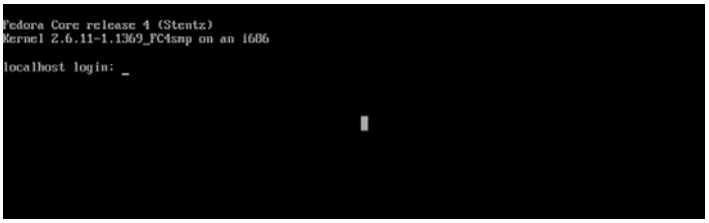
리눅스 기본 명령과 개념

만약 여러분이 윈도우나 다른 비 유닉스 운영체제를 사용하다가 지금 강의를 계기로 처음 리눅스 세계로 들어왔다면 앞으로 배워야 할 내용이 많다. 이 부분은 솔직히 말하고 싶다. 유닉스는 유닉스 만의 세계다

유닉스 운영체제를 사용해본 경험이 전혀 없는 학생들을 위해 유닉스 기본 명령과 개념을 소개할 것이다. 마이크로소프트 윈도우나 그밖에 다른 환경에서 리눅스의 세계로 왔다면 이 번 내용이 절대적으로 중요하다. 다른 운영체제와는 달리, 유닉스는 전혀 직관적이지 않다. 언뜻 보기에도 이상한 이름과 문법을 가진 많은 명령으로 가득하다. 이번 내용은 텍스트 프로세싱, 셸 문법, 그밖에 다른 암흑의 올가미 속으로 빠져들기 전에 유닉스가 아닌 운영체제에서 처음 온 사람들이 리눅스 시스템에 빠르게 적응할 수 있도록 도와줄 수 있는 기본적인 명령을 익혀 볼것이다.

로그인

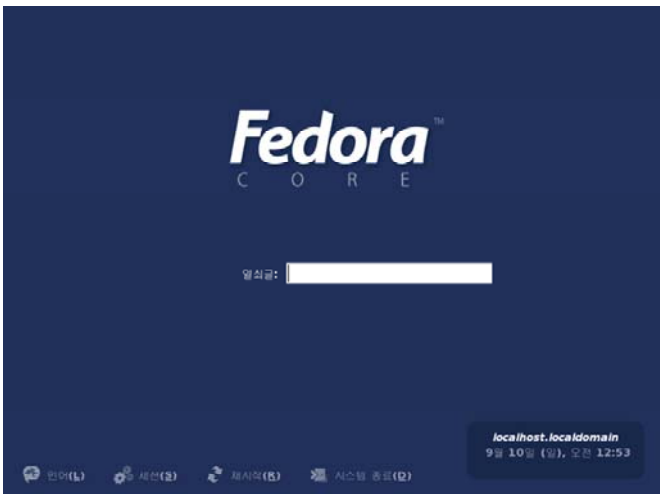
리눅스 설치가 제대로 이루어졌을 경우, 시스템을 부팅하면 다음과 같은 메시지나 화면이 나올 것이다.



로그인은 사용자를 서로 구별하게 해주는 행위다. 이렇게 함으로써 여러 명이 동시에, 같은 시스템에서 작업할 수 있으며, 여러분의 파일은 여러분만 접근할 수 있도록 보장해준다.

리눅스를 집에다 설치해놓고 이런 생각을 하고 있을 것이다. "좋다. 이 시슽엠은 오로지 나만 쓸 것이기 때문에 로그인할 필요가 없다." 그러나 여러분의 개인 계정으로 로그인하게 되면 어느 정도 시스템 보호 기능을 갖게 된다. 여러분의 계정은 중요한 시스템 파일을 파괴하거나 제거할 수 있는 권한을 가지고 있지 않기 때문이다. 이러한 작업을 하는 것은 오로지 시스템 관리자계정 뿐이다.

계정명에 root 라고 입력한다. 계정을 선택하여 입력하면 다음과 같은 메시지나 화면이 나타날 것이다.



여기서 정확한 패스워드를 입력해야 한다. 이때 터미널은 에코 기능(echo, 입력한 글자를 화면에 보여주는 기능)을 끄기 때문에 패스워드를 입력해도 화면에는 아무 것도 표시되지 않는다.(X Windows 환경에서는 예외) 이름과 패스워드는 모두 대소문자를 구분한다는 사실에 주의하자. 즉 Caps Lock 키가 눌러있으면 안 된다. 즉 root 대신 ROOT 라고 입력하면 작동하지 않는다.

root 계정으로 성공적으로 로그인을 하면 "[root@localhost ~]# " 이런 프롬프트가 나오게 되며 사용자로 로그인을 하게 되면 "[test@localhost ~]$ " 이런 프롬프트가 나오게 된다.

기본 명령어

앞으로 모든 명령어를 입력할때 셸의 모양인 [root@localhost ~] 이것을 # 으로 생략해서 작성하겠다.

pwd

- 윈도우즈는 나의 현재 위치를 주소표시줄에 처음 시작부터 현재 위치까지 전체 경로에 대해서 표시를 해준다. 하지만 유닉스 운영체제는 자신이 위치한 현재의 위치만 화면에 출력해 줄뿐이다. 그래서 리눅스 사용자는 pwd 라는 명령어를 이용해서 항상 자신의 위치를 파악하고 있어야 올마른 명령어 수행 결과를 기대 할수 있다.

cd

- 계속 한 디렉토리에만 머물러 있어야 한다면 재미없을 것이다. cd 라는 명령어를 사용하여 다른 디렉토리로 이동할 수 있다.

Ex> # cd /etc/rc.d
    # pwd
    /etc/rc.d

ls

- 디렉토리에 있는 내용을 보려면 ls 명령을 입력한다. 아무런 인수 없이 명령어를 사용하면 현재 디렉토리의 내용을 보여 준다. 다른 디렉토리를 보려면 다음과 같이 인수를 사용한다.

Ex> # ls : 현재 디렉토리의 내용 확인
    # ls /etc : 인수를 사용하여 /etc 디렉토리의 내용 확인
    # ls -l : 현재 디렉토리의 내용을 자세히 출력
    # ls -l /etc : 인수를 사용하여 /etc 디렉토리의 내용을 자세히 확인
    # ls -a : 현재 디렉토리의 숨겨진 파일 이나 디렉토리 표시
    # ls -a /etc : 인수를 사용하여 /etc 디렉토리의 내용중 숨긴 파일/디렉토리 표시
    기타 등등.. 학원에서 재수강 하던지 책 사서 자세히 볼것... 너무 많음 —,.—;;

cp

- 파일이나 디렉토리 를 복사할때 사용하는 명령어

Ex> # cp /etc/inittab /tmp : /etc/inittab 파일을 /tmp 디렉토리로 복사
    # cp -r /bin /tmp : /bin 디레톨리를 /tmp 디렉토리로 복사
    # cp –i : 복사할 디렉토리에 같은 이름의 파일이 있으면 확인
    # cp –p : 파일의 속성값을 유지한체로 복사
    # cp –v : 원본파일이 복사되는 과정 표시
    기타 등등.. 역시 학원에서 재수강 하던지 책 사서 자세히 볼것.... 너무 많음 —,.—;;

mv

- 파일과 디렉토리의 이름 변경이나 위치 이동시 사용

Ex> # mv /tmp/inittab /root : /tmp 디렉토리에 있는 inittab 파일을 /root 디렉토리로 이동
    # mv /root/inittab /root/abcd : /root 디렉토리에 있는 inittab 파일을 /root 디렉토리로 abcd 라는 이름으로 이동
    : 이름 바꾸기 기능

mkdir

- 새로운 디렉토리 생성시 사용되는 명령

Ex> # mkdir /test : 최상위 디렉토리 밑에 test 라는 이름의 디렉토리 생성
    # mkdir -p /1/2/3/4/ : 최상위 디렉토리 밑에 1 디렉토리를 생성하고 그밑에 2 3 4 디렉토리를 하위로 한꺼번에 생성

rmdir

- 디렉토리 삭제 명령어로서 사용 조건은 비어있는 디렉토리만 삭제 할 수 있다.

Ex> # rmdir /test : /test 디렉토리 삭제
    # rmdir /1 : /1 디렉토리를 삭제하는 명령어 이지만 하위에 2 3 4 디렉토리가 존재하므로 삭제 되지
    : 않는다.

rm

- 파일이나 디렉토리를 삭제하는 명령어

Ex> # rm /root/abcd : /root 디렉토리에 있는 abcd 파일 삭제. 삭제시 확인
    # rm -f /root/abcd : /root 디렉토리에 있는 abcd 파일 삭제. 삭제시 확인하지 않는다.
    # rm /1 : /1 디렉토리 삭제. 디렉토리는 삭제 되지 않는다.
    # rm -r /1 : /1 디렉토리를 포함한 하위 디렉토리 삭제. 삭제시 확인
    # rm -rf /1 : /1 디렉토리를 포함한 하위 디렉토리 삭제. 삭제시 확인하지 않는다.

touch

- 파일/디렉토리의 속성값 변경이나 0Byte 파일 생성시 사용

Ex> # touch /etc/passwd : /etc 디렉토리에 있는 passwd 파일의 시간정보가 변경 된다.
    # touch /tmp/abcd : /tmp 디렉토리에 abcd 라는 0Byte 파일 생성

cat

- 표준 입/출력을 관리하는 명령어로서 파일 출력이나 입력시에 사용

Ex> # cat > /tmp/1234 : 키보드로 입력한 내용을 /tmp 디렉토리에 1234라는 파일로 출력(저장)
    # cat /tmp/1234 : /tmp 디렉토리에 있는 1234 파일의 내용을 출력
    # cat >> /tmp/1234 : 키보드로 입력하는 내용을 /tmp 디렉토리에 있는 1234 라는 파일의 추가

head
- 파일의 앞에서 10줄을 화면에 출력
Ex> # head /etc/inittab : /etc 디렉토리의 inittab 파일의 내용중 앞에서 10줄만 출력(기본값)
head -20 /etc/inittab : /etc 디렉토리의 inittab 파일의 내용중 앞에서 20줄만 출력

tail
- 파일의 뒤에서 10줄을 화면에 출력
Ex> # tail /etc/inittab : /etc 디렉토리의 inittab 파일의 내용중 뒤에서 10줄만 출력(기본값)
tail -20 /etc/inittab : /etc 디렉토리의 inittab 파일의 내용중 뒤에서 20줄만 출력

more
- 파일을 출력시 한화면을 넘어가는 큰 파일의 경우 한 화면식 구분 지어 출력
Ex> # more /etc/inittab : /etc 디렉토리의 inittab 파일의 전체내용을 화면단위로 끊어서 출력
: Enter 키는 한줄씩, Space bar 는 한 화면식 이동(파일의 내용이 끝나면 프로그램 자동 종료)

less
- 파일을 출력시 한 화면을 넘어가는 큰 파일의 경우 한 화면식 구분 지어 출력
Ex> # less /etc/inittab : /etc 디렉토리의 inittab 파일의 전체내용을 화면 단위로 끊어서 출력
: more 의 기능을 모두 사용할 수 있으며 윗 방향키 아래 방향키도 사용할 수 있고 파일의 내용이 끝나더라도 프로그램이 종료 되지 않으며 q 를 입력해야지만 종료 된다.

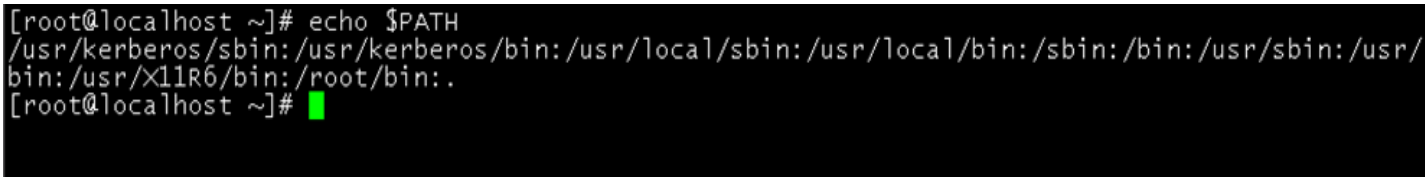
alias
- 명령어 재 정의 기능을 가지고 있는 명령어
Ex> # alias 1='clear' : 숫자 1 키를 입력후 Enter 키를 입력하면 clear 명령이 동작한다.
unalias 1 : 숫자 1키에 정의 했던 명령어를 삭제

명령이란

유닉스에는 엄청나게 많은 명령이 있으며 또 여러분이 추가하 수도 있다. 바로 이것이 한정된 수의 명령만을 가지는 대부분의 운영체제와 결정적으로 다른 점이다. 그렇다면, 유닉스 명령이란 무엇이며 저장되는 방법은 어떤 것이지 살펴보자. 유닉스 에서 명령은 그냥 파일일 뿐이다. 예를 들어, ls 명령은 bin 디렉토리에 들어 있는 바이너리 파일이다. 따라서 ls 대신에 /bin/ls(절대경로) 을 사용할 수 있다.

이 때문에 유닉스는 유연성을 가지면서도 강력하다. 새로운 유틸리티를 제공하려면 시스템 관리자는 명령을 저장하는 표준 디렉토리에 설치하기만 하면 된다. 또한 한 명령의 여러 버전도 설치할 수 있다. 예를 들어, 테스팅을 위한 새 버전의 유틸리티를 한 곳에 설치하고 나머지 버전은 다른 곳에 둠으로써 사용자들이 선택하여 실행할 수 있도록 할 수 있다.

일반적으로 겪게 되는 문제 중에 하나가 시스템에 있을 것이라고 예상되는 명령을 입력하면 'Not found'라는 메시지를 만나게 되는 것이다. 아마도 셸이 검색하지 않는 위치에 명령이 저장되어 있기 때문일 것이다. 셸이 명령을 찾기 위해 검색하는 디렉토리 목록을 경로라고 한다. 여러분의 경로가 어떻게 설정되어 있는지 보려면 다음과 같이 한다.



주의 깊게 살펴보자. 경로명은 콜론(:)으로 구분되어 있다. 여기서 첫번째 경로는 /usr/kerberos/sbin, 둘째는 /usr/kerberos/bin이 된다. 따라서 한 명령의 두 버전이 /usr/kerberos/sbin에도 있고 /usr/kerberos/bin에도 있으면 /usr/kerberos/sbin에 있는 것이 먼저 실행된다.

예제에서 마지막 경로명은 그냥 도트(.)다. 도트는 현재 디렉토리를 가리킨다. 윈도우의 명령 해석기와 달리 유닉스는 자동으로 현재 디렉토리를 찾지 않는다. 위에서처럼 명시적으로 지정해줘야만 찾는다. 이것은 편리하기는 하지만 현재 디렉토리를 경로에 추가하는 것은 보안 면에서 나쁘다(크래커가 들어와서 여러분의 현재 작업 디렉토리에 악의적인 프로그램을 설치해 두었다고 생각해보면 알것이다).

리눅스의 구성

리눅스는 크게 시스템의 하드웨어 관리 작업을 비롯하여 시스템이 운영되기 위해서 필요한 기본적인 관리 작업을 하는 커널, 사용자와 커널 사이에서 다리 역활과 명령어 해석기 역활을 하는 쉘, 하드디스크나 CD-ROM과 같은 저장 장치에 데이터를 효율적으로 관리하는 디렉토리 트리, 이 세 가지 요소로 구성된다.

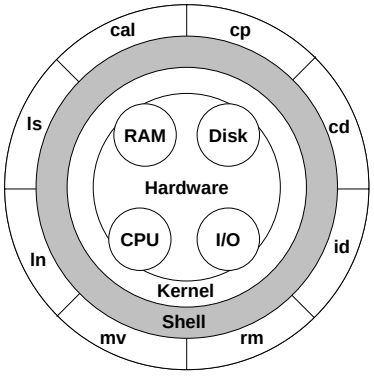
커널

커널은 유닉스에서 가장 핵심적인 부분으로 시스템의 하드웨어 관리와 관련된 작업을 주로 담당한다. 즉, 시스템 부팅 시에는 시스템 자체의 초기화를 위하여 현재 설치된 하드웨어의 초기화나 필요한 프로세스를 띄우는 역활을 담당하며, 그 이후에는 하드웨어, 스왑 공간, 프로세스, 메모리, 파일 시스템 등의 관리를 담당한다. 인간의 뇌 중에서 간뇌에 해당하며 가장 기본적인 운영체제의 작업을 커널에서 하게 된다. 또한 하드웨어에 데이터를 읽거나 기록하려고 할 때, 하드웨어와 이들 사이에서 다리 역활을 한다.

예를 들어 웹 서핑을 한다면 네트워크 어뎁터를 통하여 데이터가 오고 가야지만 서핑이 가능하다. URL을 직접 입력하거나 링크를 클릭하면 웹 브라워저에서 네트워크 어뎁터에 데이터를 기록하고 웹 서버에 접속한 데이터가 네트워크 업댑터에 전달이 되면 이 데이터를 웹 브라우저가 받아서 처리를 해야 한다. 이 모든 작업을 웹 브라우저가 한다면 할 일이 얼마나 많을까? 모든 네트워크 어댑터를 제어할 수 있어서 데이터를 기록하거나 읽을 수 있어야 하고, 마우스의 움직임도 신경 쓰기 위해서 마우스를 제어할 수 있어야 하고, 인쇄를 위해서는 프린터를 제어할 수 있어야 하며, 화면에 그림이나 글자를 출력하기 위해서는 그래픽 어댑터를 제어해야 한다. 새로운 네트워크 어댑터가 출연하면 제어하는 루틴을 새로 추가해야 할 것이다. 이처럼 웹 서핑을 하기 위해서 웹 브라우저가 해야 할 일이 너무 많다. 예전의 도스에서 실행되는 웹 브라우저는 실제 네트워크 어댑터, 마우스, 그래픽 어댑터, 프린터 등의 제어와 관련한 작업도 모두 담당했었다. 하지만 요즘 시스템에서 동작되는 웹 브라우저는 이런 세세한 작업은 전혀 신경 쓰지 않는다. 하드웨어에 대한 제어와 데이터 입출력은 모두 커널에서 작동하기 때문이다. 새로운 하드웨어의 지원도 커널이 해야 될 몫이다. 애플리케이션은 쉘을 통해서 커널에게 요구만 하면 모든 필요한 작업이 이루어진다. 네트워크 어댑터에 데이터를 입출력 한다면 그냥 쉘을 통해서 네트워크 어댑터에 데이터를 보내거나 받으면 작업은 끝이다. 각각의 네트워크 어댑터를 어떻게 제어를 해서 데이터 입출력을 하는 지는 전혀 신경을 쓰지 않는다. 네트워크 어댑터와 같은 하드웨어 제어는 모두가 커널의 역활이기 때문이다.

셸

셸을 한마디로 표현하면 명령어 해석기이다. 사용자가 셸 프롬프트에서 내린 명령어를 해석하여 커널에게 전달하고, 커널이 실행한 것을 처리하여 사용자에게 전달하는 역활을 한다. 커널이 하드웨어와 셸 사이에서 다리 역활을 한다면, 셸은 커널과 사용자나 사용자가 실행한 애플리케이션 사이에서 다리 역활을 한다. 다음 그림에 커널과 셸의 관계가 잘 나타나 있듯이 우리가 유닉스 시스템을 사용할 때 실제로 접하는 것은 셸뿐이다. 커널은 우리가 직접 접할 수는 없고, 단지 셸을 통해서 커널을 접할 수 있다. 우리가 사용하는 모든 프로그램들은 셸에서 먼저 해서을 하는 전처리 과정을 거치고, 그 결과를 커널이 받아서 처리할 뿐이다. 커널에서 처리한 작업의 결과도 셸에서 받아서 여러 애플리케이션에게 적절하게 전달할 뿐이다.



리눅스 에서 제공되는 쉘

C Shell

BDS와 Sun OS를 개발한 버클리의 천재 빌 조이(Bill Joy)가 만든 셸로써 본 셸과는 쓰는 방법이 많이 다르다. 셸 프로그래밍에 쓰이는 문법이 이미 존재하는 C언어의 문법을 따서 만든 탓에 C 프로그래머들 사이에서 크게 유행했다. 그리고 Korn Shell이 나오기 저에 Bourne Shell이 가지지 않은 펴린한 기능들(알리아스(alias, history), 명령 행 치환(command line substituting), 작업 제어(job control) 등)을 제공한 근대적인 셸의 시초이다. Korn Shell이 널리 쓰이기 전까지 C Shell이 제공하는 편리한 기능 덕에 많은 사용자를 확보했었다. 초창기에 많은 버그가 있었고, 문법이 애매하여 시스템에서 쓰이는 쉘 스크립트로는 잘 쓰이지 않았다. Bourne Shell이나 Korn Shell과 달리 C Shell을 쓰면 셸 프롬프트는 시스템의 이름 다음에 % 기호가 붙어서 firefox% 의 형태로 출력된다.

Korn Shell

AT&T의 데이빗 콘(David Korn)이 만든 셸로써 본 셸이 가지는 모든 기능을 가지고 있으며, 추가적으로 C Shell이 가지는 alias, history, job control 등의 기능과 명령 행 편집 과 같은 새로운 기능을 많이 추가하였다. SVR4에서 Bourne Shell을 대체한 새로운 표준 셸로 쓰이고 있다. Bourne Shell과 마찬가지로 Korn Shell을 쓸 경우에도 셸 프롬프트는 $로 표시된다.

Bash Shell

리눅스에서 가장 많이 쓰이는 셸로서 Korn Shell에 C Shell의 많은 장점을 추가한 느낌이 있다. 기존의 Korn Shell 사용자와 C Shell사용자 모두 쉽게 Bash Shell로 전환하여 사용할 수 있다. 주요 기능으로서 향상된 명령어 행 편집 기능, 명령어 행 치환 기능, 명령어의 이름을 완성하는 기능, 명령어의 이름, 아규먼트로 사용된 파일의 이름, 셸 변수의 이름 중 잘못된 것을 정정하는 기능, 강화된 히스토리 기능 등이 있다. 셸 프롬프트는 배시 셸의 이름과 버전이 표시되고 $ 기호가 추가된 bash-2.05$ 형태로 출력된다.

Z Shell

기존의 Korn Shell을 다시 재작성 하면서 현대적인 기능을 많이 추가하였다. 새롭게 추가된 기능으로 는 향상된 명령어 행 편집 기능, 명령어의 이름, 아규먼트, 셸 프롬프트는 C 셸과 동일하게 시스템의 이름 다음에 % 기호가 붙어서 firefox% 형태로 출력된다.

TC Shell

BSD 계열에서 가장 많이 사용되는 C Shell을 재작성한 셸로서 명령어 행 치환 기능에 덧붙여 명령어 행 편집 기능, 명령어 완성 기능, 명령어 이름, 아규먼트, 셸 변수의 이름을 정정하는 기능, 향상된 히스토리 기능 등이 추가 되었다. 셸 프롬프트는 >로 표시 된다.

디렉토리 트리

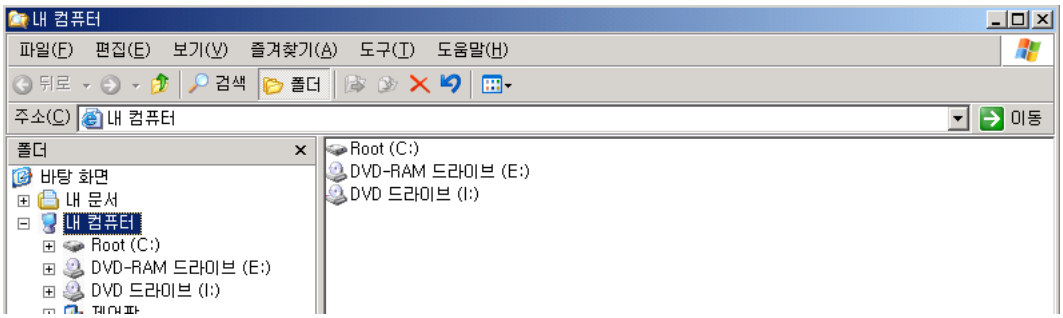
유닉스에서는 데이터를 하드디스크에 저장하기 위하여, 디렉토리와 파일로 이루어진 디렉토리 트리라는 구조를 사용한다. 이 구조는 파일을 조직적으로 보관하기 위하여 디렉토리를 쓰로 있고, 디렉토리들이 서로 연관된 모습이 나무를 거꾸로 뒤집어 놓은 듯한 모습과 흡사하여 디렉토리 트리라고 한다.
나무의 시작이 뿌리(root) 이듯이 디렉토리 트리도 그 시작은 root부터 시작한다. 나무 뿌리 위에 굵은 가지인 줄기가 있고, 이 가지에 작은 가지가 있거나 나뭇잎이 달린다. 디렉토리 트리도 root 밑에 디렉토리가 있고, 이 디렉토리 밑에 다른 디렉토리가 있거나 파일이 존재한다. 나무의 가지가 디렉토리 트리에서는 디렉토리에 해당하고, 나무의 잎은 파일에 해당한다.

윈도우에서도 이와 유사한 방법으로 하드디스크에 데이터를 구조화하여 저장한다. 다만 차이점이 있다면 디렉토리와 디렉토리를 구별하는 기호로 ₩를 쓰는것이 아니라 /를 쓴다는 것과 대소문자를 구별하지 않는다는 것이다.

기본적인 디렉토리

root(/) 디렉토리

시스템의 가장 상위에 위치한 디렉토리로서 모든 디렉토리의 출발이 된다. 윈도우는 root 디렉토리가 각 파티션마다 존재한다. 즉 C:₩, D:₩, E:₩등과 같이 모든 파티션 마다 root 디렉토리가 존재한다. 하지만 솔라리스는 윈도우와 달리 오직 하나의 root 디렉토리만이 존재한다. 솔라리스에 존재하는 여러 다른 디렉토리는 모두 다 이 root 디렉토리 아래에 존재한다.

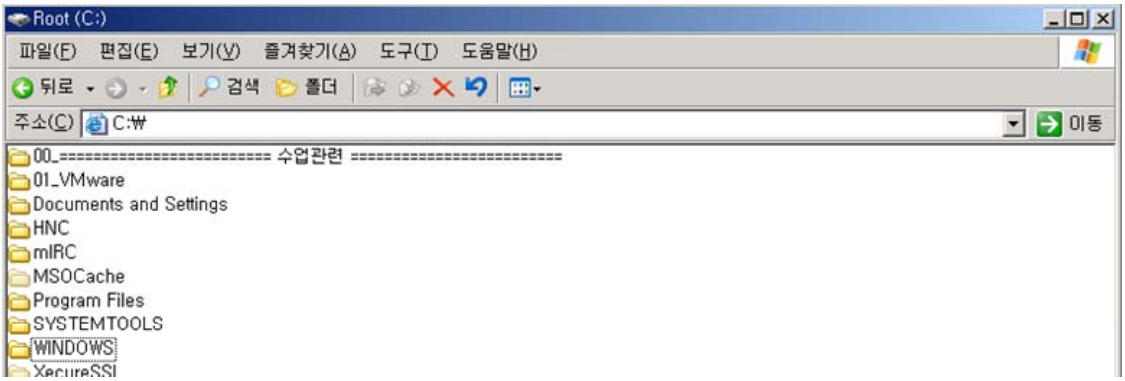


/usr 디렉토리

리눅스에서 기본적으로 제공하는 수많은 프로그램이 존재하는 디렉토리이다. 리눅스의 모든 명령어와 명령어가 실행하는 데 필요한 부가적인 데이터 파일, 프로그래밍을 위한 툴, GUI환경을 제공하기 위한 프로그램들도 이 디렉토리 밑에 존재한다. 윈두우에서는 C:₩Windows나 C:₩WinNT에 해당한다.

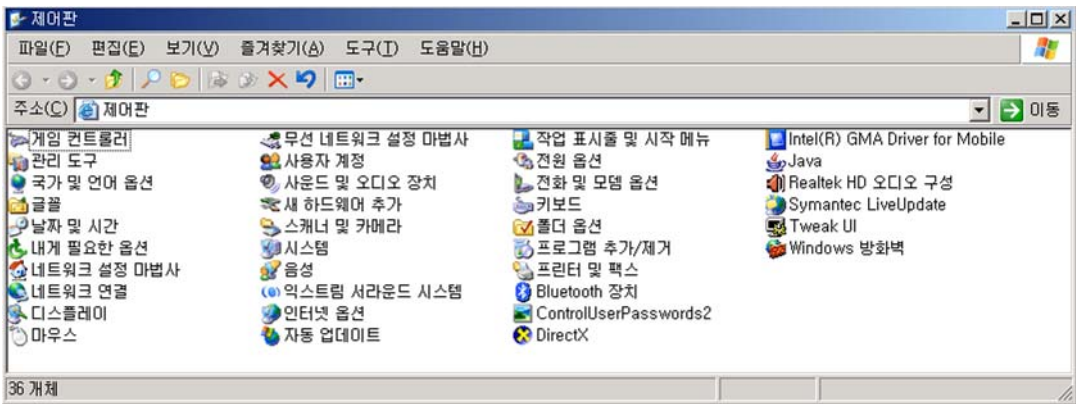
/opt 디렉토리

유닉스의 설치가 끝난 후 유닉스에서 기본적으로 제공하지 않는 프로그램을 추가로 설치하면 이 디렉토리 밑에 기본적으로 설치가 된다. 넷스케이프 6.X, C 컴파일러, GNOME 등의 부가 소프트웨 등이 이 디렉토리 밑에 설치된다. 윈두우에서는 C:₩Program Files에 해당한다. 리눅스(Red Hat)에서는 사용되어지지 않고 /usr 디렉토리를 사용한다.



/etc 디렉토리

시스템에 관련된 설정 데이터가 아스키 텍스트 파일로 저장되는 디렉토리이다. 시스템을 관리하기 위하여 여러 명령어를 사용하거나 설정 파일들을 수정한다. 명령어를 사용하는 것은 시스템이 부팅을 하면 원래대로 되돌아가는 경우가 대부분이다. 부팅을 하더라도 병경 내용을 그대로 유지하는 겨우 명령어가 /etc/ 디렉토리 밑의 시스템 설정 파일을 변경을 할 경우이다. 시스템을 관리한다는 것은 명령어를 사용하여 여러 작업을 하는 것을 의미하지만, /etc 디렉토리 밑의 여러 시스템의 설정 데이터를 변경한다는 것도 같은 의미이다.
윈도우에서 제어판의 여러 프로그램을 실행하는 것과 유사한 역활을 한다.



/bin
ls와 같이 가장 중요한 유닉스 명령이 위치한다.

/usr/bin
기타 명령을 포함한다. /bin 과 /usr/bin의 구별은 임의적이다. 초창기 디스크 공간이 적은 환경에서 유닉스가 명령을 구분했던 방식으로부터 유래한다.

/usr/sbin
시스템 관리를 위해 관리자가 사용하는 명령이다.

/boot
때때로, 부팅 중 사용되는 커널과 기타 파일이 위치하는 곳이다.

/etc
네트워킹, NIS, 메일 등으 | 하위 시스템에서 사용하는 파일이다. 보통 네트워크 서비스, 마운트할 디스크 등의 테이블 정보를 담고 있는 파일이 위치한다. 여기에 놓인 많은 파일은 시스템을 부팅하거나 개별 시스템을 시동하는 데 사용한다.

/var
다양한 유틸리티에서 사용하는 로그 파일과 같은 관리용 파일이다.

/usr/lib
libc.a와 같은 표준 라이브러리가 위치한다. 프로그램을 링크 할 때 링커는 -l 옵션이 지시한 라이브러리를 여기서 찾는다.

/dev
이 디렉토리에는 이른바 장치 파일이라고 하여 파일시스템과 하드웨어간의 인터페이스를 하는 파일이 들어 있다.

/proc
/dev가 파일시스템과 하드웨어 장치간의 인터페이스 역활을 하듯, /proc은 파일시스템과 실행 프로세스, CPU와 메모리간의 인터페이스다. 여기에 있는 파일(실제 파일은 아니고 여러분이 파일을 보려고 할 때 자동으로 만들어내는 가상의 파일이라고 할 수 있다.)을 통해 특정 프로세스에 대한 환경, CPU의 상태와 설정, I/O 포트 설정 방식을 알 수 있다.

작성자 : Xstone Member 김진홍 (firefoxexpert@gmail.com / kjh1908@paran.com / http://www.hackerscollege.org)