션 + 원본 파일 + 복사 위치’ 순으로 명령어를 입력하여 실행시킬 수 있다. 옵션 –a는 원본 파일의 속성이나 정보 등을 그대로 유지하며 복사할 수 있도록 해준다. –i는 복사를 할 위치에 같은 파일이 있을 때 덮어쓸 것인지 확인하는 옵션이다. –r은 디렉토리 복사 시에 하위 디렉토리까지 모두 포함하여 복사하게 하고 –f는 i옵션과 다르게 같은 이름의 파일이 존재한다면 자동으로 덮어쓰기를 진행한다. –b 옵션은 덮어쓰기를 할 때 백업 파일이 생성되도록 하는 기능이 있고 –p는 가지고 있는 모든 정보를 복사한다. –u는 복사되는 그 대상이 최신의 파일일 경우에 복사되지 않도록 한다.

세 번째로 mv 명령어이다. mv는 move의 준말의 명령어이다. LINUX에서 파일을 이동시킬 때 사용하는 명령어이다. ‘mv + 옵션 + 대상 파일 + 이동 위치 경로’ 순으로 입력하여 실행시킬 수 있다. –b 옵션은 이동 위치로 파일을 옮길 때 같은 이름의 파일이 존재하면 기존 파일을 백업한 후 복사하는 과정이 거치도록 하고 -i옵션은 이동 위치에 같은 파일이 있을 시 덮어쓸 것인지 확인한다.

-v 옵션은 이동 위치로 이동시키기 전 파일명을 출력시키도록 하고 –f는 같은 이름의 파일이 이동 위치에 있을 때 자동으로 덮어쓰기를 진행하는 기능이 있다.

rm은 remove의 준말로 파일이나 디렉토리를 삭제할 때 사용하는 명령이다.

‘rm + 옵션 + 삭제 파일 대상’ 순으로 입력한다. –i옵션은 삭제할 각 파일에 대해 삭제 확인 여부를 묻는 옵션이고 –f는 어떠한 메시지 없이 강제적으로 대상 파일을 삭제한다. –r 옵션은 하위 디렉토리를 포함한 모든 파일을 삭제한다.

마지막으로 -v옵션은 삭제된 파일의 이름을 출력한다.

Cat은 파일을 연결하고 표시하는 기능을 하는 명령어이다. ‘cat + 옵션 + 대상 파일’ 순으로 입력하여 명령을 실행시킨다. 여기서 –n 옵션은 행 번호를 화면의 왼쪽에 나타내는 기능을 제공하고 –E 옵션은 각 줄 맨 끝에 $ 문자를 붙여서 출력한다. –A 옵션은 화면을 제어하는 문자 ($ 등) 을 표시한다.

head 명령어는 파일 또는 데이터의 앞 줄을 출력하는 기능을 한다. ‘head + 옵션 + 대상 파일’ 순으로 입력하여 기능을 작동시킨다. 사용을 하면 10줄을 기본적으로 출력해준다. –n num 옵션은 입력한 num 만큼 출력을 하고 –c num 옵션은 num 바이트만큼 출력한다. –q 옵션은 여러 개의 파일을 출력할 때에 제목을 붙이지 않고 출력한다.

tail명령어는 head 와 정반대로 문서 파일이나 지정한 데이터의 마지막 몇 줄을 출력하는 데 사용한다. ‘tail + 대상 파일’ 을 입력하여 명령을 실행시킨다. 로그파일과 같이 실시간으로 바뀌는 파일의 내용을 볼 때 주로 사용한다.

More 명령어는 텍스트 파일에 있는 내용을 한번에 보기 위하여 사용한다. 행의 개수나 또는 자료의 개수가 너무 많아 화면에 모두 출력되지 못하는 경우에 사용한다.

■ git 이란?

LINUX에 대해 위에서 설명한 것과 같이 LINUX는 모두에게 소스가 공개된 운영체제 프로그램이었다. 과거 git이 없을 때에는 여러 명이 협업을 하여 개발을 하는 상황에서 파일을 각자 따로 만든 후 합치는 과정을 거쳐야 했다. 이러한 비효율적인 과정은 이들로 하여금 엄청난 시간을 들게 만들었고 이를 해결하기 위하여 분산 버전 관리 시스템 (Distributed Version Control Systems)가 등장하였다.

이러한 시스템을 가진 처음 공개된 프로그램은 Bitkeeper라는 프로그램이었다.

하지만 Bitkeeper 프로그램 개발자들이 이를 유료화하여 상품으로 파려는 시도를 하게 되자 LINUX의 개발자인 리누스 토르발스는 Bitkeeper와 같이 분산 버전 관리 시스템인 Git을 만들어 무료로 배포하였다.

위에서 언급한 VCS, 즉 버전 관리 시스템은 버전을 효율적으로 관리하여 시간과 용량을 효율적으로 사용하게 해주는 시스템이다. 이 버전 관리 시스템의 종류는 3가지가 있다. 첫 번째로 로컬 VCS, 두 번째로 중앙집중식 버전 관리 시스템, 세 번째로 분산 버전 관리 시스템이 있다. VCS를 사용하지 않는다면 새로운 버전을 만들 때 전체 파일을 복사해야 했지만 VCS를 이용하면 부분적으로 변경하여 용량을 최대로 아낄 수 있다는 장점이 있다. 하지만 단점도 엄연히 존재한다. 로컬 VCS는 백업을 해놓지 않을 시 바이러스에 굉장히 약하다는 점과 협업이 힘들다는 점이 있고 중앙집중식 버전 관리 시스템은 협업이 힘들다는 위의 단점을 보완할 수 있지만 바이러스와 오류에 취약하다는 점이 단점으로 꼽힌다.

분산 버전 관리 시스템은 모든 버전에 대해 서버의 백업본을 각 로컬에서 가지고 있기 때문에 서버에 오류가 발생하거나 또는 오프라인 환경에서도 로컬에서 버전 관리를 할 수 있다.. 또한 서버 내의 내용이 바이러스 침입으로 인해 삭제되어도 로컬에서 다시 업로드를 하면 복구가 가능하다는 장점이 있다.

Git은 세 가지의 구성요소로 이루어진다.. 첫 번째로 Working Directory이다. Working Directory , 즉 작업 디렉토리는 프로젝트의 파일과 폴더가 포함되어 있는 로컬 내의 디렉토리를 뜻한다. 개발자들은 이 작업 디렉토리에서 파일을 생성, 수정 또는 삭제한 후, 이를 Git에 추가하여 필요시에 사용할 수 있다.

두 번째로 Staging Area 즉 스테이징 영역은 변경된 파일들을 추적하고 다음 커밋에서 스냅샷 기능을 이용해 저장할 준비가 된 파일을 표시하는 중간단계를 뜻한다.

세 번째로 Repository 즉 저장소는 Git이 추적하고 관리하는 프로젝트의 모든 변경 내용을 포함하는 데이터베이스라고 불리운다. 모든 커밋 및 브랜치 또는 태그 등의 정보가 모두 저장되어 있고, Git의 모든 기능은 이 저장소 내에서 작동된다.

Working Directory  ->  Staging Area  -> (commit)->  Repository  
순으로 동작한다.

워킹디렉토리의 파일 중에서 변동사항을 추적할 내용을 지정하여서 스테이징 에어리어에 할당한다. 그 후 이 추적 내용은 스테이징 에어리어에서 커밋을 통해 저장소에 저장된다. 커밋이 이루어지면 스냅샷을 생성하고 이러한 내용을 저장소에서 관리한다. 이러한 방식으로 git은 작동한다.

그렇다면 깃허브는 무엇일까. 앞서 이야기했던 Repository에 저장된 커밋의 내용을 Push를 하게 되면 로컬이 아닌 원격 저장소로 올라가게 된다. 이를 바로 github라고 부른다.

■ git 명령어 정리

Git init 명령어는 현재 작업 중에 있는 디렉터리를 git 레포지토리로 초기화한다. 이 명령을 실행시키면 숨겨진 폴더 생성되고 이 폴더에 git이 모든 버전의 이력을 저장한다. 이렇게 한 번 init 명령어로 초기화한 폴더에서 git 명령어를 사용할 수 있다. 새로운 프로젝트를 시작할 때 한 번 실행시키면 된다.

Git add + 파일명 명령어는 변경 사항을 Stage에 추가하는 역할을 한다. 다음에 커밋에 포함시킬 파일을 올려둔다. ‘ Git add . ‘ 을 입력하면 현재 디렉터리에 포함된 모든 변경 파일을 한꺼번에 stage에 올린다. Git add –A 또는 git add –all 명령어를 입력하면 git add . 과 비슷하게 프로젝트 내 모든 변경사항에 대하여 Stage에 추가한다.

작업 중 현재 git 의 상태나 커밋의 이력을 확인하고 싶다면 git status 또는 git log를 입력하면 된다.

Git은 브랜치라는 기능이 있다. 이는 독립된 작업의 흐름을 만들기 위해 사용되는데 새로운 기능을 개발할 때 기존 코드에서 브랜치를 만들어서 작업할 수 있고 이렇게 여러 개의 작업이 동시에 진행되어도 작업이 끝나면 다시 예전 브랜치에 합할 수 있기 때문에 충돌 위험의 가능성이 낮다.

‘Git branch 브랜치 이름’ 명령어를 입력하면 새로운 브랜치를 만들어낸다. 그 후 ‘git branch 브랜치 이름’을 입력하면 작업 중인 커밋이 브랜치로 이동한다. 이렇게 하면 해당 브랜치에서 작업을 할 수 있다.

브랜치를 더 이상 사용하지 않는다면 ‘git branch –d 브랜치이름’ 명령어를 입력해서 브랜치를 삭제할 수 있다. –d 옵션은 이미 병합되어 있는 브랜치만 삭제하기 때문에 해당 브랜치가 병합되지 않은 상태라면 git이 삭제를 실행하지 않는다. 이때 강제로 삭제하고 싶다면 –D룰 통해 완벽하게 삭제할 수 있다. 브랜치를 삭제한다고 하더라도 그 브랜치의 커밋 내용이 다른 브랜치에 합쳐져 있다면 그 이력을 지워지지 않는다. 원격 저장소에 있는 브랜치를 지우고 싶다면 git push origin -–delete 브랜드이름을 명령어로 입력하여 원격 브랜치를 제거할 수 있다.

감사합니다