

[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(1일차)

수업 OT

VMWARE 사용법

리눅스 설치

■수강동의

온라인 출석부 사이트(<http://mgr.eduitbank.com>)에 로그인 하신 후
수강동의 를 체크해 주시기 바랍니다.

나의정보(오른쪽상단)-출석정보(왼쪽메뉴)-과목명 클릭-수강동의 확인버튼 클릭

■출석확인

■수업 준비사항

1. 윈도우 탐색기 실행후 드라이브 확인
2. 2_주말반 폴더 확인후 없으면 생성
3. 2_주말반 폴더 안에 "공용" 폴더 생성(폴더 있으면 생략)
4. 2_주말반 폴더 안에 자신이 사용할 폴더 생성
(폴더 이름은 자신의 이름으로 지정)
5. 인터넷 되는지 확인하기

예)

D:\2_주말반

D:\2_주말반\공용

D:\2_주말반\홍길동

■강사 게시판

<http://car2100.ivyro.net/hs1>

사용자 : start

암호 : unix

■알FTP

자료다운

알FTP 이용하여 강의장 임시서버 접속한 후

아래의 파일을

D:\2_주말반\공용 폴더에 다운받기

(윈도우 탐색기에서 확인해 보고 자신의 컴퓨터에 파일 있으면 다운 생략)

XP_2013.alz

■수업 OT

■가상화

1. 하나의 물리적인 하드웨어 박스를 가상적으로 몇 개의 논리적 시스템으로 제공함으로써 시스템 자원을 효율적으로 사용하는 방식

2. vmware : 윈도우 운영체제 상태에서 가상의 컴퓨터를 만들고 그 안에 다른 운영체제를 설치해서 사용할 수 있는 가상화 프로그램

■PDF 파일 보기

수업용 FTP 서버 120.UPLOAD2 폴더 안의

Foxit Reader-kr.exe 를 다운받아 사용

■vmware 사용하기

-본점 402 호는 Master(관리자) 계정을 막아 놓아서 프로그램 설치 및 제거가 안 됨

-vmware 프로그램 설치 연습은 xp 가상머신을 이용함

1. D:\2_주말반\공용 폴더의 XP_2013.alz 를

자신의 폴더에 압축 풀기

예)

D:\2_주말반\자신의이름폴더\XP_2013

2. VMWARE 실행하기

File-Open 클릭

위의 압축 풀 폴더를 찾아간 후 XP_2013.vmx 파일 열기

3. 녹색 삼각형 버튼(>) 눌러서 부팅시키기

질문나오면 "I copied it" 선택

■리눅스 종류(리눅스 배포판, Distribution)

- 1) Redhat Fedora Linux(공개버전)
- 2) Redhat Enterprise Linux(유료버전)
- 3) CentOS
- 4) Debin Linux

.....

☞레드햇 엔터프라이즈 리눅스(Redhat Enterprise Linux)

리눅스라고 하면 레드햇을 생각할 정도로 유명한 배포판
유료 버전이며 RHEL 로 표시하기도 한다.

<https://www.redhat.com/wapps/store/catalog.html>

☞페도라(Fedora)

레드햇이 지원하는 페도라 프로젝트 라는 커뮤니티에서 만든 배포판(무료 버전,테스트 버전)

☞센트OS(CentOS)

레드햇 엔터프라이즈 리눅스(RHEL)를 기반으로 만든 운영체제
RHEL 의 소스코드는 GNU GPL 을 따르므로 공개되어 있다.
이 소스 코드를 이용해 만든 것이 센트OS 이다.

☞데비안(Debian)

자유 소프트웨어와 오픈소스 소프트웨어로 구성된 배포판 중에서
가장 인기 있고 영향력을 갖춘 배포판이다.

☞우분투(Ubuntu)

데비안 GNU/Linux 를 기반으로 만든 배포판으로 최근에 인기가 있는 배포판이다.
줄루족의 말에서 이름을 가져왔다고 하며
우분투는 '다른 이들을 위한 인간애' 라는 의미라고 한다.

☞슬랙웨어(Slackware)

초기 리눅스 배포판이면서 아직도 배포되고 있다.
1993년 슬랙웨어 리눅스 라는 회사의 패트릭 볼커딩이 발표했고
2008년 12월 10일에 12.2 버전이 나왔다.

슬랙웨어는 *.tgz 라는 압축 파일 형태의 패키지를 쓴다.

☞ 오픈수세(openSUSE)

수세(SUSE Linux)는 2004년 노벨(Novell)에 인수되었다.

노벨은 수세 리눅스 프로페셔널을 100% 오픈 소스로 발표하기로 했는데
그 결과가 오픈 수세다.

수세 리눅스는 슬랙웨어 리눅스를 기반으로 독일에서 만든 것으로
SUSE 는 Software und System Entwicklug 의 약자다.

☞ 맨드리바(Mandriva)

처음에는 맨드레이크 리눅스로 불렸으나 9.2 버전부터 맨드리바로 바뀌었다.

유프미(urpmi)라는 자체 패키지 관리자를 쓰고 맨드리바 제어 센터와
결합되어 있는 rpm드레이크(rpmdrake)라는 GUI 툴을 이용하면 소프트웨어 설치가 쉽다.

※리눅스 종류가 틀려도 기본 명령어는 동일하며 설정파일, 응용 프로그램 등이 차이가 난다.

■ 수업용 리눅스 : CentOS

1. CentOS 특징

- 1) Redhat Enterprise Linux 를 복제(Clone)한 운영체제
- 2) 무료 버전
- 3) 레드햇 리눅스와 사용법이 유사하다.

참조)

<http://ko.wikipedia.org/wiki/CentOS>

2. CentOS 다운로드 사이트

- 1) CentOS 홈페이지 : <http://www.centos.org>
- 2) Daum 사이트 :
<http://ftp.daum.net>
<http://ftp.daum.net/centos/6.4/isos/i386/>

3. CentOS 버전별 비교

<http://wiki.centos.org/About/Product>

■vmware 가상머신 만들기(실제 컴퓨터 윈도우 7 에서 해야 함)

1. vmware 실행

2. File-New Virtual Machine

Virtual Machine : 가상 컴퓨터

Window : 새로운 창 출력

3. Welcome 메시지

Typical : 미리 설정된 옵션을 사용하는 방식

Custom : 사용자가 옵션을 선택하는 방식

Typical(표준설치)-다음

Install from: 이라고 나오면

세번째 버튼 (*)I will install the operating system later 선택

4. Guest Operating System-Linux 선택

Version-CentOS 선택-다음

5. Virtual machine name : Cent_WeekendAM10

Location : D:\2_주말반\자신의폴더\Cent_WeekendAM10

--->Browse 버튼 누른 후 자신이 사용할 폴더로 찾아가기

"새폴더 만들기" 버튼을 누른다음

자신의폴더에 새폴더를 생성하고

F2 눌러서 Cent_WeekendAM10 으로 이름변경하기-확인-다음 클릭

6. Disk Capacity-20G 그대로 사용-Next 클릭

////////////////////////////////////

Store virtual disk as a single file

가상 하드디스크 파일을 1개의 파일로 만드는 것

Split virtual disk into multiple files

가상 하드디스크 파일을 분할해서 만드는 것

////////////////////////////////////

7. Finish 클릭

■용어 정리

1. 파티션(Partition)

하나의 하드디스크를 여러 개의 영역으로 나누어서 사용할 수있는데 이 영역을 파티션이라 한다.

리눅스가 설치되려면 기본적으로 2 개의 파티션이 존재해야 한다.

1) / 파티션(루트 파티션이라고 부른다.)

2) swap 파티션(메모리 처럼 사용하는 영역이다.)

swap 은 물리적 메모리의 2배 정도 할당해 주면 된다.(권장용량)

2. 마운트(mount)

파티션을 사용자가 이용할 수 있도록 디렉토리(윈도우의 폴더 개념)와 연결하는 것

[C: | D:] (윈도우 운영체제)

하드디스크

[/ | swap | /home] (리눅스)

하드디스크

3. 리눅스에서 하드디스크 장치명(5 버전 까지 해당)

/dev/hda (첫번째 IDE 방식의 하드디스크)

/dev/sda (첫번째 SCSI 방식의 하드디스크)

※6 버전에서는 방식에 상관 없이 /dev/sd? 로 나타남(/dev/sda, /dev/sdb ...)

■해상도 조정

터미널 실행

#gnome-display-properties

■로그인/로그아웃

1. 로그인(login)

계정을 이용하여 리눅스 명령어나 프로그램을 실행시킬 수 있는 상태로 들어가는(in) 것
시스템 사용 권한을 얻는 과정

2. 로그아웃(logout)

리눅스 사용후에 리눅스 명령어나 프로그램을 실행시킬 수 없는 상태로 빠져나오는(out) 것

1)리눅스 바탕화면-메뉴표시줄-시스템-로그아웃-로그아웃 버튼 선택

2)단축키는 Ctrl + Alt + Back space

리눅스를 사용 하려면 로그인을 해야하고 사용후에는 로그아웃을 한다.

3. 리눅스 계정 종류

계정(Account) : 리눅스로 로그인할 때 사용되는 사용자명(아이디)

터미널 : 리눅스 명령어를 입력해서 실행하고 결과를 확인할 수 있는 프로그램(MS-DOS 창과 유사)

1)수퍼유저(Super User)

루트(root) 를 의미, 리눅스의 모든 작업을 할 수 있는 관리자 계정

터미널을 실행하면 # 으로 나타난다.

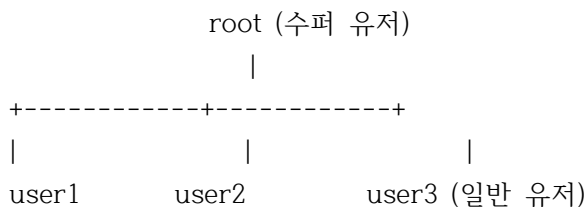
2)일반유저

루트를 제외한 사용자는 모두 일반 계정이다.

일반유저는 일반적인 프로그램은 실행할 수 있으나

네트워크 설정, 하드디스크 작업 등 시스템 관리 작업은 할 수 없다.

터미널을 실행하면 \$ 로 나타난다.



■리눅스 프롬프트 구조

[현재로그인한사용자아이디@호스트명(컴퓨터이름) 현재작업디렉토리(폴더)]#
---> 관리자 계정(root)

[현재로그인한사용자아이디@호스트명(컴퓨터이름) 현재작업디렉토리(폴더)]\$
---> 일반 계정(apple)

■vmware 아이콘 설명

1. ■ 가상머신 종료(Power Off)

2. || 가상머신 일시정지(Suspend)

3. ► 가상머신 시작(Power On)

4. 일시정지 테스트

■리눅스 종료 방법

1. 리눅스 바탕화면 위쪽 메뉴표시줄-시스템-끄기 선택후 시스템 종료 버튼 클릭

2. 리눅스 바탕화면-마우스 오른쪽 버튼-터미널 열기 선택

```
#shutdown -h now
```

🔗shutdown 형식

shutdown 옵션 시간 메시지

사용 예 :

1)shutdown -r +2 (분단위)

---> 2분뒤에 리부팅 하는 명령

2)shutdown -h now

---> 즉시 종료

3)shutdown -r now "system check"

---> 즉시 리부팅, system check 메시지가 접속한 사용자들에게 전달 됨

4)shutdown -h 시간:분

-r // reboot

-h // halt (종료)

[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(2일차)

리눅스 디렉토리 구조

리눅스 명령어 형식

매뉴얼, 도움말 확인 방법

리눅스 기본 명령어

■리눅스 디렉토리구조

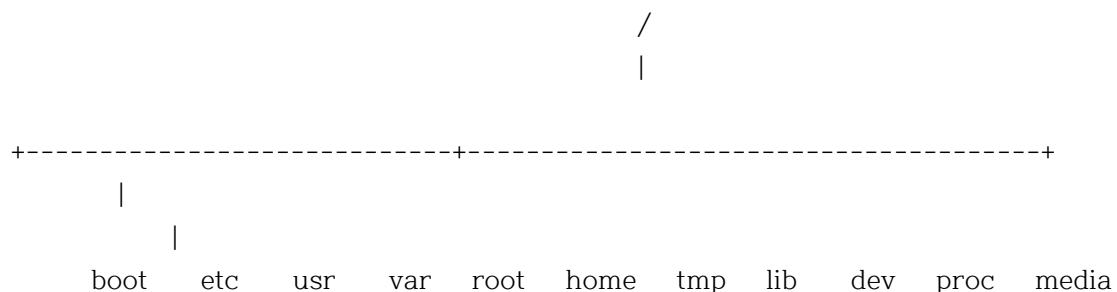
리눅스 메뉴표시줄-프로그램-시스템 도구-파일 브라우저 실행

1.리눅스는 설치를 하면 최상위 디렉토리(/ ---> 루트 디렉토리라고 읽는다.) 아래에 비슷한 성격의 파일을 저장하고 있는 여러 디렉토리가 자동으로 생성된다.
디렉토리는 윈도우의 폴더이다.

2.리눅스는 대소문자를 구분한다.

(디렉토리 또는 파일 이름, 명령어, 명령어 옵션, 계정 등)

관리자 계정 루트는 소문자 root(대문자 ROOT 는 로그인이 안 된다.)



/boot : 커널 디렉토리(부팅 관련 파일)

/etc : 시스템 설정파일 디렉토리

/usr : 응용프로그램 디렉토리

/var : 로그파일, 메일 디렉토리

/root : 관리자 홈디렉토리

/home : 일반계정 홈디렉토리 베이스

/tmp : 임시 디렉토리

/lib : 라이브러리 디렉토리

/dev : 장치파일 디렉토리

/proc : 프로세스 정보 디렉토리

/media, /mnt : cdrom 마운트 디렉토리

■실행파일(명령어) 디렉토리

/bin

essential binaries

필수 명령어(All user = root + 일반유저)

/sbin

system binaries

필수 관리자용(root) 명령어

/usr/bin

most user commands

대부분의 사용자 명령어

/usr/sbin

non essential binaries

비 필수적인 명령어(시스템 관리용)

/usr/local/bin

local software

Linux OS 설치후에 추가된 프로그램 명령어

■자격증 문제

LPIC(수업용).pdf 참조

327 번

■파티션 설정 예

아래는 예를 든 것으로 서버의 성격과 하드디스크의 용량을 고려하여 파티션 설정을 해 준다.

(하드디스크 용량을 100G 로 가정)

1)네임서버(기본형)

/ (98%)

swap (2%)

2)web 서버, ftp 서버

/ (15%)

swap (2%)

/home (나머지)

3)메일서버

/ (15%)
swap (2%)
/home (30%)
/var (나머지)

4)DB 서버

/ (15%)
swap (2%)
/usr/local (나머지)

■리눅스 명령어(command) 형식

기본 형식)

#명령어 [옵션] [인자값]

---> [] 부분은 생략 가능하며

명령어 와 옵션 인자값 구분은 스페이스바(공백) 로 구분을 해 주어야 한다.

명령어나 옵션 다음에 적는 단어를 argument, 인자값, 인수 라고 부른다.

옵션은 보통 - 를 지정하고 생략 가능한 경우도 있다.

■리눅스 명령어 실행 형식 종류

1. 명령어 만 사용하는 경우
2. 명령어 옵션 을 사용하는 경우
3. 명령어 옵션 인자값 을 사용하는 경우
4. 한 줄에 여러 명령어 를 사용하는 경우

예1) #ls

예2) #ls -l

예3) #ls -l /tmp

예4) #ls ; date

옵션은 - 를 한 번만 사용해도 되고 여러 번 사용해도 된다.

#ls -a -l

```
#ls -al
```

```
#ls -la
```

옵션의 순서는 변경할 수도 있으나 순서를 지켜야 하는 경우도 있다.

명령어에 대한 사용법 및 옵션은 man 또는 --help 사용해서 확인한다.

☞man 명령어

매뉴얼(설명서) 페이지 확인

☞--help

도움말을 확인하는 옵션

☞형식

#man 옵션 리눅스명령어

#명령어 --help

예)

```
#man ls
```

```
q // 매뉴얼 종료
```

```
#man -K "print a calendar"
```

-K(대문자) // 키워드 지정 옵션

---> -K 다음에 지정한 문자열을 매뉴얼 페이지에서 검색

---> print a calendar 에서 공백이 여러 번 들어가면 안 되고 대소문자를 구분함

```
#man -K 달력
```

--->한글 설명서가 있는 경우만 검색 됨

한/영 전환 : Ctrl + space

```
#ls --help
```

▣명령어 연습

터미널 실행하기

```
#man ls
```

--->터미널에서 man ls 를 입력후 엔터누르기
(#은 입력하는 것이 아님)

---> ls 명령어에 대한 매뉴얼이 출력된다.
ls 는 디렉토리 목록 확인 명령
리눅스의 명령어는 .exe .com 등 확장자로 구분하지 않는다.

매뉴얼 페이지 확인 시 사용하는 키

- 1) 스페이스바 : next page
- 2) f(또는 ctrl + f) : next page
- 3) b(또는 ctrl + b) : previous page
- 4) 엔터 : next line
- 5) PageUp, PageDown, 화살표 키를 이용할 수 도 있다.
- 6) h : 매뉴얼 도움말(help) 출력(도움말 종료는 q)
- 7) -N 입력후 엔터 : 라인번호 출력
- 8) -n 입력후 엔터 : 라인번호 해제
- 9) G : 마지막 라인으로 이동
- 10) 1G 또는 소문자 lg : 첫번째 라인으로 이동
- 11) 100G(또는 100g) : 100 번 라인이 있는 페이지로 이동(G 는 Go 를 의미)
- 12) q : 매뉴얼 종료(quit)

🔍whereis 명령어

실행파일 위치, 매뉴얼 파일 위치 출력

🔍passwd : 암호(password)를 설정하거나 변경하는 명령어

#whereis passwd

passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz (1번 설명
서 파일) /usr/share/man/man5/passwd.5.gz (5번 설명서 파일)
#

#man passwd

--->1번 설명서(명령어) 확인

PASSWD(1)

User utilities

PASSWD(1)

NAME

passwd - update a user's authentication tokens(s)

#man 5 passwd

---> 5번 설명서(설정파일) 확인

■매뉴얼의 종류(Section)

1. 사용자 명령 (User Commands)
2. 시스템 호출 (System Calls)
3. C 라이브러리 함수 (C Library functions)
4. 디바이스와 네트워크 인터페이스 (Devices and Network Interfaces)
5. 파일 포맷 (File Formats)
6. 게임과 데모 (Games and Demos)
7. 환경, 테이블, 매크로 (Environments, Tables, and Macros)
8. 시스템 관리 (Maintenance Commands)

<http://man.kldp.net/wiki/ManPage>

1p : POSIX(유닉스 표준) 명령어

■passwd 명령어(password)

- 루트는 일반 계정의 암호를 변경할 수 있다.
- 일반 계정은 자신의 암호만 변경할 수 있다.
- passwd 만 입력하면 현재 로그인한 계정의 암호가 변경된다.
- 루트는 간단한 암호를 사용할 수 있다.
- 일반 계정은 간단한 암호를 사용할 수 없다.

-루트 암호 변경

#passwd 또는 #passwd root

-apple 계정 암호 변경

#passwd apple

---> 암호 입력시 아무것도 안 보임

---> 암호 문자를 입력하고 엔터를 누르면 됨

■인터넷 설정

```
#gedit /etc/sysconfig/network-scripts/ifcfg-eth0
```

--> 리눅스 메모장 프로그램으로 네트워크 설정 파일을 열기

```
ONBOOT=yes
```

로 변경후 저장

```
#ifconfig
```

웹브라우저 실행후 인터넷 확인

저장후 인터넷 연결이 안 되면 아래 명령 실행후 확인

```
#service network restart
```

■실습

1. passwd 명령과 옵션을 이용하여 apple 계정을 락시키고
root 로그아웃 한 후 apple 계정 로그인 안되는지 확인하기

man passwd 또는 passwd --help 이용하여 옵션 확인

2. 다시 root 로그인 후 apple 계정 락 해제시키고 apple 로그인 되는지 확인하기

-계정을 락(Lock) 시키면 락을 해제(Unlock)하기 전까지
로그인 할 수 없다.(인증 실패했다는 메시지 출력)

■경로

1. 절대경로(Absolute Path, 최상위 디렉토리인 / 가 기준)
2. 상대경로(Relative Path, 현재 디렉토리가 기준)

. (점 1개) : 현재 디렉토리를 의미

.. (점 2개) : 상위 디렉토리를 의미

~ (틸드) : 현재 로그인한 사용자의 홈디렉토리를 의미

////////////////////////////////////

경로 비교

C:\users\andy\test.txt

---> 윈도우 경로

/users/andy/test.txt

---> 리눅스 경로

////////////////////////////////////

■기본 명령어

1. pwd

print working directory, 현재 작업 디렉토리의 전체경로 출력

2. clear

화면 지우기

3. cd

change directory, 디렉토리 변경

#cd

디렉토리명을 지정하지 않으면 현재 사용자의 홈디렉토리로 이동된다.

#cd 디렉토리명

디렉토리명은 상대경로나 절대경로를 이용할 수 있다.

경로 연습

#cd /tmp

---> 절대 경로

#pwd

#cd ..

---> 상대 경로

#pwd

#ls

#cd /usr/local

---> 절대 경로


```
#pwd
#cd ../../
---> 상대 경로
#pwd
```

4. ls 또는 dir

list, 디렉토리의 목록을 출력

옵션 : -l, -a, -F, -R, -d

- l(long) 퍼미션, 소유권 등 자세한 정보 출력
- a(all) 점으로 시작하는 숨김속성 파일도 출력
- f 디스크에 저장된 순으로 출력
- F(classify) 파일의 종류를 구분해서 출력
- R(recursive) 하위 디렉토리를 구분해서 출력
- r(reverse) 역순정렬(z - y - x c - b - a)
- d(directory) 경로안의 내용을 출력하지 않고 경로를 출력

```
#cd
#pwd
/root
#ls
#ls -a
#ls -F
---> 폴더(디렉토리)는 마지막에 / , 실행파일은 * 가 붙는다.
    @ 는 심볼릭 링크 파일
#ls -r
#ls -R
#ls /etc
---> 현재 폴더에서 다른 폴더 목록 출력
#ls -d /etc
---> 디렉토리(폴더) 이름만 출력
#ls -dl /etc (또는 ls -ld /etc 또는 ls -l -d /etc)
```

5. mkdir (make directory)

디렉토리 생성하기

옵션)

-p : 계층적으로 디렉토리 만들 때 사용(mkdir -p /a/b/c)

```
#mkdir --help
```

```
#cd
```

```
#pwd
```

```
/root
```

```
#mkdir /work
```

---> 최상위 디렉토리(/) 아래에 work 디렉토리가 생성된다.

mkdir work 로 실행하면 /root/work 디렉토리가 생성된다.

```
      /
      |
root  work
```

```
#cd /work
```

```
#pwd
```

```
/work
```

```
#touch apple1.txt
```

---> touch 는 빈파일(empty file) 을 만드는 명령어

```
#ls
```

```
#touch apple2.txt apple3.txt
```

```
#ls
```

```
#pwd
```

```
/work
```

```
#mkdir /work22
```

mkdir work22 로 하면 현재 디렉토리

즉, /work 안에 work22(/work/work22) 가 만들어진다.

```
      /
      |
work  work22
```

```
#ls /
```

```
#ls /work22
```

비어있음

6. cp

파일이나 디렉토리를 다른 파일 또는 다른 디렉토리로 복사(Copy)
물리적으로 새로운 파일이 생성됨

옵션)

- a : 원본 파일의 속성, 링크 파일 정보를 유지하며 복사(archive)
- b : 파일이 존재할 경우 기존 파일을 백업(backup)
- i : 복사할 대상이 있으면 질문 출력(interactive)
- r : 디렉토리 복사할 때 사용(recursive)
- v : 복사과정 자세히 출력(verbose)
- u : 복사할 대상의 변경 날짜가 같거나 더 최근 것이면 복사하지 않음(update)

사용 예)

```
#cp 파일명1(원본) 파일명2(대상)
#cp 파일명 디렉토리명
#cp -r 디렉토리명1 디렉토리명2
```

▷모든 파일을 복사할 때는 * 를 이용할 수 있다.

*.txt : txt 로 끝난 모든 파일을 의미

▷원본은 여러 개를 지정할 수 있지만

대상은 여러 개 지정 안 됨

```
cp --help
```

```
#cp /etc/hosts /tmp/one.txt
--->파일 이름을 변경하면서 복사
#ls /tmp/*.txt
```

```
#cp -v /etc/hosts /tmp/two.txt
---> /etc 폴더의 hosts 파일을 /tmp 폴더안에 two.txt 로 복사
#ls /tmp/*.txt
```

```
#cp -v /etc/hosts /tmp
--->동일한 파일 이름으로 복사
#ls /tmp/ho*
```

7. mv

move, 이름변경 및 이동

형식)

mv 원본파일(또는 디렉토리)명 대상파일(또는 디렉토리)명
대상파일명이 디렉토리로 존재하면 그 디렉토리 안으로 이동이 되고
존재하지 않으면 이름이 변경된다.

옵션)

- i : 같은 이름의 파일이 있을 때 질문 출력
- f : 동일 파일이 있을 때 확인 질문없이 강제로 덮어쓰기
- v : 실행과정 자세히 출력
- b : 파일이 존재할 경우 기존 파일을 백업(backup)
- u : 복사할 대상의 변경 날짜가 같거나 더 최근 것이면 이동하지 않음(update)

▷디렉토리 복사 및 삭제 : -r(또는 -R) 옵션 사용

▷디렉토리 이동(mv) : -r 옵션 필요 없음

사용예)

```
mv a.txt new.txt
(이름이 변경됨)
mv a.txt b.txt /work
(현재 디렉토리의 a.txt 와 b.txt 가 /work 디렉토리로 이동이 됨)
mv *.txt /work
```

▷파일 이름변경

```
#cd /work
#pwd
/work
#ls
apple3.txt
#mv -v apple3.txt orange.txt
---> 이름 변경
#ls
```

▷이동 테스트

```
#pwd
/work
#ls
#ls /tmp/*.txt
#mv -v orange.txt /tmp
```

--->이동(현재 폴더 /work 의 orange.txt 가 /tmp 디렉토리로 이동 됨)

```
#ls
```

```
#ls /tmp/*.txt
```

orange.txt 가 이동 된 것을 확인

8. rm

remove, 파일 및 디렉토리 삭제

옵션)

-i(interactive) : 확인 질문 출력하는 옵션

-r(recursive) : 디렉토리 삭제시 사용

-f(force) : 확인 질문없이 강제 삭제시 사용

```
#cd /work
```

```
#pwd
```

```
/work
```

```
#rm apple1.txt
```

rm : 파일 삭제

질문 나올 때 엔터 또는 n 을 입력하면 삭제가 안되고

y 를 입력하고 엔터를 쳐야 삭제 된다.

```
#ls
```

파일삭제 확인

```
#rm -f apple2.txt
```

-f //force, 질문없이 강제 삭제

```
#ls
```

--->터미널에서 삭제할 때는 바탕화면의 휴지통으로 들어가지 않으므로
중요한 파일을 지울 때는 주의해야 한다.

--->중요한 파일은 다른 곳(다른 디렉토리, 다른 하드디스크 등)에
백업(저장)을 해 두는 것이 좋다.

9. rmdir

remove directory, 디렉토리 삭제하기

(디렉토리가 비어 있지 않을 때는

삭제되지 않으므로 rm -rf 를 대신 사용하기도 한다.)

```
#cd /work
```

```
#pwd
/work
#touch apple3.txt
#ls
apple3.txt
#touch /work22/apple5.txt
#ls /work22
```

```
#rmdir /work22
---> 디렉토리가 비어 있지 않으면
      삭제되지 않는다.
```

```
#mkdir /work23
#ls /
#ls /work23
#rmdir /work23
#ls /
```

```
#rm /work22
rm: cannot remove directory `/work22': 디렉토리입니다
#
```

```
#rm -r /work22
---> 비어 있지 않은 디렉토리를 지우려면 rm -r 을 이용할 수 있으나 확인 질문이 출력된
다.
```

```
#rm -rf /work22
(rm -r -f /work22 와 동일)
---> 질문없이 바로 삭제 : 사용할 때 주의
```

```
#ls /
```

(주의 : rm -rf / 하면 새로 설치해야 한다.)

////////////////////////////////////

▷ascend

[동사](격식)오르다, 올라가다

The path started to ascend more steeply.

길이 더 가파르게 올라가기 시작했다.

▷descend

[동사](격식)내려오다, 내려가다

The plane began to descend.

비행기가 하강하기 시작했다.

////////////////////////////////////

10. cat

파일의 내용을 화면에 출력

옵션)

-n : 라인 번호를 표시

-e : -vE 와 같음

-v : 개행문자, 탭키를 제외한 제어문자를 ^(캐럿) 형태로 표시

-E : 각 행의 끝에 \$ 표시

-T : tab 키를 ^I 로 출력

-A : -vET 옵션과 동일

사용 예)

#cat 파일명1

#cat 파일명1 파일명2 ...

#cat /etc/hosts

---> /etc 는 폴더명, hosts 는 파일명

#cat -n /etc/hosts

#cat -A /etc/hosts

```
#cat /etc/hosts.allow  
#cat -A /etc/hosts.allow
```

■실습

1. /down 디렉토리를 생성하시오
2. /down 디렉토리에 test1.txt , test2.txt 파일을 생성하시오
3. /down 디렉토리에 있는 test1.txt , test2.txt 파일을
각각 test3.txt , test4.txt 파일로 /tmp 디렉토리에 복사하시오
4. /backup 디렉토리 생성후 /boot 디렉토리를 /backup 으로 복사
5. /usr/sbin 폴더에서 group 으로 시작하는 파일만 /backup 디렉토리로 복사

```
////////////////////////////////////  
#mkdir /down  
#cd /down  
#pwd  
/down  
#touch test1.txt test2.txt  
#cp -v test1.txt /tmp/test3.txt  
#cp -v test2.txt /tmp/test4.txt  
#ls /tmp/*.txt  
#mkdir /backup  
#cp -r /boot /backup  
#ls /backup  
#cp /usr/sbin/group* /backup
```

```
////////////////////////////////////
```


[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(3일차)

VI(편집기)

Shell(셸)

■vi

vi(visual editor) :

- 리눅스나 유닉스에서 일반적으로 사용하는 텍스트 편집기
- 자체 윈도우가 있는 것이 아니라 터미널에서 실행 됨
- 메뉴가 없다.

☞vi 는 세가지 상태(모드,Mode)가 있다.

- 1) 명령상태 또는 esc 상태
- 2) 입력상태 또는 편집 상태
- 3) ex상태 또는 :(콜론) 상태 또는 실행 상태

#cd

#pwd

/root

#vi

vi 를 실행하면 처음상태가 명령상태이다. 키보드 입력이 하나의 명령으로 동작한다.

i(insert) 나 a(append) 를 누르면 입력상태로 전환된다.

왼쪽하단에 -끼워넣기- 또는 -INSERT- 로 표시됨

다시 명령상태로 전환하려면 esc 키를 누르면 된다.

ex 상태는 콜론(:) 으로 시작하는 상태를 말한다.

:set nu ---> 라인번호 출력

:set nonu ---> 라인번호 해제

처음 vi 를 실행하면 명령상태 이므로 아래 문자키를 입력하여
입력상태로 만들고 입력을 하게 된다.

☞입력상태 설명

i // 현재 커서 앞쪽에 입력
a // 현재 커서 뒤쪽에 입력
I // 현재 라인 처음에 입력(라인이 공백으로 시작하면 실제 단어 앞에 입력 됨)
A // 현재 라인 마지막에 입력
o // 현재 라인 아래에 새로운 줄을 만들고 입력
O // 현재 라인 위쪽에 새로운 줄을 만들고 입력
s // 현재 커서의 글자 삭제후 입력
S // 현재 라인 삭제후 입력

명령상태로 바꾸려면 esc 키를 누르면 된다.

(왼쪽 하단에 -끼워넣기- 또는 -INSERT- 가 표시되면 입력상태,
없으면 명령상태이다.)

☞삭제(esc 키를 눌러서 명령상태에서 입력하는 것임)

x 또는 delete 키 // 현재 커서 문자 한자 삭제
X // 현재 커서 앞의 글자 삭제
dd // 한 줄 삭제
숫자 + dd // 현재 커서부터 숫자만큼 행 삭제
d\$ // 현재 커서 부터 현재 라인의 마지막까지 삭제
(\$는 마지막을 의미)
d0 // 현재 커서 앞 문자부터 라인의 처음부분까지 삭제
(숫자 0 은 처음을 의미)
d^ // 현재 커서 앞 문자부터 라인의 처음부분까지 삭제
(^ 는 처음을 의미)
dw // 단어 삭제(단어의 첫번째 글자에 커서를 위치 시켜야 함)
u // undo, 작업 취소

☞교체(Replace)

r // 한 자 교체
R // 여러 글자 교체
cw // 한 단어 교체(단어의 첫번째 철자에 커서를 위치시켜야 함)
cc // 한 줄 교체(Change entire line)

☞저장 및 종료 또는 저장하지 않고 종료

명령상태로 바꾼다음(esc 누르기) 아래 명령을 입력

- 1) :q (저장하지 않고 종료, q ---> quit)
- 2) :w (파일 저장)
- 3) :wq (저장후 종료, w ---> write, q ---> quit)

vi 실행시 파일명을 지정하지 않았으면
리눅스 명령어 상태로 나올 때 저장할 파일명을 입력해야 한다.
:wq 파일명

- 4) :q! (저장하지 않고 강제 종료)
 - 5) :wq! (저장후 강제종료)
- >리눅스는 대소문자를 구분함

vi 실행된 상태에서
esc 눌러서 명령 상태 만들고 저장후 빠져 나오기
:wq
---> 파일명이 없다고 출력 됨(No file name)
:wq test.txt
--->현재 폴더에 test.txt 로 저장후 종료

#ls

#cat test.txt

#cp /etc/services /root/test2.txt

---> /etc 폴더 안의 services 파일을 /root 폴더 안에 test2.txt 로 복사

#vi test2.txt

:set nu

☞ 문자키를 이용한 커서 이동하기(명령상태에서 하기)

	k(위)	
h(왼쪽)		l(오른쪽)
	j(아래)	

0(숫자 영) //행(라인)의 처음으로 이동

\$ //행(라인)의 마지막으로 이동

G //문서의 마지막으로 이동
gg //문서의 처음으로 이동

33번 라인으로 이동하기
---> 33G 또는 :33 엔터

PgUP, PgDN //페이지 이동

🔍검색

1)위에서 아래로 검색
/검색단어(명령상태에서 / 누르고 검색단어 입력후 엔터)
n 을 누르면 다음 단어로 이동된다.
N 을 누르면 이전 단어로 이동

2)아래에서 위로 검색
?검색단어
n 을 누르면 위 방향으로 계속 찾기
N 을 누르면 아래 방향으로 계속 찾기

📄복사 및 붙여넣기

한줄 복사 //yy (yank)
숫자 + yy //현재 커서부터 숫자 만큼의 행을 복사
붙여넣기 //p (paste) ---> 현재 라인 아래 줄에 붙여넣기
//P(대문자) ----> 현재 라인 윗 줄에 붙여 넣기

🔧ex 모드

1)vi 상태에서 리눅스 명령어 실행
:!명령어

:!cal

:.!cal

---> 현재 커서 위치에 리눅스 명령어 결과를 가져오기

2)색상 적용하기

:color evening

:color morning

:color default

!!ls /usr/share/vim/vim72/colors/

---> 적용할 수 있는 색상 단어 확인(VI 상태)

#ls /usr/share/vim/vim72/colors/

---> 적용할 수 있는 색상 단어 확인(터미널 상태)

vim 은 vi 를 개선한 프로그램(Vi Improved)

#vim test.txt

3)split(화면 분할)

:split(수평 분할, 원래 파일 화면)

:vs(수직 분할, 원래 파일 화면)

ctrl + ww ---> 창간의 이동

ctrl + wn ---> 수평분할(빈 화면)

ctrl + wv ---> 수직분할(빈 화면)

:e 파일명 ---> 지정한 파일 열기

:enew ---> 현재 문서를 닫고 새로운 빈문서 열기

🔍vi 치환

s ---> search

:1,10s/원래단어/바꿀단어 ---> 1번 라인에서 10번 라인까지 변경

:%s/원래단어/바꿀단어 ---> 파일 전체에서 첫번째 단어만 변경

:%s/원래단어/바꿀단어/g ---> 파일 전체에서 한 줄의 모든 단어 변경(g 는 global 을 의미)

#vim linux.txt

---> vim 저장할파일명

---> ~ 표시는 라인에 아무 내용도 없다는 의미
---> 아래와 같이 입력하기

```
apple  
apple apple apple
```

```
orange  
orange orange orange
```

```
mango  
mango mango mango
```

위와 같이 입력후
:%s/apple/tico (명령상태에서 입력후 엔터-결과 확인)
:%s/orange/matiz/g

```
:set nohls  
---> 검색 단어 강조해제
```

```
:set hls  
---> 검색 단어 강조
```

👉 VI 설정 파일 사용하기

vi 설정파일 ---> .vimrc
로그인한 계정의 홈디렉토리에 존재해야 한다.
루트의 경우에는 /root 에 존재해야 한다.

```
#cd  
#pwd  
/root  
#vi .vimrc  
set nu  
color evening  
#vi test2.txt  
라인번호와 색상이 적용되는지 확인
```

■ 실습

1. /root 디렉토리로 이동하기(#cd /root)
2. vim 이용하여 아래와 같이 입력하고 치환 기능 이용하여 치환 결과 처럼 치환하기
파일명 : test3.txt (#vim test3.txt)

입력할 내용)

test.com

test.com test.com

test.com test.com test.com

치환 결과)

yahoo.com

yahoo.com yahoo.com

yahoo.com yahoo.com yahoo.com

3. vim 이용하여 test.html 파일 생성하기
(동일한 내용은 복사해서 붙여넣기 기능 사용)

//파일명 : test.html

<html>

<body>

<h1>Linux 1</h1>

<h2>Linux 2</h2>

<h3>Linux 3</h3>

<input type=text>

<input type=text>

<input type=password>

<input type=radio>

<input type=radio>

<input type=radio>

<input type=checkbox>

<input type=checkbox>

<input type=checkbox>

<input type=checkbox>

<hr color=red>

VI TEST

</body>

</html>

■리눅스 기본 명령어

1. which

알리아스, 실행파일 경로(위치) 출력

```
#which ls
```

2. alias

명령어 별칭 출력 및 생성(반대는 unalias)

라우터에서 사용하는 것과 유사

```
#ls -l
```

```
#ll
```

---> 명령어 alias , ll 은 ls -l 과 같다.

```
#alias
```

---> 미리 설정되어 있는 alias 확인하기

3. echo

화면상에 문자열이나 변수의 값(내용)을 출력

변수의 값을 출력시킬 때는 \$ 기호로 시작한다.

```
#echo test
```

```
#echo TEST
```

```
#echo $HOME
```

4. export

환경변수로 만들어 주는 명령어

■Unix 및 Linux 구조

사용자

Shell

Kernel

H/W

컴퓨터 본체-----커널-----셸-----사용자

■용어 정리

1. Kernel(커널)

운영체제의 핵심되는 프로그램(자동차의 엔진 역할)

2. 셸(shell)

-명령어 해석기(Command Interpreter)

-사용자가 입력한 명령어를 커널(운영체제의 핵심프로그램)이 이해할 수 있도록 번역해서 전달하고 결과를 사용자에게 리턴하는 프로그램

-셸도 하나의 프로그램이며 리눅스로 로그인하여 터미널을 실행하면 셸이 메모리에 로딩된다.

-원격 접속을 해도 셸이 메모리로 로딩된다.

-터미널에서 텍스트 명령어를 사용하면 셸을 이용하게 되는 것이다.

-명령어 대기 상태 : 셸프롬프트 라고 부른다.

-리눅스 명령어를 셸명령어 라고도 부른다.

■셸변수

shell(셸)

kernel(커널)

리눅스로 로그인 한 후 터미널 실행했을 때 메모리 상태

////////////////////////////////////

▷셸 : bash ksh csh

▷컴퓨터 바이러스 백신 프로그램 : 알약 V3 네이버백신

////////////////////////////////////

☞변수 : 프로그램(또는 명령어)이 실행될 때 메모리(기억장치)에
데이타(자료)를 저장하기 위해 그릇과 같은 임시 기억 장소가
만들어 지는데 이를 변수(variable) 라고 한다.

☞환경변수 : 현재 셸에서 또 다른 셸이 실행되었을 때
상속되는 변수가 환경변수(외부변수)이다.
일반 변수(지역변수)는 상속되지 않는다.

※변수정리

변수(지역변수, 로컬변수) : 현재 셸에서만 사용가능한 변수

환경변수(외부변수, 전역변수) : 현재 셸에서 다른 셸을 실행시켰을 때 상속되는 변수

☞변수확인

1)set, declare

---> 변수와 환경변수 모두 출력

형식)

변수이름=값

2)env, printenv

---> 환경변수 만 출력

☞변수생성

변수이름=값

--->변수 이름은 숫자로 시작하면 안되고 영문 또는 _ 만 시작가능

--->중간에 공백이 있으면 안 됨

--->데이타 타입은 지정할 필요 없음

--->변수를 출력 할 때는 echo 명령어를 사용하고 변수이름 앞에 \$ 기호 사용

☞변수삭제

unset 변수명

```
#cd /work
#echo $HOME
#unset HOME
#echo $HOME
#cd
---> ?
```

☞ 주요 환경변수

```
-----
[/a:/b:/c]      [ ko_KR ]    [ /root ]
PATH 변수      LANG 변수    HOME 변수
-----
```

메모리상의 셸 영역

셸의 메모리 영역에는 변수도 있고 환경변수도 있는데
PATH 가 대표적인 환경변수이다.(--->대문자)

※POSIX : 유닉스 표준을 의미
<http://ko.wikipedia.org/wiki/POSIX>

--->환경변수는 POSIX 표준에 따라서 대문자

PATH(패스) : 명령어를 탐색하는 경로(디렉토리,폴더) 정보를 저장하고 있는 환경변수
LANG : 언어 정보 저장
HOME : 로그인한 계정의 홈디렉토리 저장

```
#echo $PATH
```

--->명령어가 실행될 때 셸에 의해서 패스(PATH) 같은 환경변수가 참조된다.
--->#ls 라고 입력했을 때 실제로는 /bin/ls 실행이 되는데
셸이 메모리상에서 PATH 변수를 탐색해서 실행을 해 주므로
/bin/ls 라고 입력할 필요가 없다.

```
#cat /etc/shells
---> 사용 가능한 셸
```

※알리아스와 환경변수를 터미널에서 추가한 경우 메모리에 저장되므로

리부팅하면 사용할 수가 없음

새로 정의한 알리아스와 환경변수를 계속 사용하려면 쉘설정파일에 등록해 주면 됨

-알리아스는 명령어와 옵션을 을 짧게 지정하여 사용하는 명령어 별칭 기능

-변수는 알리아스 처럼 명령어를 저장할 수도 있고 데이터를 저장하여 사용할 수 도 있다.

■셸설정파일

▷ bash

--->리눅스의 디폴트 쉘 프로그램인 배쉬

▷리눅스는 로그인을 하면 메모리에 쉘이 실행 되는데

/etc/profile, /etc/bashrc 을 먼저 읽고 자신의

홈디렉토리에서 .bashrc, .bash_profile 을 읽어서 알리아스,

패스(PATH)같은 환경변수를 설정하게 된다.

▷/etc/bashrc, .bashrc 는 터미널을 실행할 때 마다 읽고

/etc/profile, .bash_profile 로그인시 한 번 만 읽는다.

▷/etc 에 있는 쉘설정파일은 모든 사용자에게 적용되는 전체설정 파일이다.

(/etc/profile, /etc/bashrc)

▷bash셸 사용자 설정파일

1) .bashrc (비-로그인 쉘 설정파일) : 주로 알리아스 등록

2) .bash_profile (로그인 쉘 설정파일) : PATH 같은 환경변수 등록

3) .bash_history : 명령어 기록 파일

4) .bash_logout : 로그아웃시 실행할 명령 기록

--->계정의 홈디렉토리에 존재한다.

--->루트의 경우에는 /root 안에 쉘설정파일이 존재

☞셸설정 파일 수정 테스트

#cd

#pwd

/root

```
#vi .bashrc  
마지막 부분에 추가
```

```
alias l='clear'  
alias vi='vim'  
alias shut='shutdown -h now'
```

터미널을 종료후 다시 실행하면 적용 됨
(또는 source 명령 사용)

```
#cd  
#pwd  
/root  
#vi .bash_profile  
마지막에 추가
```

```
net='cd /etc/sysconfig/network-scripts/'
```

```
#source /root/.bash_profile  
#$net  
---> 변수 치환이라 함  
#pwd
```

▷변수치환
#C=cal
#\$C

■원격 접속 사용하기

[윈도우 7]-----가상 네트워크-----[리눅스 가상머신]

Putty, CRT : 원격 접속 터미널 프로그램

리눅스 아이피 확인 : ifconfig
현재 접속자 확인 : who
현재 사용자 확인 : whoami , who am i

▷Putty 사용하기

■실습

1. touch 명령어를 MYTOUCH 변수에 대입
2. /work/linuxtest.txt 를 DEST 변수에 대입
3. 만들어진 변수를 이용하여 /work/linuxtest.txt 파일 생성하기
4. shutdown -r now 를 shutr 으로 알리아스 설정하기
5. shutr 입력하여 테스트
6. #en 을 입력하면 영문으로 설정되도록 알리아스 설정
7. Putty 이용하여 apple 로그인하고 PATH 변수에 /bin , /usr/bin 폴더만 설정(ifconfig 실행 확인)

※BASH 셸 환경변수

HISTFILE // 명령어 히스토리를 저장할 파일 이름, 기본값은 ~/.bash_history
PWD // 현재 작업 디렉토리
SHELL // 현재 사용되어지고 있는 셸
UID // 현재 사용자의 UID(유저 번호)

////////////////////////////////////

#MYTOUCH=touch

또는

#which touch

/bin/touch

#MYTOUCH=/bin/touch

#mkdir /work

#DEST=/work/linuxtest.txt

---> 변수 생성할 때는 \$ 기호 가 필요 없음

#\$MYTOUCH \$DEST

--->#touch /work/linuxtest.txt

#ls /work

#alias shutr='shutdown -r now'

#shutr

#alias en='LANG=en_US.UTF-8'

#en

#cp

apple 로그인

[apple@linux1 ~]\$ PATH=/bin:/usr/bin

[apple@linux1 ~]\$ ifconfig

////////////////////////////////////

[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(4일차)

쿼팅(Quoting)

리다이렉션(Redirection)

grep

<리눅스 기본 명령어>

head

파일의 앞쪽 부분 10줄 출력

옵션 : -숫자(출력 행수 지정)

```
#head /etc/passwd
--> /etc 는 폴더(디렉토리), passwd 파일
```

```
#head -3 /etc/passwd
```

tail

파일의 뒷쪽 부분 10줄 출력

옵션)

-숫자(출력 행수 지정)

```
#tail /etc/passwd
#tail -3 /etc/passwd
```

more

파일의 내용을 한페이지씩 확인 할 때

PageUp, PageDown, 화살표키 사용 불가

```
#more /etc/services
한 페이지씩 출력
space bar // next page
q // 종료
```

```
#more -d /etc/services
```


-d // 디스플레이 옵션

f // 다음 페이지(next page)
b // 이전 페이지(previous page)
엔터 // 다음 라인(next line)

 file

대상 파일의 종류(type)를 출력

```
#file /bin/ls
#file /etc/hosts
```

 rdate

외부 타임서버와 시스템의 시간을 맞출 수 있다.

```
#rdate -p time.bora.net
```

타임서버의 시간출력

```
#rdate -s time.bora.net
```

자신의 컴퓨터 시간을 타임서버와 동기화

```
#date
```

<쿼팅(Quoting)>

작은따옴표, 큰따옴표, \ 사용하는 것을 쿼팅 이라 한다.

셸이 특별하게 생각하는 문자의 의미를 제거할 때 사용한다.

- 1) ' 작은따옴표
- 2) " 큰따옴표
- 3) \ 역슬래쉬(해석금지의 의미)

---> 변수의 경우 '(작은따옴표) 는 \$ 표시와 변수이름 자체가 출력되고
" 는 변수의 값이 출력된다.

---> 작은 따옴표와 큰따옴표를 섞어서 사용하면 안 됨

---> 문자열에 공백이 들어간 경우에도 쿼팅을 사용할 수 있다.

```
#echo '$LANG'
---> 변수이름 출력(변수 해석 안 함)
```

```
#echo "$LANG"
---> 변수 값 출력(변수 해석 함)
```

```
#echo $LANG
---> 변수 값 출력
```

```
#echo \ $LANG
---> 변수 해석(또는 변수치환) 금지
```

☞ 명령어 치환
` 역따옴표 : 1번키 왼쪽키, Back Quote
`(역따옴표) 는 명령어의 결과를 변수에 저장할 때 쓰인다.

```
#temp=`date`
---> 역따옴표 사용
---> = 사이에 공백이 있으면 안 됨
---> date 명령어의 결과(return 값)가 메모리상의 temp 라는 변수에 저장 됨
```

```
#echo $temp
```

```
#temp='date'
---> 작은따옴표
```

또는

```
#temp="date"
---> date 라는 문자열이 변수에 저장 됨
```

```
#echo $temp
```

☞ 톨드(~) 치환
#echo ~
홈디렉토리 출력

#cd ~사용자계정명
지정한 사용자의 홈디렉토리로 이동

#cd ~- 또는 #cd -
바로 이전의 작업 디렉토리로 이동

<치환 정리>

1. 변수치환(\$ 사용)
2. 명령어 치환(` 사용)
3. 틸드 치환(~ 사용)

""(큰따옴표) 만 변수치환, 명령어 치환을 허용함

▷find 명령어

1. find 형식

find <탐색을 시작할 경로> [옵션]

파일이 어느 경로에 있는지 모를 때는 / 를 사용하면 된다.

최상위 부터 검색하므로 시간이 많이 걸린다.

탐색 경로를 생략하면 현재 디렉토리(폴더)를 기준으로 찾는다.

2. find 옵션

-name // 이름으로 검색
-empty // empty 파일 검색
-uid UID // UID(사용자 번호)로 검색
-gid GID // GID(그룹 번호)로 검색
-perm // 퍼미션으로 검색
-user // 소유자로 검색
-group // 그룹으로 검색
-exec // 찾은 파일을 대상으로 명령어 실행
-ok // exec 와 비슷하나 질문 출력
-atime -3 // Access time(접근 시간)이 3일 보다 작은 파일
-atime +3 // Access time(접근 시간)이 3일 보다 큰 파일
-newer hello.c // 지정한 파일(hello.c) 이후에 수정된 파일 검색

#mkdir /work5

```
#cd /work5
#touch test1.txt test2.txt cent.dat nice.c
#ls
```

```
#find /work5 -name *
---> X (실행 안 됨)
#find /work5 -name *.txt
---> X (실행 안 됨)
---> /work5 부터 txt 로 끝난 모든 파일을 찾으라는 의미
```

```
#find /work5 -name "*"
---> O
#find /work5 -name '*'
---> O
#find /work5 -name \*
---> O
```

※셸명령어 사용시 * 나 공백이 들어간 경우
셸이 특별하게 생각하기 때문에 결과가 올바르게
나오지 않을 수 있다.
이 때는 쿼팅을 사용하면 된다.

```
#man find
```

```
////////////////////////////////////
NON-BUGS
```

```
$ find . -name *.c -print
find: paths must precede expression
Usage: find [-H] [-L] [-P] [path...] [expression]
```

This happens because *.c has been expanded by the shell resulting in
find actually receiving a command line like this:

```
find . -name bigram.c code.c frcode.c locate.c -print
```

That command is of course not going to work. Instead of doing things
this way, you should enclose the pattern in quotes:

```
$ find . -name '*.c' -print
```

```
////////////////////////////////////
```

☞ 파일을 찾아서 지우는 예제

```
#mkdir /work7
```

```
#cd /work7
```

```
#touch user1.txt user2.txt cent.dat
```

```
#find /work7 -name cent.dat -exec rm {} \;
```

---> 질문없이 바로 삭제 된다.

{ } 는 찾은 파일을 의미하고

\; 는 명령어가 끝났음을 의미한다.

{ } 와 \; 는 공백없이 타이핑

※세미콜론(:)의 의미

명령어를 연속적으로 실행시킬 때 사용하는 기호

```
#date ; cal ; ls
```

<Redirection(방향 재지정)>

1) > 출력(기존의 파일이 있으면 overwrite 됨)

2) >> append(기존내용에 추가하기)

3) < 입력

4) << 입력 종료 문자 지정

리눅스의 내부 통로

리눅스 운영체제 내부에는 데이터가 이동하는 3가지 통로가 있다.

표준입력(Standard Input, stdin) ---> 0 : 키보드와 연결

표준출력(Standard Output, stdout) ---> 1 : 모니터와 연결

표준에러(Standard Error, stderr) ---> 2 : 모니터와 연결

```
Standard Input      0    <----- 키보드
-----

-----
Standard Output    1    -----+
-----+
|
|
+----> 모니터
|
-----
Standard Error     2    -----+
-----
```

```
#cd /work
#pwd
/work
#vi result.txt
```

redirection test

```
#cat result.txt
#cat < result.txt
---> cat result.txt 와 동일하다
```

```
#cat
---> 입력 리다이렉션(< 기호)을 사용하지 않았으므로
      키보드로부터 입력을 받으려고 커서가
      다음 줄로 넘어간다.
      ctrl + d 를 누르면 종료가 된다.
```

```
Ctrl + d // 입력 종료
Ctrl + c // 명령실행 중단(인터럽트)
```

```
#cat > result.txt (키보드 입력을 받아 파일로 출력하라는 의미)
one
two
three
ctrl + d : 입력종료 단축키
```

```
#ls
#cat result.txt
```

덮어쓰기(Overwrite)가 된 것을 확인할 수 있다.

```
#cat >> result.txt
four
five
six
ctrl + d(타이핑 하는 것이 아니고
ctrl 키 누른 상태에서
d 키를 누르라는 의미)
```

```
#cat result.txt
```

추가가 된 것을 확인할 수 있다.

```
#cat > result.txt << EOF
kor
eng
math
eof
EOF(입력 종료 문자로 동작함)
#cat result.txt
```

```
#cat << C > test_file
```

지정된 문자열 "C" 가 입력되면 입력 중지

```
#cp
#cp > error.txt
#cat error.txt
```

```
#cp 2> error.txt
#cat error.txt
```

▷파이프(Pipe, |)

리눅스 명령어는 파이프(|)를 이용하여 여러 개를 연결해서 사용할 수 있다.
| ---> 백스페이스(←) 왼쪽키

#명령어1 | 명령어2 | 명령어3 ...

파이프(|) : 리눅스 명령어들을 연결해서 사용할 수 있는 기호

파이프를 이용하면 한 명령어의 출력을 다른 명령어의
입력으로 사용하여 여러 명령어가 적용된 출력 결과를
얻어낼 수 있다.

A | B | C ...

▷sort

정렬 명령

-r 은 역순 정렬 옵션(reverse)

#cd /work

#cat > sort.txt

mango

apple

candy

banana

5

3

Ctrl + d

#cat sort.txt

#sort sort.txt

#sort -r sort.txt

#cat /etc/passwd | sort -r | more

////////////////////////////////////

#cat /etc/hosts

#cat /etc/hosts | vi

모든 명령어가 파이프를 사용할 수 있는 것이 아니다.

#cat /etc/hosts | vi -

esc :q!

#man vi

- stdin

////////////////////////////////////

※ 파이프를 사용할 수 있는 명령어는 grep, more, sort 등 한정되어 있다.

▷용어정리

1. 메타

근원적인 정의, 일련의 규칙

2. 메타문자(Metacharacters)

^ 시작을 의미, 또는 대괄호([]) 안에서는 except(제외) 를 의미

\$ 줄의 끝을 의미

. 한 개의 문자를 의미

* 바로 앞의 문자(단어가 아님)가 0개 또는 반복을 의미

[] 문자를 집합으로 묶어 줌

\ 특수 문자를 원래의 문자 의미 그대로 해석하게 해 줌

A\{숫자\} 문자 A 를 숫자 만큼 반복

A\{m,n\} 문자 A 를 m번에서 n번 반복

3. 정규표현식

어떤 문자열의 집합을 묘사하는데 사용되는 텍스트 스트링

특정 패턴을 찾는 규칙을 정규표현식(Regular Expression)이라 한다.

[] 범위를 지정, [a-z]* ----> 소문자

[^] ^ 다음의 문자를 제외(except)하는 의미,

[^a-c] ----> a 에서 c 사이의 문자를 제외한 모든 문자를 나타냄

[0-9] 숫자를 의미

▷grep 명령어(필터링)

파일내에서 특정 패턴을 검색하여 그 패턴을 포함하는 모든 라인을
화면에 출력하는 명령어

🔖grep 형식

1)

grep 검색단어 파일명

또는

2)

명령어 | grep 검색단어

#cd

#pwd

/root

#vi meta.txt

113

1133

11333

13

134

1334

13334

1345

14

x

y

c

d

B

F

Linux

*

사용 형식)

grep "정규표현식" 파일명

grep '정규표현식' 파일명
grep 정규표현식 파일명
--->예제에 따라 따옴표를 생략 가능한 경우도 있고
따옴표를 사용해야 결과가 정확히 나오는 경우도 있다.

#grep "13." meta.txt
--->점 자리에는 한 개의 문자가 반드시 있어야 함
--->13 은 출력 안 됨

#grep "[xyz]" meta.txt
--->x,y,z 중에 한 글자와 일치

#grep "[a-z0-9]" meta.txt
--->소문자 혹은 숫자와 일치

#grep "[^b-d]" meta.txt
--->b 에서 d 사이의 문자를 제외한 모든 문자를 나타냄

#grep "[b-d]" meta.txt
---> b 또는 c 또는 d 가 포함된 패턴 매칭

#grep "^L" meta.txt
---> L 로 시작되는 패턴

#grep "1\{2\}" meta.txt
---> 숫자 1 이 2번 반복되는 패턴 매칭

#grep "1\{1,2\}" meta.txt
---> 숫자 1 이 1번에서 2번 반복되는 패턴 매칭

#grep Telnet /etc/services
---> grep 은 대소문자를 정확하게 구분 함

#grep -i Telnet /etc/services
-i // ignore-case (대소문자 구분 안함)

#grep -n skytelnet /etc/services
-n // line-number 출력

```
#grep -c skytelnet /etc/services  
-c // count (검색된 라인 갯수)
```

■파일 다운

```
#yum -y install ftp  
---> 프로그램 설치
```

```
#ftp 아이피
```

Name : 아이디 입력

Password : 암호 입력(입력시 안 보이므로 정확히 타이핑하고 엔터를 누르면 된다.)

Login successful

```
ftp>ls
```

---> 서버쪽 파일 목록 확인

```
ftp>cd 120.UPLOAD2
```

---> 서버쪽 폴더 이동

```
ftp>pwd
```

---> 서버쪽 경로 이동

```
ftp>ls ph*
```

```
ftp>get phone.txt
```

또는 mget ph*

---> 파일 다운로드

```
ftp>by
```

---> ftp 접속 종료

```
#ls
```

▷Putty 설정 변경하기

1. putty 실행하기

putty 실행된 화면에서 아래와 같이 입력

Host Name : x.x.x.x (리눅스 아이피를 적는다.)

Port : 22

Connection Type(Protocol) : SSH 선택

Saved Sessions : WeekendLinux1AM10

Save 버튼 누르기

2. 왼쪽 메뉴에서 Window-Appearance 선택

Font settings 항목의 Change 버튼 누르고 글꼴 선택

3. Window-Translation 선택
Remote character set 을 UTF-8 로 선택
---> 한글 설정
4. Window-Colours
Default Foreground(글자색) 선택후 Modify 버튼 눌러서
원하는 색상 선택
Default Background(배경색) 선택후 Modify 버튼 눌러서
원하는 색상 선택
5. 왼쪽 메뉴에서 Session 선택후 Save 버튼 누르기
6. Open 버튼 눌러서 루트 로그인 테스트 하기

■실습

핸드폰(집전화) 번호 패턴을 정규표현을 통해 정의

1. 지역번호 및 통신사 번호는 '0' 으로 시작하는 2 혹은 3자리 숫자(02,032,010)
2. 중간자리 번호 패턴은 3자리 혹은 4자리가 반드시 한 번은 나와야 한다
3. 끝자리 번호 패턴은 4자리가 반드시 한 번은 나와야 한다

grep 정규표현식 phone.txt

phone.txt 파일이 없을 경우 수업용 FTP 서버 120.UPLOAD2 폴더에서 다운받아 사용

4. /etc/hosts 파일을 입력받아 /work/mydata.txt 파일로 출력하기(리다이렉션 이용)
5. /AA BB 폴더 생성하고 그 폴더로 이동 테스트(쿼팅 이용)

```
////////////////////////////////////  
#grep "^0[0-9]\{1,2\}-[0-9]\{3,4\}-[0-9]\{4\}$" phone.txt
```

```
#cat < /etc/hosts > /work/mydata.txt  
또는  
#cat > /work/mydata.txt < /etc/hosts
```

```
#mkdir "/AA BB"  
#ls /  
#cd "/AA BB"
```

```
////////////////////////////////////
```

▷awk

패턴을 스캐닝하여 처리하는 프로그램 언어

형식)

awk '패턴' 파일명

awk '{액션}' 파일명

awk '패턴{액션}' 파일명

print ---> awk 에서 사용하는 출력함수

예)awk '{print "문자열" \$1}' 파일명

문자열은 큰따옴표로 감싸 주어야 하고

\$1 을 "\$1" 와 같이 큰따옴표로 감싸주면 변수처리 되지 않고 문자열로 처리 됨

서울 apple

부산 mango

인천 candy

\$1 \$2

\$1 은 첫번째 항목

\$2 는 두번째 항목 을 의미

\$0 는 행전체

////////////////////////////////////

#man awk

GAWK(1)

Utility Commands

GAWK(1)

NAME

gawk - pattern scanning and processing language

////////////////////////////////////

☞awk 예제

#awk '/root/' /etc/passwd

---> grep 처럼 사용한 예

---> 검색 패턴 /root/ 에서 /root 로 검색하면 에러 남

```
#awk -F : '/root/{print $1, $2}' /etc/passwd
```

-F : 는 항목 구분자를 지정(공통문자를 지정하는 옵션)

-F 생략시 공백이나 탭키를 한 라인에서 항목구분자로 사용

root 간 들어간 행을 검색한 후

:(콜론) 으로 구분하여

첫번째 항목과 두번째 항목만 출력시키는 예제

a:b:c ---> 항목 구분자는 :

a-b-c ---> 항목 구분자는 -

```
#cd
```

```
#pwd
```

```
/root
```

데이타 파일 작성)

```
#vi employees
```

name	age	address	salary
bill	20	Seoul	1000
Billy	21	Pusan	2000
Jane	22	Daegu	3000
Cindy	23	Masan	4000

---> 입력시 탭키 한 번으로 항목 구분

```
#cat employees
```

```
#cat -A employees
```

---> ^I 는 탭키 한 번을 의미, \$ 는 라인의 마지막을 의미

```
#awk '{print NR, $1, $3}' employees
```

NR : 하나의 레코드(행)를 처리한 후 1이 증가하는 변수

소문자 nr 은 안 됨

```
#awk '{print $1, $3, NF}' employees
```

NF : 필드(라인의 각 항목)의 갯수 출력시키는 변수

```
#awk -F '\t' '$1 ~ /[bB]ill/' employees
```

---> 첫번째 항목(필드,field)에 bill 또는 Bill 이 존재하는지 검사(~)

-F '\t' 는 항목 구분을 탭키로 구분하라는 의미(대괄호 생략 가능)

(-F 옵션 생략시 공백이나 탭키로 항목을 구분 함)

-F ---> field separator 지정 옵션

~ ---> 패턴 매칭 연산자

```
#awk '$1 !~ /ly$/' employees
```

---> 첫번째 필드(항목)가 ly 로 끝나지 않은(!~) 행들

응용 예)

df // disk free, 하드디스크 파티션별 용량 확인 명령

-h 킬로바이트, 메가바이트, 기가바이트로 출력 시키는 옵션

```
#df -h | sort | head -2 | awk '{print $6, $5}'
```

▷/dev/null 특수 파일

```
#man null
```

```
#echo hi
```

```
#echo hi > /dev/null
```

■실습

1. /doc 디렉토리 생성하기

2. 리다이렉션 이용하여 /work 디렉토리에 menu.txt 파일 생성하기

menu.txt 의 내용)

apple

orange

mango

3. find 명령 이용하여 최상위 디렉토리(/) 부터 검색하여

menu.txt 를 찾은 후 /doc/menu2.txt 로 복사하기

4. Putty 이용하여 apple 계정으로 로그인하기

5. find 이용하여 / 부터 menu.txt 파일을 찾아서

apple 홈디렉토리(/home/apple)에 find.txt 로 저장하고

에러메시지(허가 거부 됨 메시지)는 /dev/null 로 보내기

(1번 ~ 3번 은 루트 계정으로 실습)

(5번은 apple 계정으로 로그인해서 실습)

////////////////////////////////////


```
#mkdir /doc
#cat > /work/menu.txt
apple
orange
mango
#find / -name menu.txt -exec cp {} /doc/menu2.txt \;

$find / -name menu.txt > find.txt 2> /dev/null
////////////////////////////////////
```

[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(5일차)

계정관리

퍼미션과 소유권

■공지사항

-다음 달 수강신청을 해 주시기 바랍니다.

<계정 관리(User Account Management)>

■학습개요

-리눅스를 집에서 혼자 사용할 경우에는 항상 root 로 로그인해서 작업을 하실 수 있습니다.

-리눅스 운영체제는 다중 사용자 운영체제로 여러 사람이 이용하는 서버인 경우에는 리눅스 서버 관리자 루트(root)가 아이디를 생성하고 암호를 지정해 주어야 로그인해서 리눅스 운영체제를 이용할 수 가 있습니다.

-리눅스에서 프로그램을 설치할 때도 사용자와 그룹을 추가해주어야하는 경우도 있습니다.(예:오라클)

■사용자 및 그룹 관리

☞계정관리 관련 파일

/etc/passwd	: 계정 정보 저장
/etc/shadow	: 사용자의 인코딩된 암호 및 암호 사용기간 저장
/etc/gshadow	: 그룹 암호 정보 저장
/etc/group	: 그룹 정보 저장
/etc/login.defs	: UID, GID 범위
/etc/default/useradd	: 디폴트값 저장
/etc/skel	: 계정 추가시 복사 될 파일이 있는 디렉토리
/home	: 일반 사용자 홈디렉토리 베이스

☞계정관리 명령어

1)계정(사용자) 추가(add)	----> adduser, useradd
2)계정 삭제(delete)	----> userdel
3)사용자 정보 수정(modify)	----> usermod

4)그룹 추가	---> groupadd
5)그룹 삭제	---> groupdel
6)그룹 정보 수정	---> groupmod
7)암호 설정 및 변경	---> passwd

☞사용자 추가 명령어: adduser, useradd

옵션)

- c : 사용자 설명(comment)
- d : 홈디렉토리(directory)
- e : 계정 사용기간 제한(만료날짜:expire)
- f : 암호사용기간 만료후 계정 잠금(Lock)할 날짜수
- g : 기본그룹
- G : 보조그룹
- u : UID(유저번호)
- m -k : skel 디렉토리 지정
(계정 추가시 /etc/skel 디렉토리 내용이 복사된다.)
- M : 홈디렉토리 생성 안 함
- s : 셸지정

- ▷사용자 이름(Account Name, Login Name)은 한글은 안되고
일반적으로 영문 또는 영문과 숫자를 조합해서 지정한다. (user, user1, user01)
- ▷@, #, !, *, %, ~ 같은 특수문자를 사용자 이름에 사용할 수 없다.
- ▷사용자 이름에 _(언더바) 와 .(점)은 사용 가능(andy.kim, andy_kim)
- ▷사용자 이름에 \$ 는 마지막 문자로 사용가능(nice\$ 는 가능)
- ▷그리고 대소문자를 구분한다.
- ▷#은 셸이 주석(설명)으로 처리한다.

☞그룹(Group)

사용자를 묶어 놓은 개체
회사의 부서

////////////////////////////////////
[신상품 보고서]

마케팅부 ---> 그룹
홍길동 ---> 사용자
////////////////////////////////////

////////////////////////////////////

이름 : 홍길동

부서 : 총무부, 인사부, 기술부

사용자 : apple

그룹 : apple, sys, adm

---> apple 이 기본그룹, sys 와 adm 은 보조그룹

////////////////////////////////////

☞기본 그룹(1차 그룹)과 보조 그룹(2차 그룹)

그룹에는 기본 그룹(Initial Group,Primary Group)과

보조그룹(Supplementary Group,Secondary Group)이 있다.

기본 그룹은 반드시 지정되어야 하는 그룹(-g 옵션)이고

보조그룹은 선택적으로 지정할 수 있는 그룹(-G 옵션)이다.

-g : 기본 그룹 지정(initial group)

--->파일 생성시 기본 그룹으로 지정이 되고

ls -l 로 보았을 때 출력되는 것이 기본 그룹이다.

-G : 보조 그룹 지정(supplementary group)

기본 그룹은 1개만 지정 가능하고

보조 그룹은 여러 개 지정 가능하다.

■명령어 연습

#adduser 또는 #useradd

(사용법 출력됨)

#adduser james

--->옵션이 없으면 디폴트 값으로 적용 됨

#mkdir /users

#adduser -c "James Bond" -d /users/james2 james2

-c : 계정에 대한 설명(commnet) 지정하는 옵션

-d : 홈디렉토리 지정 옵션

--->James Bond 에서 옵션의 인자값이 공백을 포함하고 있으면
쿼팅을 사용해야 한다.

--->홈디렉토리 지정시 상위 디렉토리는 이미 만들어져 있어야 한다.

--->위의 예제는 옵션 순서를 변경해도 된다.

```
#ls /users
```

james2 디렉토리가 생김

```
#cat /etc/passwd
```

(계정 등록 파일)

```
james:x:502:502:James Bond:/users/james:/bin/bash
```

아이디:패스워드자리:UID:GID:설명:홈디렉토리:셸

1) 2) 3) 4) 5) 6) 7)

1)사용자명(login name, account name)

2)암호자리(place holder)

3)사용자 번호(User ID)

4)그룹번호(Group ID)

5)comment 정보

6)사용자의 홈디렉토리 위치

7)로그인시 사용하는 셸 위치

```
#cat /etc/shadow
```

(암호 등록 파일)

```
james:!:15380:0:99999:7:::
```

아이디:인코딩암호:암호변경한날짜:min:max:warning:inactive:expire

1) 2) 3) 4) 5) 6) 7) 8)

1)사용자명

2)암호화된 패스워드

3)최종암호변경일 : 1970년 1월 1일을 기준으로 날짜수로 표현 됨

4)암호변경 최소일 수 : 최종암호변경일을 기준으로 최소일수가 지나야만 암호변경 가능

5)암호변경최대일수 : 최종암호변경일을 기준으로 현재 암호를 변경하지 않고

사용할 수 있는 날짜수

6)경고일 수 : 암호 최대 사용기간에 가까와 질 때 며칠 전에 경고를 보낼 것인지 지정하는
항목

7)비활성화 설정 : 암호 최대사용기간 만료후 설정된 날짜수까지 로그인 하지 않으면 계정이
락(lock) 됨

8)계정 만기일 : 사용자가 시스템을 마지막으로 사용할 수 있는 날짜

두번째 항목 !! 는 계정이 락(Lock)되어 있다는 의미이다.

--->로그인을 못하는 상태

```
#cat /etc/group
```

(그룹이 저장되는 파일)

```
james:x:502:
```

```
그룹명:x:GID
```

--->계정을 추가하면 /etc/passwd, /etc/shadow, /etc/group 에
동시에 등록되고 암호를 지정해 주어야 로그인 할 수 있다.

```
#passwd james
```

암호 동일하게 두 번 입력(입력시 안 보임)

--->james 계정의 암호 설정

--->root 는 짧은 암호를 설정해 줄 수 있다.(단, enter 를 두번 입력하면 안 된다.)

--->일반 사용자는 짧은 암호 또는 단순한 암호를 사용할 수 없다.

--->#passwd 만 입력하면 현재 로그인한 계정의 암호가 변경 됨

--->관리자 계정 루트는 다른 계정의 암호 설정을 할 수 있고

일반 계정은 자신의 암호만 변경할 수 있다.

```
#cat /etc/shadow
```

```
james:$1$5Two1EpN$VfNAvgg6HHICl.DLvCO6J0:15380:0:99999:7:::
```

```
#
```

---> !! 에서 인코딩된 문자열로 변경됨

---> 암호는 인코딩(암호화)되서 shadow 파일에 등록된다.

---> james 계정이 암호를 잃어버린 경우에는

root 가 암호를 다시 설정해 주면 된다.

※로그인 처리 과정

login : 사용자ID

password : 암호

암호를 잘못 입력하였을 경우에는 백스페이스키(←) 를 이용하여

지우고 다시 입력할 수 있다.
(Delete 키로 암호 삭제 불가)

/etc/passwd : ID 확인
/etc/shadow <-----> 사용자가 입력한 암호 인코딩
비교

- 1)사용자 아이디 입력
- 2)암호 입력
- 3)/etc/passwd 에서 아이디 확인
- 4)사용자가 입력한 암호를 인코딩
- 5)4번과 /etc/shadow 의 내용을 비교후 일치하면 로그인 되고
틀리면 로그인 거부 됨

```
#mkdir /skel2  
#touch /skel2/usertest.txt
```

```
#adduser -u 600 -m -k /skel2 james3  
---> james3 계정 추가시 사용자 번호는 600 으로 하고  
/skel2 디렉토리의 내용을 james3 의 홈디렉토리에 복사하라는 의미  
---> -k 옵션은 -m 옵션과 함께 사용
```

cf)
-M 옵션은 홈디렉토리를 만들지 않는 옵션

```
#grep james /etc/passwd  
james:x:502:502:James Bond:/users/james:/bin/bash  
james3:x:600:600::/home/james3:/bin/bash
```

```
#ls /home/james3  
#ls /skel2  
---> -m -k /skel2 옵션을 사용하였으므로  
/etc/skel 이 복사되지 않고 /skel2 의 내용이 복사 됨
```

--->계정을 추가하면 /etc/skel 디렉토리의 내용이 계정의
홈디렉토리에 자동적으로 복사된다.
--->/etc/skel 에는 주로 쉘설정파일들이 들어가 있다.
---> -m -k 옵션은 /etc/skel 디렉토리가 아닌 다른 디렉토리를 지정할 때 사용

☞ 계정 삭제

```
#ls /home
apple  green2  james3
#
```

```
#userdel james3
계정 삭제(홈디렉토리는 남는다.)
```

```
#ls /home
apple  green2  james3
```

```
#grep james /etc/passwd
#grep james /etc/shadow
#grep james /etc/group
```

```
#grep james /etc/{passwd,shadow,group}
```

---> 계정을 삭제하면 위의 3파일에서 모두 삭제된다.
계정삭제시 옵션을 사용하지 않은 경우 홈디렉토리는 수동으로
삭제해야 한다.

```
#adduser green2
#ls /home
apple  green2  james3
```

```
#userdel
#userdel -r green2
계정 삭제시 홈디렉토리도 지우려면 -r 옵션 사용
```

```
#ls /home
apple  james3
#
```

☞ 디폴트 옵션 확인 및 변경

```
adduser -D
-b 홈디렉토리
-e 만료날짜
```


-g 그룹변경
-s 셸변경
-f 비활성화

```
#adduser -D  
GROUP=100  
HOME=/home  
INACTIVE=-1  
EXPIRE=  
SHELL=/bin/bash  
SKEL=/etc/skel  
CREATE_MAIL_SPOOL=yes
```

INACTIVE(비활성화)=-1
-1 이면 disable 상태
0 이면 암호 사용기간이 만료되자마자 계정 사용 불가능

☞ 계정추가시 적용되는 디폴트 옵션 변경하기

```
#adduser -D -b /users
```

```
#adduser -D  
GROUP=100  
HOME=/users  
INACTIVE=-1  
EXPIRE=  
SHELL=/bin/bash  
SKEL=/etc/skel  
#
```

```
#adduser pizza  
#tail -3 /etc/passwd  
pizza:x:504:504::/users/pizza:/bin/bash  
(홈디렉토리가 /users 로 설정됨)
```

```
#ls /users  
james pizza  
#  
---> /users/pizza 디렉토리가 생성 됨
```

☞ 홈디렉토리 베이스 원래대로 설정

```
#adduser -D -b /home
#adduser -D
```

☞ 사용자 정보 변경 : usermod (User Modify)

- c comment
- d 홈디렉토리
- e 만료날짜
- g 기본그룹
- G 보조그룹
- s 쉘변경
- u UID 변경
- L 락
- U 언락

```
#adduser nice
#grep nice /etc/passwd
```

```
#usermod
(사용법이 출력됨)
```

```
#usermod -c "hong kil dong" -d /work nice
---> usermod 를 사용하면 /etc/passwd 파일의 내용이 변경되는 것이며
홈디렉토리가 생성되거나 /etc/skel 이 복사되지 않는다.
```

```
---> -m(--move-home) 옵션을 이용하여 홈디렉토리 이동은 가능
(-m 옵션 사용시 디렉토리가 존재할 경우에는 /etc/passwd 파일의 내용만 변경 됨)
```

```
#grep nice /etc/passwd
#usermod -l newuser nice
---> login name(사용자 ID) 변경
#tail -3 /etc/passwd
```

☞ 계정 락(Lock) 및 언락(Unlock)
usermod 또는 passwd 명령 이용

```
usermod -L 계정명
usermod -U 계정명
passwd -l 계정명
```

```
passwd -u 계정명
```

```
////////////////////////////////////  
#man usermod  
-L      Lock a user's password.  
-U      Unlock a user's password.  
  
#man passwd  
-l      This option is used to lock the specified account  
-u      This is the reverse of the -l option  
////////////////////////////////////
```

```
#adduser banana  
#passwd banana  
1234  
1234
```

```
#grep banana /etc/shadow
```

```
#passwd -l banana  
#grep banana /etc/shadow
```

```
#passwd -u banana  
#grep banana /etc/shadow
```

☞ 그룹추가

```
#groupadd student  
#tail -3 /etc/group
```

-그룹을 추가하면 똑같은 이름의 사용자 계정이 자동으로 생성되지 않는다.
-계정을 추가하면 계정과 똑같은 그룹이 등록되고
UID(사용자번호)와 GID(그룹번호)는 파일의 Maximum 숫자 + 1 로 등록된다.

☞ 그룹수정

```
#groupmod -g 777 student  
--->그룹 번호 수정  
#tail -3 /etc/group
```

☞ 그룹 삭제

```
#groupdel student
#tail -3 /etc/group
```

■chage 명령어

암호 사용기간, 계정 사용기간 확인 및 설정 명령어

```
chage -l 계정명
--->정보 확인
```

```
#chage -m 3 -M 10 apple
-m : 암호사용 최소일 수
-M : 암호사용 최대일 수
-W : 암호 사용 만료전 경고 날짜 수
-E : 계정 만료기간(년도-월-일)
```

```
#chage -l apple
```

```
#chage -m -1 -M -1 apple
옵션값을 -1 로 설정하면 초기화 됨
```

```
#chage -l apple
```

▷GUI 를 이용한 계정 관리

```
#system-config-users
```

■실습

1. 그룹추가

그룹이름(그룹번호)

business(1001) sales(1002) tech(1003)

2. 사용자 추가

사용자명 : user1

사용자명 : user2

유저번호 : 1001

유저번호 : 1002

기본그룹 : business

기본그룹 : user2

보조그룹 : sales, tech

설명 : test man1

홈디렉토리 : /users/user1

암호 : 1234

암호사용기간 최소일수 : 2

암호사용 최대일수 : 30

암호만료 경고일수 : 2

3. Putty 이용하여 로그인 테스트

※사용자 정보 확인 : #id 사용자명 #groups 사용자명

※사용자명과 기본그룹이 동일할 때는 -g 옵션 필요 없음

보조그룹 : tech

설명 : test man2

홈디렉토리 : /home/user2

암호 : 1234

암호사용기간 없음

////////////////////////////////////

#groupadd -g 1001 business

#groupadd -g 1002 sales

#groupadd -g 1003 tech

#useradd -u 1001 -g business -G sales,tech -c "test man1" -d
/users/user1 user1

#passwd user1

#chage -m 2 -M 30 -W 2 user1

#chage -l user1

#adduser -u 1002 -G tech -c "test man2" user2

#passwd user2

사용자 정보 확인

id : 유저번호, 그룹번호 출력

groups : 문자로 출력

#id user1

#id user2

#groups user1

#groups user2

////////////////////////////////////

<퍼미션과 소유권(Permission and Ownership)>

■학습개요

-리눅스 운영체제는 퍼미션(권한)과 소유권을 통해 다른 사용자들로부터 자신이 만든 파일을 보호할 수 있습니다.

-관리자는 퍼미션과 소유권을 통해 일반사용자들의 파일 (또는 디렉토리)에 대한 접근을 제어할 수 있습니다.

☞ 권한을 가지는 개체 : user, group, others

u ---> user, 소유자(사용자), owner

g ---> group, 그룹(사용자들을 묶을수 있는 개체, 학교의 동아리, 회사의 부서)

o ---> others, 다른 계정들(everyone)

a ---> all(3 개체를 모두 가리킴)

☞ 기본권한(permission)

r read 4 (읽기)

w write 2 (쓰기)

x execute 1 (실행)

7 rwx

6 rw-

5 r-x

4 r--

3 -wx

2 -w-

1 --x

0 ---

(권한이 없을 때는 - 로 표시)

☞ r, w, x 의 의미

1. r (read)

파일 : 파일의 내용을 볼 수 있다.

디렉토리 : 디렉토리의 내용을 ls 로 볼 수 있다.

2. w (write)

파일 : 파일을 수정할 수 있다.
디렉토리 : 디렉토리의 내용을 변경할 수 있다.
(파일을 삭제할 수 있는 권한은 파일의 w 권한이 아니고
그 파일이 있는 디렉토리에 w 권한이 있어야 파일이 삭제된다.)

3. x (execute)

파일 : 파일을 실행할 수 있다.
디렉토리 : 디렉토리 안으로 cd 를 이용하여 들어갈 수 있다.

---> 퍼미션과 소유권은 일반 계정들에 대해서는 정확하게
적용되고 슈퍼유저인 루트는 파일의 실행권한 만 빼고 퍼미션과
소유권을 초월한다.
(루트라 할지라도 파일에 실행권한이 있어야 실행할 수 있다.)

👉파일 또는 디렉토리 퍼미션 변경 : chmod(Change Mode)

(파일 소유자나 루트만이 사용할 수 있다.)

-R 옵션 : 하위 디렉토리 및 파일을 모두 변경할 때 사용

+ //권한 추가
- //권한 제거
= //새로 권한 지정

예)

chmod ugo+rx my.txt (---> chmod a+rx my.txt 와 동일하다.)

chmod 755 my.exe

👉소유자 변경 : chown(Change Owner)

👉그룹 변경 : chgrp(Change Group)

-R 옵션(디렉토리 일 때 하위 디렉토리 및 파일 모두 변경할 때 사용)

////////////////////////////////////

[신상품 보고서]

마케팅부 ---> 그룹

홍길동 ---> 사용자

////////////////////////////////////

<테스트>

[cent]

#cd /work

(디렉토리 이동이 안되면 #mkdir /work 한 후 이동)

#ls

hello.exe

////////////////////////////////////

hello.exe 파일이 없을 경우 아래와 같이 hello.exe 파일을
만든 후 실행하기

#vi hello.c

#include <stdio.h>

```
int main() {  
    printf("Linux Test \n");  
    return 0;  
}
```

#gcc -o hello.exe hello.c

---> gcc // hello.c 파일을 실행가능한 형태의 hello.exe 파일로 만드는 명령어

---> gcc -o 출력파일명 원본파일명

---> gcc 가 없으면 #yum -y install gcc 로 설치

////////////////////////////////////

#ll hello.exe

-rwxr-xr-x 1 root root 4725 Aug 7 14:42 hello.exe
7 5 5

퍼미션 : 하드링크수 : 소유자 : 그룹 : 크기(바이트) : (수정)날짜 : 파일명

퍼미션 -rwxr-xr-x 에서

첫번째 문자가 - 이면 일반파일이고 d 이면 디렉토리이다.

나머지 문자는 3개씩 묶어서

소유자(User, Owner), 그룹(Group), 다른계정들(Others) 의 퍼미션을
뜻한다.

$rw\bar{x} \Rightarrow 4+2+1 \Rightarrow 7$

$r\bar{-}x \Rightarrow 4+0+1 \Rightarrow 5$

```
#chmod u-x,g-x,o-rx hello.exe
```

u-x,g-x,o-rx 사이에 공백이 있으면 안 된다.

u-x,g-x,o-rx 의 의미

user 퍼미션에서 실행권한 x 제거

group 퍼미션에서 실행권한 x 제거

others 퍼미션에서 읽기 및 실행권한 rx 제거

```
#ll hello.exe
```

```
-rw-r----- 1 root root 4725 Jun  5 20:41 hello.exe
```

```
#./hello.exe
```

```
-bash: ./hello.exe: Permission denied
```

원하지 않은 프로그램을 실행시킬수 있으므로 루트라 할지라도 실행권한이 없으면 실행할 수 없다.

```
#chmod 755 hello.exe
```

user 개체에 7, group 개체에 5, others 개체에 5 퍼미션 할당

```
#ll hello.exe
```

```
-rwxr-xr-x 1 root root 4725 Jun  5 20:41 hello.exe
 7    5    5
```

r ---> 4

w ---> 2

x ---> 1

$$7 = 4(r) + 2(w) + 1(x)$$
$$5 = 4(r) + 1(x)$$
$$3 = 2(w) + 1(x)$$

-rwxr-xr-x 에서 첫번째 글자가 d 이면 디렉토리, - 이면 파일,

l 이면 링크파일(다른 파일과 연결되어 있다는 의미), b 는

블록장치, c 는 캐릭터 장치를 뜻한다.

```
#chmod a-x hello.exe
#ll hello.exe
-rw- r-- r-- 1 root root 4725 Jun  5 20:41 hello.exe
 6   4   4
```

```
#chmod ugo=rx hello.exe
#ll hello.exe
-r-xr-xr-x 1 root root 0  2월 12 01:26 hello.exe
```

```
#chown nobody hello.exe
파일의 소유자 변경
#ll hello.exe
-rw-r--r-- 1 nobody root 4725 Jun  5 20:41 hello.exe
```

```
#chown nobody111 hello.exe
chown: `nobody111': 잘못된 사용자
#
---> chown, chgrp 명령어 다음에는
/etc/passwd, /etc/group 에 등록되어 있는 사용자, 그룹이름을 적어야 한다.
```

```
/etc/passwd : 리눅스 사용자가 등록되어 있는 파일
/etc/group   : 리눅스 그룹이 등록되어 있는 파일
/etc/shadow  : 리눅스 사용자의 패스워드(암호)가 있는 파일
```

```
#chgrp sys hello.exe
파일의 그룹 변경
#ll hello.exe
-rw-r--r-- 1 nobody sys 4725 Jun  5 20:41 hello.exe
#
```

nobody 와 sys, adm 은 리눅스 설치시 자동으로 등록되는
시스템 계정(사용자)과 시스템 그룹이다.

```
#chown root.adm hello.exe
또는 #chown root:adm hello.exe
소유자와 그룹 동시에 변경
```

```
#ll hello.exe
-rw-r--r-- 1 root adm 4725 Jun  5 20:41 hello.exe
```

#

■실습

1. 아래와 같이 사용자, 그룹 추가 및 폴더(디렉토리) 생성 및 권한 설정
(그룹 먼저 생성후 사용자 추가)
2. myuser1 과 myuser2 는 보조그룹(-G 옵션)을 mybusiness 그룹으로 설정
3. myuser3 과 myuser4 는 보조그룹을 mytech 로 설정
4. Putty 이용하여 myuser1, myuser3 으로 로그인 한 후
/Data1, /Data2 폴더 이동 테스트

사용자 : myuser1 myuser2 myuser3 myuser4

암호 : 계정 모두 1234 로 설정

그룹 : mybusiness mytech

/Data1 : 영업부 전용 폴더

drwxrwx--- root mybusiness

/Data2 : 기술부 전용 폴더

drwxrwx--- root mytech

////////////////////////////////////

#groupadd mybusiness

#groupadd mytech

#adduser myuser1

#adduser myuser2

#usermod -G mybusiness myuser1

#usermod -G mybusiness myuser2

#adduser -G mytech myuser3

#adduser -G mytech myuser4

#id myuser1

#id myuser2

#id myuser3

#id myuser4

#passwd myuser1

1234

1234

myuser2 myuser3 myuser4 동일하게 암호 설정

```
#mkdir /Data1 /Data2
```

```
#ls /
```

```
#ls -ld /Data1 /Data2
```

```
#chmod 770 /Data1 /Data2
```

```
#chgrp mybusiness /Data1
```

```
#chgrp mytech /Data2
```

```
#ls -ld /Data1 /Data2
```

////////////////////////////////////

■리부팅 및 종료

리부팅 : reboot, init 6, shutdown -r now(시간)

종료 : halt, init 0, poweroff, shutdown -h now(시간)

☞reboot 와 shutdown 그리고 init 6 의 차이

reboot ---> shutdown ---> init

(reboot 는 shutdown 을 호출하고 shutdown 은 init 을 호출한다.)

////////////////////////////////////

```
#man reboot
```

DESCRIPTION

Halt 명령은 /var/log/wtmp 파일에 시스템 종료 기록을 남기고, 시스템 종료 나 리부팅 작업을 한다. 만약 runlevel이 0이나 6이 아닌 상태의 시스템 에 서 halt 나 reboot 명령이 사용되면, shutdown(8) 명령이호출된다.(-h나 -r 옵션 기능을 함)

```
#man shutdown
```

Shutdown은 init 프로세서의 시그널 처리에 의해서 수행되며, runlevel 바꾸기를 요청한다

////////////////////////////////////

[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(6일차)

특수권한

ACL

프로세스 관리

<특수권한>

☞special permission

4000 setuid 실행시 소유자 권한을 갖는다.

2000 setgid 실행시 그룹 권한을 갖는다.

1000 sticky bit 공용 디렉토리에 설정해서 소유자와 루트만이 파일을 삭제할 수 있도록 설정할 때 사용한다.

☞setuid

로그인한 사용자와 관계없이 프로그램을 실행할 때
그 프로그램의 소유자 권한으로 실행되는 특별한 권한
주로 루트 소유자로 되어있는 프로그램에 설정되어 있다.

[cent]

#ll /etc/shadow

#which passwd

/usr/bin/passwd

#

#ll /usr/bin/passwd

--->setuid 가 적용되어 있다.

☞테스트 계정 생성하기

[cent]

adduser : 새로운 계정(사용자)을 생성시키는 명령

apple 계정이 없으면 아래와 같이 생성해 주고 암호를 지정해 준다.

#adduser apple

--->apple 사용자 생성

#passwd apple

1234

1234

--->apple 사용자의 암호 설정
(입력시 아무것도 안 보이므로
정확히 타이핑하고 엔터를 누르면 된다.)
루트일 때는 짧은 암호를 설정할 수 있다.

[Putty]

windows 에서 Putty 이용하여 cent 리눅스로 apple 로그인
\$ id
uid=500(apple) gid=500(apple) groups=500(apple)

\$cat /etc/passwd
---> O (파일 내용 출력 됨)
---> /etc/passwd 파일은 사용자 계정이 등록되어 있는 파일

\$cat /etc/shadow
---> X (파일 내용 출력 안 됨)
---> /etc/shadow 파일은 사용자 암호가 등록되어 있는 파일

\$ ll /etc/passwd /etc/shadow

\$passwd
Changing password for user apple.
Changing password for apple
(current) UNIX password: 1234(apple 의 현재 패스워드 입력)
New UNIX password: qwer ---> 일반계정은 간단한 암호를 사용할 수 없다. (루트는 지정할 수 있다.)
BAD PASSWORD: it is too short
New UNIX password: dovmfskfk (일반 계정은 6자 이상 단순하지 않은 암호를 지정해야 함)
Retype new UNIX password: dovmfskfk
passwd: all authentication tokens updated successfully.

---> passwd 명령어에 setuid 가 있어서 /etc/shadow 파일이 업데이트 된다.
---> setuid 로 인하여 일반계정도 자신의 암호를 변경할 수 있다.

[cent]

(루트 계정으로 작업-프롬프트가 # 상태)

```
#ll /usr/bin/passwd
```

```
#chmod u-s /usr/bin/passwd
```

(setuid 제거)

```
#ll /usr/bin/passwd
```

[Putty]

(apple 계정으로 작업-프롬프트가 \$ 상태)

```
$ passwd
```

---> passwd 명령어에서 setuid 를 제거하면 일반계정은 암호를 업데이트 할 수 없다.

👉passwd 권한 원래대로 설정하기

[cent]

(루트 계정으로 작업하기)

```
#ll /usr/bin/passwd
```

```
#chmod u+s /usr/bin/passwd
```

```
#ll /usr/bin/passwd
```

👉sticky bit

setuid 와 setgid 는 일반적으로

디렉토리가 아니라

루트 소유자로 된 실행가능한 프로그램에 지정하고

sticky bit 는 디렉토리에 지정한다.

[cent]

```
#adduser banana
```

```
#passwd banana
```

암호지정

```
1234
```

```
1234
```

```
#passwd apple
```

```
1234
```

```
1234
```

```
#ls -l /
```

```
drwxrwxrwt 12 root root 4096 Jun 10 21:38 tmp
```

---> sticky bit 가 설정되어 있다.(1777)

```
[Putty]
```

```
apple login
```

```
$cd /tmp
```

```
$touch apple2.txt
```

```
$ll a*
```

```
-rw-rw-r-- 1 apple apple 0 12월 12 00:34 apple2.txt
```

```
$su banana
```

---> banana 계정으로 전환

Password: banana 계정의 암호 입력

```
$id
```

```
uid=501(banana) gid=501(banana) groups=501(banana)
```

```
$pwd
```

```
/tmp
```

```
$ll a*
```

```
-rw-rw-r-- 1 apple apple 0 Jun 10 22:19 apple2.txt
```

```
$rm apple2.txt
```

```
rm: remove write-protected regular empty file `apple2.txt'? y 입력
```

```
rm: cannot remove `apple2.txt': Operation not permitted
```

```
$ls
```

```
apple2.txt
```

```
$
```

---> sticky bit 로 인하여 다른 계정이 만든 파일이 삭제되지 않는다.

---> 디렉토리 퍼미션이 777 이라면 다른 계정이 만든 파일을 지울수 있다.

---> 파일을 삭제할 때는 파일의 w 권한이 적용되는 것이 아니고

그 파일이 속한 디렉토리의 w 권한이 적용된다.

파일의 w 권한은 파일의 내용을 수정할 수 있다는 의미이다.

---> sticky bit 는 주로 공용 디렉토리(인터넷 게시판에 해당)에 설정하며

루트나 파일 소유자만이 해당 파일을 삭제할 수 있다.

■ACL(Access Control List)

파일과 디렉토리의 확장 속성

getfacl : acl 확인

setfacl : acl 설정

-m // 수정

-x // 삭제

-b // 모든 ACL 정보 삭제

형식)

getfacl 파일명

setfacl -m [acl] 파일명

setfacl -x [acl] 파일명

setfacl -m u:apple:7 파일명

setfacl -m u:banana:rwX 파일명

setfacl -m g:sys:4 파일명

setfacl -m o:r 파일명

```
////////////////////////////////////  
#man getfacl
```

GETFACL(1)

Access Control Lists

GETFACL(1)

NAME

getfacl - get file access control lists

SYNOPSIS

getfacl [-dRLPvh] file ...

getfacl [-dRLPvh] -

DESCRIPTION

For each file, getfacl displays the file name, owner, the group, and the Access Control List (ACL). If a directory has a default ACL, getfacl also displays the default ACL. Non-directories cannot have

default ACLs.

#man setfacl

SETFACL(1) Access Control Lists SETFACL(1)

NAME

setfacl - set file access control lists

SYNOPSIS

setfacl [-bkndRLPvh] [{-m|-x} acl_spec] [{-M|-X} acl_file] file ...

setfacl --restore=file

DESCRIPTION

This utility sets Access Control Lists (ACLs) of files and directories. On the command line, a sequence of commands is followed by a sequence of files (which in turn can be followed by another sequence of commands, ...).

EXAMPLES

Granting an additional user read access

setfacl -m u:lisa:r file

Revoking write access from all groups and all named users (using the

effective rights mask)

setfacl -m m::rx file

Removing a named group entry from a file's ACL

setfacl -x g:staff file

////////////////////////////////////

[cent 가상머신]

#ls -ld /root

dr-xr-x--- root root /root

5 5 0

```
#setfacl -m u:apple:rwx /root
```

```
#ls -ld /root
```

--->ACL 을 설정하면 실제 적용되는 권한과 다르게 표시되는 경우도 있으므로
getfacl 로 확인을 해야 함

```
#getfacl /root
```

```
#id banana
```

```
#usermod -g root banana
```

---> banana 계정의 기본그룹을 root 로 변경

```
#id banana
```

---> 기본 그룹 변경확인

[Putty]

1) apple 계정으로 로그인해서 테스트

```
$cd
```

```
$pwd
```

```
/home/apple
```

```
$ls /root
```

```
$cd /root
```

```
$touch apple.txt
```

```
$ll
```

2) banana 계정으로 로그인(또는 su - banana)

```
[banana@linux1 ~]$
```

```
[banana@linux1 ~]$ ls /root
```

```
[banana@linux1 ~]$
```

```
[banana@linux1 ~]$ cd /root
```

```
[banana@linux1 root]$ pwd
```

```
/root
```

```
[banana@linux1 root]$ touch banana.txt
```

```
touch: cannot touch `banana.txt': 허가 거부됨
```

```
[banana@linux1 root]$
```

```
[banana@linux1 root]$ ls -ld /root
```

```
drwxrwx---+ 19 root root 4096 2월 11 12:36 /root
```

```
[banana@linux1 root]$
```

```
[banana@linux1 root]$ id
uid=512(banana) gid=0(root) groups=0(root)
[banana@linux1 root]$
```

3) candy 계정으로 테스트

```
[cent]
#adduser candy
#passwd candy
1234
1234
```

```
[Putty]
$su - candy
$ls /root
$cd /root
```

■일반 계정에서 루트 또는 다른 계정으로 전환하기
su : Switch User

-관리자도 일반 계정으로 사용하다가 루트 권한이 필요한 경우
su 를 이용하여 루트 로 전환하기 위해서 사용되며
테스트 할 때 A 계정에서 B 계정으로 전환할 때도 이용된다.
-루트는 계정전환시 암호를 묻지 않고
일반 계정은 암호를 물어본다.

1. 루트 계정으로 전환

- 1)su (su root 와 동일)
- 2)su - (su - root 와 동일)

2. 다른 계정으로 전환

- 1)su 계정명
- 2)su - 계정명

su 다음에

- 를 사용하면 모든 셸설정파일을 읽어서 로그인한 것과 비슷하게 되고
- 가 없으면 모든 셸설정파일을 읽지 않아서 일부 환경변수는
이전 계정의 내용으로 존재한다.

이전 계정으로 돌아올 때는 #exit 하면 된다.
 putty 접속을 종료할 때도 #exit 를 입력하면 된다.

```

        su
    ----->
apple                                root
    <-----
        exit
    
```

■실습

1. /work 디렉토리 권한과 소유권을 아래와 같이 설정

```
drwxr-x--- root sys /work
```

다른 계정은 /work 에 접근할 수 없고

apple 계정과 banana 계정만 /work 디렉토리를 이용할 수 있도록 ACL 설정

(apple 계정은 rwx 권한을 ACL 이용하여 적용)

(banana 계정은 r-x 권한을 ACL 이용하여 적용)

2. 결과 확인후 원래대로 설정

1)모든 acl 삭제

2)폴더 권한 아래와 같이 설정

```
drwxr-xr-x root root /work
```

3. cat 명령어에 SetUID 설정하고 Putty 이용하여 apple 로그인한 후

```
$cat /etc/shadow 테스트
```

4. /tmp2 폴더 생성후 777 로 권한 설정

5. Putty 로 apple 계정 로그인 후 /tmp2 폴더에 apple.txt 생성

6. banana 계정으로 전환 후 /tmp2 폴더의 apple.txt 삭제 테스트

```

////////////////////////////////////
#chmod 750 /work
#chgrp sys /work
#ls -ld /work
#setfacl -m u:apple:rwx /work
#setfacl -m u:banana:r-x /work
#getfacl /work
    
```

일반계정으로 테스트

```
#setfacl -b /work
#ls -ld /work
#chmod 755 /work
#chown root:root /work
#ls -ld /work
```

```
#chmod u+s /bin/cat 또는 #chmod 4755 /bin/cat
```

```
#mkdir /tmp2
#chmod 777 /tmp2
#ls -ld /tmp2
```

```
////////////////////////////////////
```

<umask 사용하기>

umask : 파일이나 디렉토리가 만들어 질 때 디폴트 퍼미션을
결정하는 값 또는 명령어

디폴트 퍼미션

디렉토리 : 777 - umask

파일 : 666 - umask

mode(umask)	File	Directory
0000	666	777
0001	666	776
0002	664	775
0022	644	755

※모드가 0001 일 때 파일은 665 가 아니고 666 임
(파일에는 기본적으로 실행권한이 없으므로)

r(4) w(2) x(1)

6 : rw-

5 : r-x

```
-rw-rw-rw-    -rw-rw-r-x
 6   6   6      6   6   5
```

////////////////////////////////////

umask 가 010 일 때

디렉토리 기본퍼미션)

777

010

767

파일 기본퍼미션)

666

010

666

////////////////////////////////////

[cent]

#umask

0022

---> umask 값 출력

#nl /etc/bashrc | more

(로그인시 /etc/bashrc 파일에 의해서 umask값이 결정된다.)

56 ~ 64 번 라인

기본설정

일반계정 : 002

루트 : 022

<프로세스 관리>

■용어정리

1. 프로그램 : 하드디스크에 저장되어 있는 파일중에서 실행가능한 파일
2. 프로세스 : 메모리에 로딩된 프로그램
3. 프로세스 분류
 - 1)대화형 프로세스 : 터미널 사용할 때
 - 2)배치 프로세스 : 특정 시간 또는 시스템이 한가할 때 작업을 몰아서 실행(at, batch)
터미널과 교류가 없다.

3)데 문 : 부팅중에 메모리에 로딩되어 종료될 때까지
상주해 있는 프로그램
(윈도우에서는 서비스라 부른다.)

- ▷프로세스는 여러 가지 자원(CPU,메모리,장치)을 사용한다.
- ▷프로세스는 해당 명령을 수행하기 위해 CPU 를 점유하기도 하고
명령어와 데이터를 저장하기 위해 물리적인 메모리를 사용한다.
- ▷프로세스는 운영체제의 제어를 받으면서 실행,대기,중단,좀비 의
한 상태를 갖는다.

실행(running) : 프로세스가 현재 실행중인 상태
대기(waiting) : 운영체제가 자원을 할당해 주기를 기다리는 상태
중단(stopped) : Ctrl + z 를 입력받은 경우
좀비(zombie) : 프로세스가 종료된 상태이지만 정보가 완전히
삭제되지 않고 남아 있는 경우
리눅스 프로세스는 프로그램을 실행하는데 필요한 PID,UID,GID 정보를 포함한다.

PID(Process ID) : 프로세스가 시작할 때 할당받는 프로세스 식별번호
UID(User ID) : 사용자 번호
GID(Group ID) : 그룹 번호

프로세스 정보가 있는 디렉토리 : /proc
/proc 에는 각 프로세스에 해당하는 PID 디렉토리가 있다.
/proc 는 가상의 디렉토리로 커널 메모리를 마운트 한 것이다.

■프로그램 실행 방식

1. foreground : 전면실행(터미널을 프로그램이 점유하고 있는
상태로 프로그램이 종료 될 때까지 다른 명령어
를 실행할 수 없다.)
2. background : 후면실행(프로그램은 실행되면서 셸프롬프트가
나타나므로 프로그램을 계속 실행시킬 수 있다.)
예) 윈도우 도스창

---> 리눅스 터미널은 디폴트가 포그라운드 방식이다.
백그라운드 실행시키는 방법 : 명령어 & 또는 명령어&

#gcalctool (Putty 에서 하면 안되고 리눅스 가상머신 터미널에서 실행)


---> 기본적으로 포그라운드 로 실행된다.

종료(마우스 이용)

```
#gcalctool &  
[1] 1234  
#gcalctool &  
[2] 1235  
#gcalctool &  
[3] 1236
```


백그라운드로 프로그램을 실행시키면 번호가 두 개 나타나는데
첫번째 번호는 백그라운드로 실행된 프로그램 갯수 번호
두번째는 프로세스 아이디(PID) 이다.
리눅스 운영체제는 파일,프로세스,계정 등 을 모두 숫자로 구분해서 관리한다.

PID(Process ID) : 프로세스를 구분하는 번호

 jobs
현재 터미널에서 백그라운드로 실행된 프로세스 출력


```
#jobs  
+ 는 가장 최근의 작업  
- 는 그 이전 작업
```

```
#jobs -l  
PID 도 출력
```

 fg : 포그라운드로 전환
#fg %2
2번 작업을 포그라운드로 전환

ctrl + z : 프로세스 중지(stop) 시키는 단축키

```
#jobs
```

 bg : 백그라운드로 전환

```
#bg %2
(2번 작업을 백그라운드로 전환)
#jobs
```

 ps(report process status) : 프로세스 정보 출력

옵션

```
-a          // 전체 사용자(터미널 소유)의 모든 프로세스(all) 출력
-e          // 현재 실행중인 모든 프로세스(every) 출력
-f          // full list로 출력
-l          // long list로 출력
-m          // 스레드 정보 출력
-t TTY      // 지정한 TTY를 가진 프로세스 정보 출력(ps -t pts/2)
-u 사용자이름 // 지정한 사용자가 실행한 프로세스
-p PID      // 지정한 PID(프로그램 번호)를 가진 프로세스 정보 출력
-u UID      // 지정한 UID(사용자 번호)를 가진 프로세스 정보 출력
-g GID      // 지정한 GID(그룹 번호)를 가진 프로세스 정보 출력
-x          // 제어 터미널을 갖지 않은 프로세스도 출력(? 로 나타나는 프로세스)
```

#ps

현재 터미널에서 실행된 프로세스만 출력
TTY 는 프로그램이 실행된 터미널을 의미

#tty

현재 사용하고 있는 터미널 출력
(/dev/pts/번호)

모든 프로세스 출력하기

```
#ps -ef
#ps aux 또는 ps axu
```

```
#ps aux | more
```

ps aux 결과의 헤드라인 항목

```
USER(USER)          // 사용자 이름
PID(Process ID)     // 프로세스마다 주어지는 번호
%CPU                 // 프로세스가 사용하는 CPU 점유율
%MEM                 // 프로세스가 사용하고 있는 메모리 점유율
```

```
VSZ                                // 프로세스가 사용하는 가상메모리 사이즈(램 + 스
wap), 단위는 KB
RSS(Resident Set Size)           // 프로세스가 사용하는 실제 메모리의 크기
TTY(Tele TYpewrite)              // 명령어가 실행되는 터미널의 번호
STAT(STATE)                       // 실행되고 있는 프로세스 상태
    R                             // 실행 중 혹은 실행될 수 있는 상태
    S                             // sleep, 수면상태
    T                             // 정지된 상태 (suspend)
    Z                             // 좀비 (zombie) 프로세스
    X                             // dead (should never be seen)
    W                             // 스왑 out 된 상태
    <                             // 우선 순위가 인위적으로 높아진 상태
START(START)                     // 프로세스가 시작된 시간
TIME(TIME)                       // CPU가 사용한 시간
COMMAND(COMMAND)                 // 사용자가 실행한 명령어
```

사용자가 정의한 형식으로 출력

```
#ps -o pid,state,tt,args
```

```
#ps -o pid,pcpu,pmem,args
```

📖pstree : 프로세스를 트리 구조로 출력

```
////////////////////////////////////
```

```
[root@cent ~]# pstree --help
```

```
pstree: 부적절한 옵션 -- -
```

```
usage: pstree [ -a ] [ -c ] [ -h | -H pid ] [ -l ] [ -n ] [ -p ] [ -u ]
           [ -G | -U ] [ pid | user]
```

```
pstree -V
```

- a show command line arguments
- c don't compact identical subtrees
- h highlight current process and its ancestors
- H pid highlight process "pid" and its ancestors
- G use VT100 line drawing characters
- l don't truncate long lines

-n sort output by PID
-p show PIDs; implies -c
-u show uid transitions
-U use UTF-8 (Unicode)) line drawing characters
-V display version information
-Z show SELinux security contexts
pid start at pid, default 1 (init))
user show only trees rooted at processes of that user

////////////////////////////////////

#pstree

#pstree -a

---> 커맨드 라인 인자값 출력

#pstree -n

---> PID 로 정렬

#pstree -p

---> PID 출력

#pstree -np

---> PID 로 정렬 및 PID 출력

#pstree -au

---> 인자값 및 사용자 전이 출력

☞시그널(Signal) : 리눅스 프로그램에 보내는 신호

☞시그널이 발생하는 경우

- 1)키보드를 통해 : ctrl + c(인터럽트), ctrl + z(정지)
- 2)커널 내부에서 발생
- 3)다른 프로세스에 의해서

☞시그널 종류

#kill -l

- 1 HUP, 헵시그널(일반 명령어는 종료, 데몬 프로세스의 경우 설정 파일을 다시 읽음)

- 2 INT, 인터럽트(단축키는 ctrl + c)
- 9 KILL, 강제종료(미사일)
- 15 TERM, 자연스럽게 종료(디폴트 시그널, 총)
- 19 STOP, 실행정지(ctrl + z)

////////////////////////////////////

hang up
전화를 끊다

After I hung up I remembered what I'd wanted to say.
나는 전화를 끊은 후에 하고 싶었던 말이 생각났다.

Wait a minute, and don't hang up the phone.
전화 끊지 말고 잠시 기다리세요

////////////////////////////////////

☞kill 명령어
프로세스에게 시그널을 보냄

형식)
kill PID(숫자)
kill -9 PID1 PID2
kill -SIGINT 1234(PID)
kill -int 1235(PID)
kill -s SIGINT 1236(PID)
kill %작업번호

kill 다음에는 프로그램 이름을 적으면 안 됨
#kill gcalctool
----> X

백그라운드로 계산기 3개 실행하기(#gcalctool &)

```
#ps -ef | grep gcalctool
UID  PID  PPID  C  STIME  TTY  TIME  CMD
root  3201  2521  0  16:17  pts/1  00:00:00  gcalctool
root  3202  2521  0  16:17  pts/1  00:00:00  gcalctool
```

```
root 3203 2521 0 16:17 pts/1 00:00:00 gcalctool
```

grep gcalctool 은 ps -ef | grep gcalctool 명령어 자신이 출력된 것임

PID(Process ID) : 프로세스 아이디

PPID(Parent Process ID) : 부모프로세스 아이디

(gcalctool 의 부모프로세스는 bash 셸이 된다.)

#kill 1234(자신의 가상머신 터미널에서 출력된 PID 사용하기)

디폴트로 15번 시그널 전달

종료됨 출력

#kill -9 1235(두번째 계산기 프로그램 번호)

9번 시그널 전달

죽었음 출력

#kill -2 1236

인터럽트 시그널 전달

인터럽트 출력

🔗killall 명령어

동일한 이름의 여러 프로세스에게 시그널을 보낼 때 사용

killall -l

killall -s 시그널문자(대문자) 프로그램이름

killall -시그널숫자 프로그램이름

killall -시그널문자(대문자) 프로그램이름

🔗skill 명령어

시스템에 접속해 있는 사용자를 추방 할 때

공격자를 차단할 경우

skill -KILL(소문자 가능) apple(사용자명)

skill -KILL pts/터미널번호

skill -STOP apple

skill -CONT apple

```
sleep
sleep 초
#sleep 3
```

■실습

1. Putty 이용하여 루트 로 로그인하기
2. /usr/lib 디렉토리를 /tmp 로 복사하되 중간에 중지시키고 백그라운드로 실행하기
3. find 이용하여 / 부터 확장자 jpg 파일을 찾아 /work/find.txt 로 저장하기
(백그라운드 실행)
4. sleep 3000 실행후 중지시키고 kill 이용하여 종료하기

```
////////////////////////////////////
#cp --help
-R, -r      copy directories
#cp -R /usr/lib /tmp
Ctrl + z
#jobs
#bg      또는 #bg %작업번호
--->작업 번호를 생략하면 현재 작업(jobs 에서 + 표시)을 전환

#find / -name "*.jpg" > /work/find.txt &

#sleep 3000
Ctrl + Z
#jobs -l
#kill PID
#jobs -l
#kill -9 PID
#jobs -l
////////////////////////////////////
```

[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(7일차)

하드디스크 관리

■용어 정리

☞파일 시스템(File System)

자료(data)를 저장하고 찾기 쉽도록 만들어 놓은 방법, 체계적인 구조

☞파일시스템의 종류

FAT : windows 구버전(windows 98)이 사용

NTFS : windows 2000 이상이 사용

ext2 : 구버전 리눅스가 사용

ext3 : CentOS 5 버전에서 사용

ext4 : CentOS 6 버전에서 사용

UFS : 유닉스가 사용(솔라리스)

▲하드디스크(또는 CD-ROM Drive) 장치명

(E-IDE 또는 PATA)

primary master : /dev/hda (첫번째 IDE 하드디스크 또는 CD-ROM)

primary slave : /dev/hdb (두번째 IDE 하드디스크 또는 CD-ROM)

secondary master : /dev/hdc

secondary slave : /dev/hdd

(SCSI)

/dev/sda (첫번째 스카시 하드디스크)

/dev/sdb (두번째 스카시 하드디스크)

/dev/sdc

.....

SATA 방식의 하드디스크는 /dev/sd? 로 리눅스에서 인식된다.

CentOS 6 버전에서는 IDE 방식도 /dev/sd? 로 표시 됨

IDE,E-IDE : ATA, PATA

SATA : Serial ATA

▲전송 인터페이스

1. SASI: SCSI의 모태이다.
2. SCSI (스커지): 병행연산 버스의 원형이다.
3. ST-506(시게이트 인터페이스): 현재의 플로피 디스크를 제어하는 것과 동일한 MFM 방식이다.
4. ST-412(시게이트 인터페이스): MFM 방식에 RLL 압축 기술을 적용하여 용량을 늘렸다. ST-506를 개량한 것이다.
- ESDI: ST-412/506에 비해 동작 속도가 급격히 빨라졌지만, 이전의 규격들과 호환되지 않는다.
5. ATA: 초기의 병렬 방식인 ATA와 2003년에 개발된 직렬방식인 SATA가 있다.

출처 : <http://ko.wikipedia.org>

■디스크 관리 테스트

새로운 하드디스크를 추가하기 위해서 리눅스 가상머신을 종료한다.
(리눅스 가상머신이 부팅 된 상태 또는 일시정지 상태(II 로 표시) 에서는
가상 하드디스크 추가가 안 된다.)

[cent]

#poweroff

▲SCSI 방식 2G 가상 하드디스크 2개 추가하기

- 1) vmware 메뉴표시줄-VM-Settings
- 2) Add 버튼-Hard Disk 선택-Next
- 3) Create a new virtual disk 선택(첫번째 버튼)-Next
- 4) Virtual disk type : SCSI 선택-Next
- 5) Disk Capacity-2G 로 설정-Next
- 6) Disk file(가상하드디스크 저장 파일명, 디폴트값 그대로 사용)-finish
- 7) OK

하드디스크 추가후 Linux 부팅시키기

부팅시 멈추면 사각형 종료 버튼(□) 누른 후 다시 전원 버튼(▷)을 눌러서 부팅시켜 본다.

dmesg : 부팅시 커널이 발생한 메시지 출력 명령

#dmesg

#dmesg | grep sd

---> 부팅시에 커널이 새로운 하드웨어를 인식한 것을 알 수 있다.

```
////////////////////////////////////  
결과 캡처  
#dmesg | grep sd  
SCSI device sda: 16777216 512-byte hdwr sectors (8590 MB)  
sda: cache data unavailable  
sda: assuming drive cache: write through  
SCSI device sda: 16777216 512-byte hdwr sectors (8590 MB)  
sda: cache data unavailable  
sda: assuming drive cache: write through  
sda: sda1 sda2 sda3  
Attached scsi disk sda at scsi0, channel 0, id 0, lun 0  
EXT3 FS on sda1, internal journal  
EXT3 FS on sda3, internal journal  
Adding 522104k swap on /dev/sda2. Priority:-1 extents:1  
#
```

```
////////////////////////////////////
```

<하드디스크 작업 순서>

1. 물리적 부착
2. fdisk 로 파티션 생성
3. mkfs.ext4 로 포맷하기
4. 마운트해서 사용하기

▲fdisk 명령어

1)fdisk -l

디스크 정보 확인

2)fdisk -l /dev/sda

특정 디스크 정보 확인

3)fdisk /dev/장치명

디스크 파티션 구성

▲fdisk 내부에서 사용하는 명령어

a : 부트 플래그 설정
l : 파티션 목록(list)
n : 새 파티션 생성(new)
t : 파티션 변경(toggle,change)
w : 저장(write)
p : 파티션 출력(print)
q : 종료(quit)
d : 삭제(delete)

▲파일시스템 생성(포맷 개념)

- 1) mkfs.ext4 /dev/sdb1
- 2) mkfs -t ext4 /dev/sdb1

mkfs ---> make filesystem

mkfs.ext2
mkfs.ext3
mkfs.ext4

▲마운트 와 언마운트

리눅스는 모든 것을 파일로 간주한다.

하드디스크, CDROM 드라이브도 하나의 장치파일(Device File)로 간주한다.

사용자 입장에서는 하드디스크 장치파일을 통해서

데이타를 접근할 수 없고

장치파일을 디렉토리에 연결하는 작업(마운트)을

통해서 데이타에 접근할 수 있다.

마운트 : 파일시스템(장치)을 디렉토리에 연결하는 작업

명령어 ---> mount

언마운트 : 파일시스템(장치)을 디렉토리와 연결해제 하는 작업

명령어 ---> umount(unmount 가 아님)

▲마운트 및 언마운트 형식

mount -t 파일시스템타입 장치파일 마운트디렉토리
(마운트 디렉토리는 임의로 지정 가능하다.)

mount -t ext4 /dev/sdb1 /disk2
(-t ext4 는 생략 가능)

umount 마운트디렉토리 또는 umount 장치명(/dev/hdc)

▲하드디스크 구성 테스트

```
#ls /dev/sd*
```

```
#fdisk -l
```

```
-l(list)
```

```
Disk /dev/sda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	765	6144831	83	Linux
/dev/sda2		766	830	522112+	82	Linux swap / Solaris
/dev/sda3		831	1044	1718955	83	Linux

```
#
```

Id 83 은 리눅스 데이타 파티션을 의미하고

Id 82 는 리눅스 스왑 파티션을 의미한다.

수업용 하드디스크 구분:

[/dev/sda]	[/dev/sdb]	[/dev/sdc]
첫번째 하드디스크	두번째 하드디스크	세번째 하드디스크
(부트디스크)		

▷파티션

주파티션은 1-4번이고

5번 부터는 확장파티션 안의 논리파티션(logical partition)이다.

주파티션 : primary partition(4개 까지 가능)

확장파티션 : extended partition

논리파티션 : logical partion(번호가 5번 부터 할당)

파티션을 5개 이상 만드려면

확장파티션을 만들고 그 안에 논리 파티션을 만들어야 한다.

최대 파티션 수는 CentOS 5버전 까지는 15개 이다.

(주파티션 3개 + 확장 파티션 1개 + 논리 파티션 11개)

[주파티션1 | 주파티션2 | 주파티션3 | 확장파티션(논리파티션5 | 논리파티션6 ...)]

CentOS 6 버전에서는 논리파티션수에 제한 없음

확장파티션은 포맷해서 사용할 수 없음

Q)확장 파티션(extended partition) 은 몇개 만들 수 있는가 ?

A)1 개

Q)주파티션(primary partition)안에 논리파티션(logical partition)을 만들 수 있는가 ?

A)No

▷파티션 구성

#fdisk /dev/sdb

Command (m for help): m 입력

p print the partition table

Command (m for help): l(엘) 입력

파티션 종류 출력

Command (m for help): n(파티션 새로 생성)

Command action

e extended

p primary partition (1-4)

p (엔터)

Partition number (1-4): 1

First cylinder : 엔터

Last cylinder : +1G

(총 2G 에서 1G 만 할당)

+ 를 생략하면 실린더수로 인식 됨

실린더가 모여서 파티션을 이룬다.

Command (m for help): p

Command (m for help): w(저장후 종료)

The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.

#

#ls /dev/sd*

실습용 리눅스 가상머신의 하드디스크 구분 :

/dev/sda : 첫번째 하드디스크(부트디스크)

/dev/sdb : 두번째 하드디스크

/dev/sdc : 세번째 하드디스크

#fdisk -l(엘) /dev/sdb

▷파일시스템 생성하기-윈도우의 포맷개념

#mkfs.ext4 /dev/sdb1

▷마운트

#mkdir /disk2

#mount /dev/sdb1 /disk2

또는

#mount -t ext4 /dev/sdb1 /disk2

-t ext4 는 생략가능

```
#cat /etc/filesystems
```

---> 파일 시스템 종류

```
#df -h
```

-h : human readable

```
#cd /disk2
```

```
#pwd
```

/disk2

```
#
```

```
#cat > test.txt
```

apple

orange

mango

ctrl+d 입력 : 입력 종료 단축키

```
#
```

```
#ls
```

```
#cat test.txt
```

apple

orange

mango

```
#
```

▷마운트 해제

마운트 해제를 할 때는 마운트 된 디렉토리에서 umount 를 하면 안 됨

```
#pwd
```

/disk2

```
#cd
```

```
#pwd
```

/root

```
#umount /disk2
```

```
#df -h
```

////////////////////////////////////

마운트 해제가 안 되면 fuser -u 로 프로세스 확인후

fuser -k 로 프로세스 강제 종료후 마운트 해제를 할 수 있다.

```
fuser -u /disk2
fuser -k /disk2
umount /disk2
```

////////////////////////////////////

■실습

1. /disk3 /disk4 디렉토리 생성(#mkdir /disk3 /disk4)
2. 두번째 하드디스크(/dev/sdb) 남은 용량을 파티션 설정(/dev/sdb2)하고 ext4 파일시스템생성(포맷)한 후 /disk3 디렉토리에 마운트 테스트하기
(주의 : /dev/sdb1 을 마운트 해제한 후 작업해야 함,
마운트 해제를 하지 않으면 장치 파일이 생성되지 않는다.)
3. 세번째 하드디스크(/dev/sdc) 를 아래와 같이 파티션 생성
(마지막 파티션 용량은 정확하게 400M 이 되지 않아도 된다.)

[주파티션1 | 주파티션2 | 확장파티션3(논리파티션5,논리파티션6,논리파티션7)]
400M 400M 1.2G(400M,400M,400M)

fdisk 사용시)

파티션 생성 : n

파티션 삭제 : d

파티션 저장 : w

파티션 출력 : p

메뉴 출력 : m

////////////////////////////////////

#cd

#pwd

/root

#umount /dev/sdb1

#fdisk /dev/sdb

Command (m for help): p

Disk /dev/sdb: 2147 MB, 2147483648 bytes

255 heads, 63 sectors/track, 261 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x2154df34

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		1	132	1060258+	83	Linux

Command (m for help): n

Command action

e extended

p primary partition (1-4)

p

Partition number (1-4): 2

First cylinder (133-261, default 133): 엔터

Using default value 133

Last cylinder, +cylinders or +size{K,M,G} (133-261, default 261): 엔터

Using default value 261

Command (m for help): p

Disk /dev/sdb: 2147 MB, 2147483648 bytes

255 heads, 63 sectors/track, 261 cylinders

Units = cylinders of 16065 * 512 = 8225280 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x2154df34

Device	Boot	Start	End	Blocks	Id	System
/dev/sdb1		1	132	1060258+	83	Linux
/dev/sdb2		133	261	1036192+	83	Linux

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.

#

#mkfs.ext4 /dev/sdb2

mke2fs 1.41.12 (17-May-2010)

Filesystem label=

OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
64768 inodes, 259048 blocks
12952 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8096 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 31 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.

```
#mkdir /disk3
#mount /dev/sdb2 /disk3
#df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        13G   3.3G   8.7G  28% /
tmpfs            504M   260K   504M   1% /dev/shm
/dev/sda3        5.3G   143M   4.9G   3% /home
/dev/sr0         3.5G   3.5G    0 100% /media/CentOS_6.3_Final
/dev/sdb2        996M    18M   929M   2% /disk3
#
```

////////////////////////////////////

<자동마운트>

▲부팅시 자동 마운트 설정파일

```
#cat /etc/fstab
```

▲/etc/fstab : 부팅시 자동마운트 설정파일의 형식

- 1)장치명 또는 파티션 레이블(LABEL) 또는 UUID
- 2)마운트할 디렉토리
- 3)파일 시스템 형식
- 4)옵션
- 5)덤값 : dump 라는 프로그램으로 덤프여부, ext4 는 1, 다른 파티션은 0
- 6)파일시스템점검순서 : 루트파티션(ext4)은 1, 다른 파티션(ext4)은 2, 그 외 파티션은 0

UUID :

UUID stands for Universally Unique Identifier and it is used in Linux to identify disk in the /etc/fstab file

blkid : UUID 확인

☞부팅시 자동마운트 테스트

자동마운트를 테스트 하기 전에 /dev/sdb1 을 /disk2 에 수동마운트 해 본다.

```
mount /dev/sdb1 /disk2
```

```
df -h
```

수동마운트가 안 되는 디스크를 자동마운트 설정하면 리부팅 할 때 멈춤

```
#vi /etc/fstab
```

마지막 라인에 추가

```
/dev/sdb1 /disk2 ext4 defaults 1 2
```

저장후 종료

리부팅 해서 확인

```
#reboot
```

```
////////////////////////////////////
```

부팅중에 멈추면 루트 암호 입력

```
#mount -o remount,rw /
```

한 후 /etc/fstab 파일을 vi 이용하여 /dev/sdb1 라인을 삭제하고

저장및 종료한 후 리부팅

////////////////////////////////////

루트로 로그인

#df -h

결과 확인후 /etc/fstab 에서

/dev/sdb1 라인 삭제

■아이노드와 링크

☞리눅스 파일시스템의 구성요소

[Boot Block(MBR) | Super Block | Inode(meta data) | Data Block]

1. Boot Block(MBR) : 부팅할 때 실행되어야 할 코드가 담긴 부분
2. Super Block(수퍼 블록) : 파일시스템의 전체적인 정보를 갖는다.(파일시스템이 생성된 시각, Inode 갯수 등)
3. Inode(아이노드, meta data) : 파일의 이름을 제외한 해당 파일의 모든 정보를 갖고 있다.
4. Data Block(데이터 블록) : 데이터가 존재하는 블록

☞Inode Table

파일의 종류와 접근모드(rwx)

파일의 UID, GID

파일의 접근시간, 수정시간 ,아이노드 변경시간

데이터 저장에 사용된 총 데이터 블록수

데이터가 저장된 블록에 대한 포인터

총15개의 데이터 블록 포인터(주소정보)를 갖고 있다.

☞ext4 파일시스템은 ext2 와 구조는 같고 파일저장시 로그를 남기는 저널링 기능이 포함된 형태이다.

●dumpe2fs : 파티션 논리적인 구조를 볼 수 있는 명령어
(수퍼 블록 출력)

#dumpe2fs /dev/sda3

```
dumpe2fs 1.35 (28-Feb-2004)
Filesystem volume name:   /home
Last mounted on:          <not available>
Filesystem UUID:          6b361661-6db7-4f63-a81c-da58b686f939
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index fil
etype needs_recovery sparse_super large_file
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              570080
Block count:              1138606
Reserved block count:     56930
Free blocks:              1110280
Free inodes:              570069
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      277
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         16288
Inode blocks per group:   509
Filesystem created:       Tue Feb  8 20:05:13 2011
Last mount time:          Tue Feb  8 11:50:40 2011
Last write time:          Tue Feb  8 11:50:40 2011
Mount count:              2
Maximum mount count:      -1
Last checked:             Tue Feb  8 20:05:13 2011
Check interval:           0 (<none>)
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Journal inode:            8
Default directory hash:   tea
Directory Hash Seed:      4c36ba93-e489-44ab-bd5a-6534c8cd0fc0
Journal backup:           inode blocks
```

Group 0: (Blocks 0-32767)

Primary superblock at 0, Group descriptors at 1-1
Block bitmap at 279 (+279), Inode bitmap at 280 (+280)
Inode table at 281-789 (+281)
23769 free blocks, 16277 free inodes, 2 directories
Free blocks: 8999-32767
Free inodes: 12-16288

Group 1: (Blocks 32768-65535)

Backup superblock at 32768, Group descriptors at 32769-32769
Block bitmap at 33047 (+279), Inode bitmap at 33048 (+280)
Inode table at 33049-33557 (+281)
31962 free blocks, 16274 free inodes, 4 directories
Free blocks: 33558-55295, 55312-65535
Free inodes: 16303-32576

Group 2: (Blocks 65536-98303)

Block bitmap at 65536 (+0), Inode bitmap at 65537 (+1)
Inode table at 65538-66046 (+2)
32257 free blocks, 16288 free inodes, 0 directories
Free blocks: 66047-98303
Free inodes: 32577-48864

Group 3: (Blocks 98304-131071)

Backup superblock at 98304, Group descriptors at 98305-98305
Block bitmap at 98583 (+279), Inode bitmap at 98584 (+280)
Inode table at 98585-99093 (+281)
31978 free blocks, 16288 free inodes, 0 directories
Free blocks: 99094-131071
Free inodes: 48865-65152

Group 4: (Blocks 131072-163839)

Block bitmap at 131072 (+0), Inode bitmap at 131073 (+1)
Inode table at 131074-131582 (+2)
32257 free blocks, 16288 free inodes, 0 directories
Free blocks: 131583-163839
Free inodes: 65153-81440

Group 5: (Blocks 163840-196607)

Backup superblock at 163840, Group descriptors at 163841-163841
Block bitmap at 164119 (+279), Inode bitmap at 164120 (+280)
Inode table at 164121-164629 (+281)
31978 free blocks, 16288 free inodes, 0 directories
Free blocks: 164630-196607
Free inodes: 81441-97728

Group 6: (Blocks 196608-229375)

Block bitmap at 196608 (+0), Inode bitmap at 196609 (+1)

Inode table at 196610-197118 (+2)

32257 free blocks, 16288 free inodes, 0 directories

Free blocks: 197119-229375

Free inodes: 97729-114016

Group 7: (Blocks 229376-262143)

Backup superblock at 229376, Group descriptors at 229377-229377

Block bitmap at 229655 (+279), Inode bitmap at 229656 (+280)

Inode table at 229657-230165 (+281)

31978 free blocks, 16288 free inodes, 0 directories

Free blocks: 230166-262143

Free inodes: 114017-130304

Group 8: (Blocks 262144-294911)

Block bitmap at 262144 (+0), Inode bitmap at 262145 (+1)

Inode table at 262146-262654 (+2)

32257 free blocks, 16288 free inodes, 0 directories

Free blocks: 262655-294911

Free inodes: 130305-146592

Group 9: (Blocks 294912-327679)

Backup superblock at 294912, Group descriptors at 294913-294913

Block bitmap at 295191 (+279), Inode bitmap at 295192 (+280)

Inode table at 295193-295701 (+281)

31978 free blocks, 16288 free inodes, 0 directories

Free blocks: 295702-327679

Free inodes: 146593-162880

Group 10: (Blocks 327680-360447)

Block bitmap at 327680 (+0), Inode bitmap at 327681 (+1)

Inode table at 327682-328190 (+2)

32257 free blocks, 16288 free inodes, 0 directories

Free blocks: 328191-360447

Free inodes: 162881-179168

stat : 아이노드 정보확인

#stat 파일명

※링크는

1)디렉토리의 규칙성을 유지하기 위해서

2)긴 이름을 짧은 이름 또는 쉬운 이름으로 사용하기 위해서

3)리눅스 서버 설정할 때 이용된다.

```
#ll /etc/grub.conf
```

▷링크테스트를 위한 파일 생성

```
#cd /work
```

```
#vi run.sh (--->간단한 형태의 스크립트 파일)
```

```
date
```

```
cal
```

```
#ll run.sh
```

```
#./run.sh
```

```
---> X
```

```
#chmod u+x run.sh
```

```
#ll run.sh
```

```
-rwxr--r-- 1 root root 9 Aug 12 14:22 run.sh
```

스크립트 파일의 경우 r 과 x 권한이 있어야

일반계정도 실행할 수 있고

루트의 경우에는 실행권한만 있어도 된다.

```
////////////////////////////////////
```

캡춰 결과

```
[root@cent work]# ll run.sh
```

```
-rw-r--r-- 1 root root 9 10월 8 00:08 run.sh
```

```
[root@cent work]# chmod u+x run.sh
```

```
[root@cent work]# ./run.sh
```

```
////////////////////////////////////
```

◆하드 링크 생성 방법

ln 원본파일명 링크파일명

```
#ll run.sh
```

```
-rwxr--r-- 1 root root 9 7월 15 13:14 run.sh
```

```
#
```

```
#ln run.sh good.sh
```

```
#ll
```

합계 8

```
-rwxr--r-- 2 root root 9 1월 7 12:31 good.sh
```

```
-rw-r--r-- 1 bin root 0 1월 7 11:50 hello.exe
```



```
-rwxr--r-- 2 root root 9 1월 7 12:31 run.sh
```

```
#
```

```
#!/good.sh (run.sh 를 실행한 것과 동일)
```

```
#ll run.sh good.sh
```

```
-rwxr--r-- 2 root root 9 Aug 12 14:22 good.sh
```

```
-rwxr--r-- 2 root root 9 Aug 12 14:22 run.sh
```

```
#
```

```
#ls -il (아이노드 넘버 출력 ls 옵션 : -i)
```

```
합계 8
```

```
376145 -rwxr--r-- 2 root root 9 1월 7 12:31 good.sh
```

```
376143 -rw-r--r-- 1 bin root 0 1월 7 11:50 hello.exe
```

```
376145 -rwxr--r-- 2 root root 9 1월 7 12:31 run.sh
```

```
#
```

```
---> 하드링크 파일은 아이노드가 똑 같다.
```

```
복사한 경우에는 아이노드가 틀리다.
```

◆하드링크 예

```
#ls -il /usr/bin/unzip /usr/bin/zipinfo /bin/ls
```

```
#chmod g-r,o-r good.sh
```

```
#ll run.sh good.sh (또는 ll *.sh)
```

```
-rwx----- 2 root root 9 Aug 12 14:22 good.sh
```

```
-rwx----- 2 root root 9 Aug 12 14:22 run.sh
```

```
#
```

```
하드링크는 아이노드 넘버가 동일하므로 둘 다 똑같이 출력된다.
```

```
#rm run.sh
```

```
rm: remove regular file `run.sh'? y
```

```
#ls
```

```
ex3.txt good.sh hell.exe hello.exe services
```

```
#!/good.sh (원본을 지워도 하드링크 파일로 실행할 수 있다.)
```

```
#ll good.sh
```

```
(원본을 삭제했으므로 링크카운트가 2에서 1로 감소)
```

```
-rwx----- 1 root root 9 Aug 12 14:22 good.sh
```

```
#
```

◆심볼릭 링크 생성 : -s 옵션 사용

```
ln -s 원본파일명 링크파일명
```

-하드링크는 파티션이 다르면 사용할 수 없고
심볼릭 링크는 파티션이 달라도 사용할 수 있다.
-심볼릭 링크는 파일 과 디렉토리 모두 링크 시킬 수 있다.
(하드링크는 파일만 링크가능)

```
#cd /work
#pwd
/work
#ll *.sh
#ln -s good.sh go
#
```

```
#ll good.sh go
lrwxrwxrwx 1 root root 7 Aug 12 14:47 go -> good.sh
-rwx----- 1 root root 9 Aug 12 14:22 good.sh
#
---> 심볼릭 링크파일은 l 로 시작해서 퍼미션이 모두
셋팅되어 나타나므로 실제 퍼미션은 원본 파일을 확인해야 한다.
```

```
#./go(--->good.sh 가 실행된다.)
#ls -il good.sh go
981304 lrwxrwxrwx 1 root root 7 Aug 12 14:47 go -> good.sh
981301 -rwx----- 1 root root 9 Aug 12 14:22 good.sh
#
---> 심볼릭 링크는 새로운 파일이므로 아이노드 번호가 원본과 틀리다.
```

```
#chmod u-w go
#ls -il good.sh go
981304 lrwxrwxrwx 1 root root 7 Aug 12 14:47 go -> good.sh
981301 -r-x----- 1 root root 9 Aug 12 14:22 good.sh
#
---> 원본의 퍼미션이 변경된다.
```

```
#rm good.sh
#ll
#./go
--->원본 파일을 삭제했으므로 실행되지 않는다.
```

■실습

1. /admin 폴더 생성
2. df -h, cal, date 를 실행하는 스크립트 생성 및 실행권한 주기 (/admin/admin.sh)
3. admin.sh 파일의 하드링크 생성(/admin/system.sh)
4. admin.sh 파일의 심볼릭 링크 생성(/sbin/check.sh)
5. /longnamedirectory 생성
6. /longnamedirectory/hereislong.txt 파일 생성
7. longnamedirectory 의 심볼릭 링크 생성(/lnd)
8. 테스트

```
#/admin/admin.sh
```

```
#check.sh
```

```
#cd /
```

```
#ls
```

```
#cd lnd
```

```
#pwd
```

```
#ls
```

```
////////////////////////////////////
```

```
#mkdir /admin
```

```
#cd /admin
```

```
#vi admin.sh
```

```
df -h
```

```
cal
```

```
date
```

```
#chmod u+x admin.sh
```

```
#./admin.sh
```

```
#ln admin.sh system.sh
```

```
#./system.sh
```

```
#/admin/admin.sh
```

```
#ln -s /admin/admin.sh /sbin/check.sh
```

---> 원본과 링크 파일의 디렉토리가 틀린 경우는
링크 파일 만들 때 경로를 지정해야 함

```
#ll /sbin/check.sh
```

```
#check.sh
```

```
#cd /
```

```
#mkdir longnamedirectory
```

```
#touch /longnamedirectory/hereislong.txt
#ln -s /longnamnedirectory /lnd
#ll
#cd lnd
#pwd
/lnd
---> 실제로는 /longnamedirectory 로 이동 된 것임
#ls
```

////////////////////////////////////

[ITBANK Andylec 주말 리눅스 1 과정]

■오늘의 수업내용(8일차)

패키지(package) 관리

압축 파일 관리

<패키지 관리>

패키지(Package) : 프로그램 파일이 들어있는 상자 개념

프로그램 배포 단위

레드햇 리눅스 : rpm 패키지

데비안 리눅스 : debian 패키지

rpm 패키지 이름 구성

mc-4.6.0-2.i386.rpm

--->패키지이름-버전-아키텍처

명령어형식 : rpm 옵션 패키지명

옵션)

-qa // 설치된 패키지 목록 확인 (q : query, a : all)

-e // 삭제 (e : erase)

-ivh // install

-Uvh // Upgrade

설치 및 업그레이드시 옵션)

-v // verbose, 설치과정 자세히 출력

-h, --hash // # 마크 표시, 보통 v 옵션과 같이 사용

--nodeps // 의존성을 검사하지 않음

--force // --oldpackage, --replacefiles, --replacepkgs 를 합한 옵션

--test // 실제 설치하지 않고 점검

rpm -qi // 패키지 정보 (information)

rpm -qf // 파일이 속한 패키지 출력(file)

rpm -ql // 패키지 파일 목록(list, location)

rpm -qs // 패키지 상태(status)

rpm -qc // 패키지 설정 파일(configuration)

rpm 의 단점 : 의존성 문제(먼저 설치되어 있어야 하는

패키지가 없는 경우 설치가 안 된다.)

[A 패키지]-----[B 패키지]

의존성(dependency)

yum : 인터넷 파일서버에서 패키지를 다운받아 설치해 준다.

관련 패키지가 있을 경우 한꺼번에 설치해 준다.

(의존성 문제 해결)

인터넷이 되어 사용할 수 있다.

설정파일 : /etc/yum.conf

업데이트 서버의 URL 과 기타 세부적인 설정사항

```
yum (-y) install      // 설치
yum check-update      // 업데이트 가능한 목록
yum upgrade           // 업그레이드
yum remove            // 삭제
yum info              // 정보 출력
yum list              // 패키지 목록 출력
yum grouplist         // 패키지 그룹 목록 출력
yum groupinstall      // 패키지 그룹 설치
yum search 단어       // 단어가 들어간 패키지 검색
yum localinstall 또는 yum install // CD 의 패키지를 설치할 때 사용
```

사용 예 : yum -y install mc

(-y 를 생략하면 설치시 질문이 출력된다.)

♠rpm 사용하기

(Redhat Package Manager)

#rpm -qa

리눅스 CD 에 들어있는 프로그램 형태는 rpm 패키지로

rpm -qa 하게 되면 하드디스크에 설치된 rpm 패키지 목록이

모두 출력된다.

q : query(질의하다)

a : all

```
#cd /var/lib/rpm
(패키지 데이터베이스가 있는 디렉토리)
#ls
```

패키지(프로그램) 설치 유무 확인 방법 :

```
1)rpmquery 패키지명 패키지명 ...
2)rpm -q 패키지명 패키지명 ...
3)rpm -qa | grep 패키지명
```

```
#rpmquery gzip
#rpm -q gzip
#rpm -qa | grep gzip
```

--->설치된 프로그램(패키지) 목록 중에서 gzip 관련 패키지만 필터링

gzip 패키지 삭제
#rpm -e gzip (rpm -e gzip-버전 으로 해도 동일하다.)
오류: Failed dependencies:
---> 다른 패키지와 의존성이 있어서 삭제가 안 된다.

```
#rpm -e --nodeps gzip
의존성 무시하고 삭제하기
```

또는
#rpm -e gzip --nodeps

```
#rpm -q gzip
#rpm -qa | grep gzip
gzip 패키지는 삭제했으므로 나타나지 않는다.
```

<리눅스 1 번 DVD에서 gzip 패키지를 찾은 후 /work 디렉토리로 복사하기>

```
#eject
--->마운트 해제후 CD 또는 DVD 배출 명령
```

```
#df -h
```

리눅스 1번 CD 또는 DVD 입력

```
#df -h
```

CD 마운트 확인

안되면 아래와 같이 수동 마운트

```
mount /dev/cdrom /mnt
```

```
#find /media -name "*gzip"
```

수동마운트의 경우에는 find /mnt -name "*gzip"

```
#cp 경로와파일명 /work
```

```
#cd /work
```

```
#ls
```

```
#rpm -qlp gzip-버전.i686.rpm
```

패키지 파일 안의 목록 출력

p : 패키지를 뜻하며 패키지 파일을 이용할 때는 p 옵션을 지정해야 한다.

```
#rpm -ivh gzip-버전.i686.rpm
```

(rpm -Uvh gzip-버전.i686.rpm 해도 설치된다.)

--->패키지(리눅스 프로그램) 설치하기

```
#
```

```
rpm -ivh : 설치
```

```
rpm -Uvh : 업그레이드(이전 버전이 있으면 업그레이드를 하고  
설치가 안 되어 있으면 프로그램이 설치 됨)
```

```
#rpm -q gzip
```

```
#rpm -qa | grep gzip
```

gzip 패키지가 설치되었으므로 출력된다.

```
#rpm -qs gzip
```

패키지 상태 출력

```
////////////////////////////////////
```

```
#man rpm
```

```
-s      패키지 안에 든 화일의 상태를 보여준다.(-l은 포함) 각  
화일의 상태는 normal(정상), not installed(설치 되 지
```


않음), replaced 다른 것으로 교체됨)의 값을 갖는다.

////////////////////////////////////

```
#rpm -ql gzip
```

설치된 특정 패키지의 파일목록 출력

설치가 된 후에는 패키지 목록 출력시 -p 옵션이 필요없고

rpm 확장자를 지정하면 안 된다.

```
#rpm -qi gzip
```

패키지 정보 확인

♠yum 사용하기

(인터넷의 파일서버를 이용하는 방식)

[리눅스 서버] <----- yum -----> [인터넷의 CentOS 프로그램 서버]

```
#rpm -qa | grep mc
```

mc 패키지(프로그램) 확인

```
#rpm -qa | grep '^mc-'
```

```
#yum -y install mc
```

mc 패키지 설치하기

-y 옵션을 생략하면 질문 나올 때 y 를 입력해 주어야 설치된다.

```
#rpm -q mc
```

mc 패키지 설치됨

```
#mc
```

mc 는 유틸리티 프로그램

종료는 exit 입력

```
#yum info mc
```

Setting up repositories

Reading repository metadata in from local files

Installed Packages

Name : mc

Arch : i386

Epoch : 1

Version: 4.6.1

Release: 0.8.5

Size : 4.2 M

Repo : installed

Summary: User-friendly text console file manager and visual shell.

Description:

Midnight Commander is a visual shell much like a file manager, only with many more features. It is a text mode application, but it also includes mouse support if you are running GPM. Midnight Commander's best features are its ability to FTP, view tar and zip files, and to poke into RPMs for specific files.

#

#yum remove mc

-y 옵션을 지정하지 않았으므로 질문이 출력된다.

(질문에 y 를 입력한다.)

rpm -e mc 로 삭제해도 된다.

yum 을 이용하면 의존성 있는 패키지까지 함께 삭제 된다.

#rpm -q mc

♠KDE Desktop

리눅스 윈도우 매니저

#yum grouplist

#LANG=en_US.UTF-8

---> 영문 설정

#yum grouplist

KDE Desktop

#yum -y groupinstall "KDE Desktop"

---> 쿼팅 반드시 사용

-루트 로그아웃

-root 아이디 입력후 하단의 메뉴에서 KDE 선택

-암호 입력후 로그인

☞ GUI 를 이용한 패키지 관리

- 1)리눅스 메뉴표시줄-시스템-관리-소프트웨어 추가/제거
- 2)#gpk-application

♠ 실습

1. gedit(메모장 프로그램) 패키지를 rpm 명령 이용하여 설치 확인후 rpm 명령 이용하여 제거 하기
2. 1번 DVD 이용하여 gedit 패키지를 /work 에 복사하기
(gedit-plugins 는 라이브러리로 설치할 필요 없음)
3. /work 디렉토리로 이동하여 gedit 패키지 설치하기
4. yum 명령 이용하여 gedit 제거하기
5. yum 명령 이용하여 gedit 설치하기
6. 웹브라우저 firefox 업그레이드
7. Web browser 패키지 검색하기

```
////////////////////////////////////  
#yum list firefox  
#yum upgrade firefox  
#yum search "Web browser"  
#yum install elinks  
#elinks  
////////////////////////////////////
```

<묶음과 압축 명령어>

- 1) tar(압축 기능은 없고 여러 개의 파일을 하나로 묶는 역할을 함)
 압축 압축해제
- 2) gzip gunzip
- 3) bzip2 bunzip2

※tar 에서 gzip, bzip2 옵션이 지원 된다.

♠ tar 명령어 형식

tar 옵션(-는 생략가능) 대상파일 원본파일
(cp 와는 반대이다.)

☞tar 는 압축은 하지 않고 여러 파일을 하나로 묶어주거나 해제하는 프로그램

☞tar 는 옵션을 줄 때 - 를 생략할 수 있다.

☞ 옵션의 순서는 변경해도 된다.(- 사용시 f 옵션은 마지막에 사용)

☞ zxf 대신 xvf 를 사용해도 결과는 똑같다.

z : gzip 사용
x : extract (타르 해제)
v : verbose (자세히 출력)
f : file 지정
t : list 출력
c : create (타르 파일 생성)
j : bzip2 사용

////////////////////////////////////

#man tar

NAME

tar - The GNU version of the tar archiving utility

Operations:

[-]A --catenate --concatenate
[-]c --create
[-]d --diff --compare
[-]r --append
[-]t --list
[-]u --update
[-]x --extract --get
--delete

Common Options:

-C, --directory DIR
-f, --file F
-j, --bzip2
-p, --preserve-permissions
-v, --verbose
-z, --gzip

-p, --same-permissions, --preserve-permissions
extract all protection information

////////////////////////////////////

////////////////////////////////////

<타르 파일과 타르.압축 파일의 생성 형식>

1)tar cvf 파일명.tar

--->tar

2)tar cvzf 파일명.tar.gz

--->tar + gzip

3)tar cvjf 파일명.tar.bz2

--->tar + bzip2

<타르 파일과 타르.압축 파일의 해제 형식>

1)tar xvf 파일명.tar

2)tar xvzf 파일명.tar.gz

3)tar xvjf 파일명.tar.bz2

압축된 tar 파일도

tar tvf 파일명.tar.gz

tar xvf 파일명.tar.gz 로 사용 가능

////////////////////////////////////

♠tar 와 gzip 테스트

```
      /
      |
+-----+-----+
tmp                boot
```

#tar -cvf /tmp/my1.tar /boot

--->/boot 폴더의 모든 내용을 /tmp/my1.tar 파일로 저장

---> 점(.) 으로 시작하는 파일도 모두 저장 됨

#ls /tmp/my*

#tar c /tmp/my2.tar /boot

---> f 옵션을 생략하면 화면상에 결과가 출력된다.

터미널의 프롬프트가 깨지면

터미널을 종료후 다시 실행하면 된다.

```
#ls /tmp/my*
```

---> f 옵션을 생략해서 파일이 생성되지 않는다.

```
#tar cf /tmp/my3.tar /boot
```

---> v 옵션을 생략하면 처리과정이 자세히 출력되지 않는다.

```
#ls /tmp/my*
```

```
#tar cvf /tmp/my5.tar /boot
```

---> 옵션 사용시 - 생략 가능

```
#ls /tmp/my*
```

```
#tar cvzf /tmp/boot.tar.gz /boot
```

또는

```
#tar -cvzf /tmp/boot.tar.gz /boot
```

---> /boot 디렉토리의 모든 내용을 /tmp 디렉토리에 boot.tar.gz 파일로 저장

---> tar 로 묶고 gzip 으로 압축이 한꺼번에 된다.

tar 옵션

z : gzip 사용

c : create

v : verbose

f : file

j : bzip2 사용

t : 목록 출력

```
#cd /tmp
```

```
#ll bo* my*
```

```
#ls -hl bo* my*
```

```
#tar tvzf boot.tar.gz
```

t : 파일이 풀리는 것이 아니라 내용을 확인하는 옵션

tar 파일의 내용을 볼 때는 #tar tvf test.tar 형식을 사용

tar.bz2 파일의 내용을 볼 때는 #tar tvjf test.tar.bz2

```
#ls
```

※ tar.gz 파일은 gunzip 으로 압축을 풀고
tar xvf 로 tar 파일을 풀 수도 있지만
tar xvzf 옵션을 이용하면 압축과 tar 해제가 한 번에 된다.

```
#tar xvzf boot.tar.gz
기본적으로 현재 디렉토리에 파일이 풀린다.
#ls -d bo*
boot
```

```
#tar xvzf boot.tar.gz
#tar xvzf boot.tar.gz -k
기본적으로 덮어쓰기 하므로 기존 파일을 보존하려면 -k 옵션을 이용한다.
```

```
#man tar
-k 옵션
```

```
-k, --keep-old-files
    keep existing files; don't overwrite them from archive
```

```
#pwd
/tmp
#ls -d bo*
boot.tar.gz
```

```
#gunzip boot.tar.gz
---> 압축해제
#ls -d bo*
boot.tar
```

```
#tar xvf boot.tar (타르 풀기)
---> 기본적으로 현재 디렉토리에 풀리고 다른 디렉토리에
    풀려면 "-C 경로" 를 지정하면 된다.
```

```
#mkdir /mywork
#tar xvf boot.tar -C /mywork
---> tar 파일이 /mywork 디렉토리에 풀린다.
```

---> -C (Change) 는 대문자이다.

```
#ls /mywork
```

♠tar 와 bzip2 테스트

```
#tar cvf /tmp/boot2.tar /boot
```

---> /boot 디렉토리의 모든 내용을 /tmp 의 boot2.tar 파일로 저장하기

---> 압축옵션이 없으므로 압축은 되지 않는다.

c : create

v : verbose

f : file

```
#pwd
```

```
/tmp
```

```
#ls -dl bo*
```

```
#bzip2 boot2.tar
```

--->bzip2 로 압축(확장자 bz2 파일 생성)

```
#gzip my1.tar
```

--->gzip 으로 압축(확장자 gz 파일 생성)

```
#ls -dl bo* my*
```

```
#ls -dhl bo* my*
```

```
boot.tar
```

```
boot2.tar.bz2
```

```
my1.tar.gz
```

(h 옵션을 주면 킬로바이트,메가바이트 형태로 출력 된다.)

```
#bunzip2 boot2.tar.bz2
```

---> 압축해제

```
#ls -dhl bo*
```

```
boot.tar
```

```
boot2.tar
```

```
#tar cvjf /tmp/boot3.tar.bz2 /boot
```

---> tar 로 묶고 bzip2 로 압축

```
#pwd
```



```
/tmp
#ls -dhl bo*
boot.tar
boot2.tar
boot3.tar.bz2
```

```
#tar xvjf boot3.tar.bz2
---> 압축 해제하고 타르 풀기
```

```
#pwd
/tmp
#ls -dhl bo*
boot.tar boot2.tar boot3.tar.bz2
```

```
#gzip boot2.tar
--->gzip 으로 타르 파일 압축
#ls -dhl bo*
boot.tar boot2.tar.gz boot3.tar.bz2
```

```
#file boot.tar
#file boot2.tar.gz
#file boot3.tar.bz2
---> file 명령을 이용하여 파일의 종류를 확인 할 수 있다.
```

☞ GUI 를 이용한 tar 파일 만들기
/work 의 txt 파일을 /tmp/fedora.tar.gz 로 만들기

#file-roller

- 1) 위쪽 메뉴아이콘에서 "새로 만들기" 클릭
- 2) 위쪽의 이름: fedora 입력(확장자는 제외)
- 3) 압축 종류 에서 "gzip 으로 압축된 Tar" 선택
- 4) "다른 폴더 찾아보기" 클릭
- 5) 왼쪽의 파일시스템 더블클릭하고 오른쪽에서 /tmp 선택
- 6) "만들기" 버튼 클릭
- 7) 새로운창이 나타나면
"더하기" 버튼을 누르고 원하는 파일을 선택후 종료하면 됨
복수 파일을 선택할 때는 Ctrl 키 사용

■실습

1. /bin 디렉토리의 모든 파일을 gzip 압축 및 tar 로 묶어서 /home 디렉토리 안에 binlist.tar.gz 라는 파일로 생성
2. binlist.tar.gz 압축만 풀기
3. binlist.tar 에 /etc/passwd 파일 추가하기(옵션확인)
4. 리눅스 웹브라우저 실행후 아래 주소 입력
http://httpd.apache.org/download.cgi

httpd-2.0.65.tar.bz2

httpd-2.0.65.tar.gz

5. 위의 파일을 다운받고(기본적으로 /root/다운로드 폴더에 다운 됨)
/work, /work2 디렉토리 없으면 생성하기
#mkdir /work /work2

6. /work 에 httpd-2.0.65.tar.gz 를 이동시키고 압축 및 타르 풀기
7. /work2 에 httpd-2.0.65.tar.bz2 를 이동시키고 압축 및 타르 풀기

```
////////////////////////////////////  
#tar cvzf /home/binlist.tar.gz /bin  
#cd /home  
#ls  
#gunzip binlist.tar.gz  
#tar --help | grep append  
-r , --append  
#tar rvf binlist.tar /etc/passwd  
#tar tvf binlist.tar
```

```
////////////////////////////////////
```

<배치 프로세스>

- 일련의 작업을 몰아서 특정 시간에 실행시키는 것이며 터미널과의 입/출력 교류가 없다.
- 중요하지 않은 작업을 시스템 사용률이 낮을 때 처리하는데 유용
- at 와 batch 명령어를 사용

at : 지정한 시간에 명령어 수행

batch : 시스템 부하 수준이 0.8 이하 일 때 명령 수행

#man at

at 지정한 시간에 명령을 실행한다.

예를 들어, 지금부터 3일후 오후 4시에 작업을 수행하려면 at 4pm + 3 days,

7 월 31일 오전 10:00에 작업을 수행하려면 at 10am Jul 31

내일 오후 1시에

작업을 수행하려면 at 1pm tomorrow 과 같이 하면 된다.

at 커맨드 형식 : at 옵션 TIME

#mkdir /backup

#mkdir /work19

#cp /etc/hosts /work19

#rdate -s time.bora.net

#date

#at 시간:분 today

at> /bin/tar cvzf /backup/at.tar.gz /work19

at> touch /attest.txt

at> <EOT> (---> ctrl + d 를 입력)

---> 리눅스 가상머신의 시간을 확인해서 현재 시간보다 3분 뒤로 설정하기

예약작업 대기목록 확인하기

#atq

#at -l

#at -c 1

---> 1 은 atq 에서 출력된 예약 작업 번호를 적는다.

////////////////////////////////////

#man at

-m 출력 결과가 없더라도 작업이 완료될 때 사용자에게 메일을 보낸다.

-f file 표준 입력이 아닌 file 에서 작업 명령을 읽는다.

-l atq와동일

-d atrm과동일

-c 명령행에 나열된 작업들을 표준 출력으로 보여준다.

////////////////////////////////////

#atq

---> 예약작업이 실행되었으면 출력되는 것이 없다.

(예약작업 수동 삭제는 #atrm 예약번호)

#cd /backup

#ls

at.tar.gz 생성 확인