# **소개**

## **유닉스(Unix) 소개**

1970년 AT&T 벨연구소 켄 톰슨(Ken Tompshon)이 Unix를 혼자 개발하였다. 그래서 별 기능은 없지만, 작고 가볍고 단순하지만 융통성있는 구조로 개발되었다.

미국에는 독과점 규제법이 존재하였고, 거대 통신기업인 AT&T가 이제 막 성장하는 컴퓨터 산업에 진출하지 못하도록 1970년부터 AT&T는 컴퓨터 사업을 할 수 없었다.

그래서 유닉스는 개발되자마자 무료로 소스코드가 공개되었고, 많은 개발자들이 사용하였다.

70년대 대표적인 PC였던 8bit 애플2는 개발자용으로는 많이 부족했다. 혼자 게임을 개발하려고 만든 작고 가벼운 운영체제, 그렇지만 그 당시 PC운영체제보다 훨씬 뛰어나고 융통성 있었던 유닉스가 70년대부터 대표적인 개발자용 운영체제가 되었다.

1970년에 AT&T 벨연구소에서 유닉스가 처음 개발되었을 때, 그 운영체제의 이름 유닉스는 고유명사였지만 구 후 여러 회사에서 각자 자신의 유닉스를 개발하기 시작하여 여러 종류의 유닉스가 만들어졌기 때문에, 유닉스는 보통 명사가 되었다. 애플 MAC OSX도 유닉스의 한 종류이다.

## **리눅스(Linux) 소개**

80년대 대표적인 PC는 IBM PC 호환 컴퓨터이다. 16bit 8086 CPU의 IBM PC XT는 개발자용 컴퓨터가 될만한 성능이어지만, MS-DOS 운영체제가 너무 사양이 낮았다. 80386 CPU가 나오자 PC는 개발자용 컴퓨터가 되기에 충분한 성능이 되었고, 80386 CPU용 유료 유닉스들도 시장에 나왔다. 그러나 개인용으로 사기에는 너무 비쌌다.

1991년 당시 핀란드의 리너스 토발즈(Linus Tovalds)가 취미로 386 CPU 유닉스 커널을 혼자 만들어 무료로 공개했다. 이 커널의 이름이 리눅스(Linux) 이다.

당시 토발즈는 16bit 8086 CPU 용을 구현된 아주 작은 교육용 운영체제의 내부 구현에 대한 책을 읽고, 그 책의 소스 코드를 시작으로 80386 CPU 용 리눅스 커널 최초 버전을 몇 달만에 완성하여 공개했다. 그 이후로 지금까지 계속 리눅스 커널을 업데이트 하고있다.

리눅스가 당시 다른 유닉스 운영체제들보다 뛰어났던 것은 아니다.

하지만 소스코드까지 공개된 무료 유닉스이고, 386 PC에서 실행 가능한 유닉스 였다는 점 때문에, 발표되자마자 폭발적인 인기를 끌었다. 386 PC를 가진 수많은 개발자들이 자신의 컴퓨터에 리눅스를 설치하여 사용했다.

리눅스 이전부터 오픈 소스 그룹이 있었고, 그 중심에 리처드 스톨만이 있었다. 리처드 스톨만은 GNU 프로젝트(https://www.gnu.org/)를 이끌고 있었는데 이 프로젝트는 무료인 완전한 유닉스 호환 컴퓨팅 시스템을 만드는 것이 목표였다.

(컴퓨터 환경 = 운영체제 커널 + 개발도구 + 기본 유틸리티 + 기본 앱 + 생산성 앱)

이 컴퓨터 환경 목표 중에서, 운영체제 커널 개발이 특히 부진했다. 매우 강력한 커널을 만들려다 보니 진도가 느렸기 때문이다. 그래서 리눅스 커널이 발표되자 GNU 프로젝트는 바로 리눅스 커널을 채택했다.

즉, 리너스 토발즈가 만든 것은 지금도 만들고 있는 리눅스 커널이고, 개발 도구 + 기본 유틸리티 + 기본 앱 + 생산성 앱 등은, GNU 프로젝트에 참여하는 많은 오픈소스 개발자들이 만들고 있다.

리너스 토발즈도 GNU 개발도구를 사용해서 리눅스 커널을 개발했다. (gcc)

## **WSL2**

(windows Subsystem for Linux, Version 2)

서버 운영체제의 대세가 리눅스인 흐름에서 밀리지 않기 위해 MS는 window10 최신 업데이트에 리눅스 커널을 포함시켰다.

즉 window10 운영체제에는 windows커널과 리눅스 커널 둘다 들어있다.

## **GNU 핵심 유틸리티**

### **GNU 핵심 유틸리티**

- 파일 복사, 삭제, 압축 등 아주 기본적인 작업을 위한 도구이다.

- 유닉스의 기본 명령과 사용 방법이 같다.

- GNU 프로젝트에서 만들어졌다.

- 리눅스 뿐만 여러 운영체제에 이식되었다.

- Windows 용 Git 클라이언트에도 포함되어 있다.

### **쉘(shell)**

명령 프롬프트 창에서 입력된 명령을 실행하는 프로그램을 쉘(shell)이라고 한다.

windows의 기본 쉘은 cmd.exe 이다.

Linux의 기본 쉘은 bash 이다.

### **Git bash shell**

Git 클라이언트에는 windows 용 bash 쉘과 GNU 핵심 유틸리티들이 포함되어있다.

이 bash 쉘과 GNU 핵심 유틸리티들의 사용법은 리눅스와 같다.

Git bash 쉘에서 유닉스 기본 명령 실습을 할 수 있다.

# **기본 명령어**

## **경로명**

유닉스 명령에 파일 이름이나 디렉토리 이름을 입력할 때, 경로명을 사용할 수 있다.

rm.test1.c : 현재 디렉토리에 있는 test1.c 파일을 삭제한다.

rm src/test1.c : 현재 디렉토리 아래에 src 디렉토리에 있는 test1.c 파일을 삭제한다.

### **디렉토리 구분문자**

Window의 디렉토리 구분 문자는 \ 이고, 유닉스의 디렉토리 구분 문자는 / 이다.

예 :

C:\tmp\test1.c

/tmp/test1.c

### **/ 절대 경로**

경로명이 / 문자로 시작하면 절대 경로이다.

절대 경로는 루트 디렉토리부터 시작하는 경로이다.

rm /tmp/test1.c : 루트 디렉토리 아래 tmp 디렉토리 아래 test1.c 파일을 삭제한다.

### **상대 경로**

경로명이 / 문자로 시작하지 않으면 상대 경로이다.

상대 경로는 현재 디렉토리부터 시작하는 경로이다.

rm tmp/test1.c : 현재 디렉토리 아래 tmp 디렉토리의 test1.c

### **부모 디렉토리**

경로명에서 .. 문자는 바로 위 디렉토리를 의미한다.

rm ../test1.c : 현재 디렉토리 바로 위 디렉토리의 test1.c 파일을 삭제한다.

rm ../src/test1.c : 현재 디렉토리 바로위 디렉토리에 있는 src 디렉토리의 test1.c 파일을 삭제한다.

rm ../../test1.c : 현재 디렉토리 바로 위 디렉토리 바로 위 디렉토리의 test1.c 파일을 삭제한다.

### **현재 디렉토리**

경로명에서 . 문자는 현재 디렉토리를 의미한다.

rm ./test1.c : 현재 디렉토리에 있는 test1.c 파일을 삭제한다.

### **도트 파일**

유닉스에서 설정 파일은 파일 이름이 마침표 문자로 시작하는 것이 관례이다.

예: .bashrc

설정 파일들이 들어있는 디렉토리 이름도 마침표 문자로 시작하는 것이 관례이다.

예: .settings

### **~ 홈 디렉토리**

경로명에서 ~ 문자는 홈 디렉토리를 의미한다.

유닉스에서 홈 디렉토리는 사용자 개인에게 할당된 디렉토리이다.

홈 디렉토리 아래에만 내 파일을 저장할 수 있다.

홈 디렉토리를 벗어나면, 쓰기 권한이 없다.

Windows에서 홈 디렉토리는 C:/User/로그인명 이다.

cd ~ : 홈 디렉토리로 이동한다.

## **디렉토리 명령**

### **pwd**

pwd(print working directory)

현재 내가 어느 디렉토리에 있는지 출력한다.

### **cd**

cd(change directory)

디렉토리를 이동하는 명령

cd .. : 바로 위 부모 디렉토리로 이동한다.

유닉스나 Windows의 경로명에서 .. 은 바로 위 디렉토리를 의미한다.

cd test1 : test1 : test1 이름의 디렉토리로 이동한다.

현재 디렉토리 아래에 test1 이름의 디렉토리가 있어야 한다.

cd ../test1 : 부모 디렉토리 아래의 test1 이름의 디렉토리로 이동한다

부모 디렉토리 아래에 test1 이름의 디렉토리가 있어야 한다.

cd ../test1/test2 : 부모 디렉토리 아래의 test1 이름의 디렉토리 아래의 test2 디렉토리로 이동한다.

cd /test1 : 루트 디렉토리 아래의 test1 디렉토리로 이동한다.

cd ~ : 홈 디렉토리로 이동한다.

cd ~/test1 : 홈 디렉토리 아래 test1 디렉토리로 이동한다.

### **mkdir**

mkdir(make directory)

디렉토리를 생성한다.

mkdir test1 : 현재 디렉토리 아래에 test1 이름의 디렉토리를 만든다.

### **rmdir**

rmdir(remove directory)

디렉토리를 삭제한다.

rmdir test1

현재 디렉토리 아래에 test1 이름의 디렉토리를 삭제한다.

비어있는 디렉토리만 이 명령으로 삭제할 수 있다.

디렉토리에 파일이 들어있으면 삭제는 실패한다.

### **rm -r**

디렉토리와 그 내용을 모두 삭제

rm -r test1 : test1 디렉토리와 그 아래 내용을 모두 삭제한다.

### **mv**

mv(move)

디렉토리의 이름을 변경하거나, 옮긴다.

mv test1 test2 : 현재 디렉토리 아래에 test2 디렉토리가 없다면, test1 디렉토리 이름을 test2로 바꾼다. test2 디렉토리가 있다면 test1 디렉토리를 test2 디렉토리 아래로 옮긴다.

mv ../test1 . : 바로 위 디렉토리의 test1 디렉토리를 현재 디렉토리로 옮긴다.

mv test1 ~ : 현재 디렉토리의 test1 디렉토리를 홈 디렉토리로 옮긴다.

mv test1 ../test2 : 부모 디렉토리 아래에 test2 디렉토리가 없다면, test1 디렉토리를 부모 디렉토리로 옮기고 이름을 test2로 바꾼다. 부모 디렉토리 아래에 test2 디렉토리가 있다면, test1 디렉토리를 부모 디렉토리 아래의 test2 디렉토리 아래로 옮긴다.

## **ls 명령**

ls(list)

### **ls 명령은 다음과 같은 형태로 사용된다.**

ls : 현재 디렉토리의 내용(파일과 서브디렉토리)을 출력한다.

ls 디렉토리 : 디렉토리 아래의 내용을 출력한다.

ls 파일 : 파일에 대한 정보만 출력한다.

ls -d 디렉토리1 : 디렉토리1의 내용이 아니고, 디렉토리1에 대한 정보만 출력한다.

### **예**

ls . : 현재 디렉토리의 내용을 출력한다.

ls test1 : 현재 디렉토리 아래 test1 디렉토리의 내용을 출력한다.

test1이 디렉토리가 아니고 파일이라면, 그 파일에 대한 정보를 출력한다.

ls .. : 부모 디렉토리의 내용을 출력한다.

ls ../test2 : 부모 디렉토리 아래 test2 디렉토리의 내용을 출력한다.

test2가 디렉토리가 아니고 파일이라면, 그 파일에 대한 정보를 출력한다.

## **ls 명령 옵션**

### **ls -l**

l (long)

ls 명령의 -l 옵션

디렉토리의 내용을 자세히 출력한다.

|  |
| --- |
| $ ls -l  total 2  drwxr-xr-x 1 Seunghyeon 197609 0 10월 11 01:32 src/  -rw-r--r-- 1 Seunghyeon 197609 123593 10월 11 01:25 t.txt  -rw-r--r-- 1 Seunghyeon 197609 32 10월 11 02:58 Test.c  drwxr-xr-x 1 Seunghyeon 197609 0 10월 11 01:32 test1/  drwxr-xr-x 1 Seunghyeon 197609 0 10월 11 02:41 test2/ |

### **drwxrwxrwx**

첫문자 ‘d’ = 디렉토리 ‘-‘ = 파일

rwx 읽기 쓰기 실행 권한이 다 있다.

rw- 읽기 쓰기 권한만 있다.

r—읽기 권한만 있다.

--- 아무 권한이 없다

x 대상이 파일인 경우 실행 권한, 대상이 디렉토리인 경우에는 들어갈 수 있는 권한

첫번째 rwx = 파일을 소유한 사람의 권한

두번째 r-x = 파일 소유 그룹에 속한 사람의 권한

세번째 r-x = 소유자도 소유 그룹도 아닌 사람의 권한

### **ls -a**

(all)

-a 옵션이 있으면, 디렉토리 내용을 출력할 때, 도트 파일에대한 정보도 출력한다.

-a 옵션이 없으면 도트 파일에 대한 정보는 출력되지 않는다.

### **ls – S**

(size)

파일 크기 내림차순으로 정렬하여 출력한다.

-S 옵션이 없으면 파일명 오름차순으로 정렬하여 출력한다.

### **ls -R**

(recursive)

하위 디렉토리와 그 하위 디렉토리까지 재귀적으로 반복해서 출력한다.

### **ls -lh**

파일 크기를 K, M, G 등 적절한 단위로 출력한다.

### **ls -lt**

(long time)

파일을 마지막으로 수정한 시각의 내림차순으로 정렬하여 출력한다.

### **ls – ltr**

(long time reverse)

파일을 마지막으로 수정한 시각의 오름차순으로 정렬하여 출력한다.

### **ls –help**

도움말을 출려한다.

ls 명령 뿐만 아니라, 모든 명령에 이 도움말 출력 옵션이 있다.

## **cp 명령**

copy

파일이나 디렉토리를 복사한다.

### **cp 명령은 다음과 같은 형태로 사용된다.**

cp 파일1 파일2 : 파일1을 복사하여 파일2를 생성한다. 파일2가 이미 있다면 덮어써진다.

cp 파일2 파일2 파일3 디렉토리1 : 파일1, 파일2, 파일3을 디렉토리1 아래에 복사한다. 디렉토리1이 없다면 에러가 발생한다. 디렉토리1 아래에 동일한 이름의 파일이 이미 있다면, 덮어써진다.

cp -R 디렉토리1 디렉토리2 : 디렉토리1과 그 아래 내용을 모두 복사하여 디렉토리2를 생성한다. 만약 디렉토리2가 이미 있다면, 디렉토리1과 그 아래 내용을 복사하여 디렉토리2/디렉토리1을 생성한다.

cp -R 디렉토리1 디렉토리2 디렉토리3 디렉토리4 : 디렉토리1, 디렉토리2, 디렉토리3 들과 그 아래 내용을 모두 복사하여 디렉토리4/디렉토리1, 디렉토리4/디렉토리2, 디렉토리4/디렉토리3 들을 생성한다. 디렉토리4가 없다면 에러가 발생한다.

### **cp 예**

cp test1.txt test2.txt : 현재 디렉토리의 test1.txt 파일을 test2.txt 이름의 파일로 복사한다. test2.txt 파일이 이미 있다면 덮어써지고, 없다면 새로 생성된다.

cp test1.txt .. : 현재 디렉토리 아래의 test1.txt 파일을 바로 위 디렉토리에 복사한다.

cp ../test1.txt : 바로 위 디렉토리의 test1.txt 파일을 현재 디렉토리에 복사한다.

cp test1.txt ~ : 현재 디렉토리의 test1.txt 파일을 홈 디렉토리에 복사한다.

cp test1.txt src : 현재 디렉토리에 src 디렉토리가 있다면, 현재 디렉토리의 test1.txt 파일을 src 디렉토리에 복사한다. 없다면 현재 디렉토리의 test1.txt 파일을 src 이름의 파일로 복사한다.

cp test1.c test2.c ../src : 현재 디렉토리의 test1.c 파일과 test2.c 파일을 부모 디렉토리 아래의 src 디렉토리에 복사한다. 부모 디렉토리에 src 이름의 디렉토리가 없다면 에러가 발생한다.

cp -R src src\_backup : src 디렉토리 아래 내용을 모두 복사하여 src\_backup 디렉토리를 생성한다. src\_backup 디렉토리가 이미 있다면, src\_backup/src 디렉토리를 생성한다.

cp -R src test backup : src, test 디렉토리 아래 내용을 모두 복사하여 backup/src, backup/test 디렉토리를 생성한다. backup 디렉토리가 없다면 에러가 발생한다.

### **cp 명령 옵션**

-I (interactive) : 기존 파일을 덮어쓰게 되는 경우, Y/N 를 확인한다.

-l (link) : 파일의 내용을 복사하는 대신 hard link를 만든다.

-s (symbolic link) : 파일의 내용을 복사하는 대신 symbolic link를 만든다.

-R (recursive) : 디렉토리와 그 아래 내용을 몽땅 복사한다.

-u (update) : 기존 파일을 덮어쓰게 되는 경우, 덮어쓸 파일이 덮여쓰여지는 파일보다 최신 날짜인 경우에만 덮어쓴다.

-v (verbose) : 복사 과정을 출력한다.

## **mv 명령**

move

파일이나 디렉토리 이름을 바꾸거나 다른 디렉토리 아래로 올긴다.

### **mv 명령은 다음과 같은 형태로 사용된다.**

mv 파일1 파일2 : 파일1의 이름을 파일2로 변경한다. 파일2가 이미 있다면 덮어써진다.

mv 파일1 파일2 파일3 디렉토리1 : 파일1, 파일2, 파일3을 디렉토리1 아래로 옮긴다. 디렉토리1이 없다면 에러가 발생한다. 디렉토리1 아래 동일한 이름의 파일이 이미 있다면, 덮어써진다.

mv 디렉토리1 디렉토리2 : 디렉토리2가 없다면 디렉토리1의 이름을 디렉토리2로 변경한다. 디렉토리2가 이미 있다면, 디렉토리1을 디렉토리2 아래로 옮긴다. 디렉토리2 아래 이미 디렉토리1이 있다면 에러가 발생한다.

mv 디렉토리1 디렉토리2 디렉토리3 디렉토리4 : 디렉토리1, 디렉토리2, 디렉토리3을 디렉토리4 아래로 옮긴다. 옮길 디렉토리와 동일한 이름의 디렉토리가 디렉토리4에 이미 있다면, 그 디렉토리의 이동만 취소되고 다른 디렉토리는 이동한다.

### **mv 예**

mv test1.txt test2.txt : 현재 디렉토리의 test1.txt 파일의 이름을 test2.txt 로 변경한다. test2.txt 파일이 이미 있다면 덮어써지고, 없으면 새로 생성한다.

mv test1.txt .. : 현재 디렉토리 아래의 test1.txt 파일을 바로 위 디렉토리로 이동한다.

mv ../test1.txt . 바로 위 디렉토리의 test1.txt 파일을 현재 디렉토리로 이동한다.

mv test1.txt ~ : 현재 디렉토리의 test1.txt 파일을 홈 디렉토리로 이동한다.

mv test1.txt src : 현재 디렉토리에 src 디렉토리가 있다면, 현재 디렉토리의 test1.txt 파일의 src 디렉토리 아래로 옮긴다. 없다면 현재 디렉토리의 test1.txt 파일을 src 이름으로 변경한다.

mv test1.c test2.c ../src : 현재 디렉토리의 test1.c 파일과 test2.c 파일을 부모 디렉토리 아래의 src 디렉토리로 옮긴다. 부모 디렉토리에 src 이름의 디렉토리가 없다면 에러가 발생한다.

mv src src\_backup : src\_backup 디렉토리가 없으면, src 디렉토리 이름을 src\_backup으로 변경한다. scr\_backup 디렉토리가 이미 있으면, src 디렉토리를 src\_backup 디렉토리 아래로 옮긴다.

cp src test backup : src, test 디렉토리를 backup 디렉토리 아래로 옮긴다. backup 디렉토리가 없다면 에러가 발생한다.

### **mv 명령 옵션**

-i (ineractive) : 기존 파일을 덮어쓰게 되는 경우, Y/N 를 확인한다.

## rm 명령

(remove)

삭제