redhat CentOS

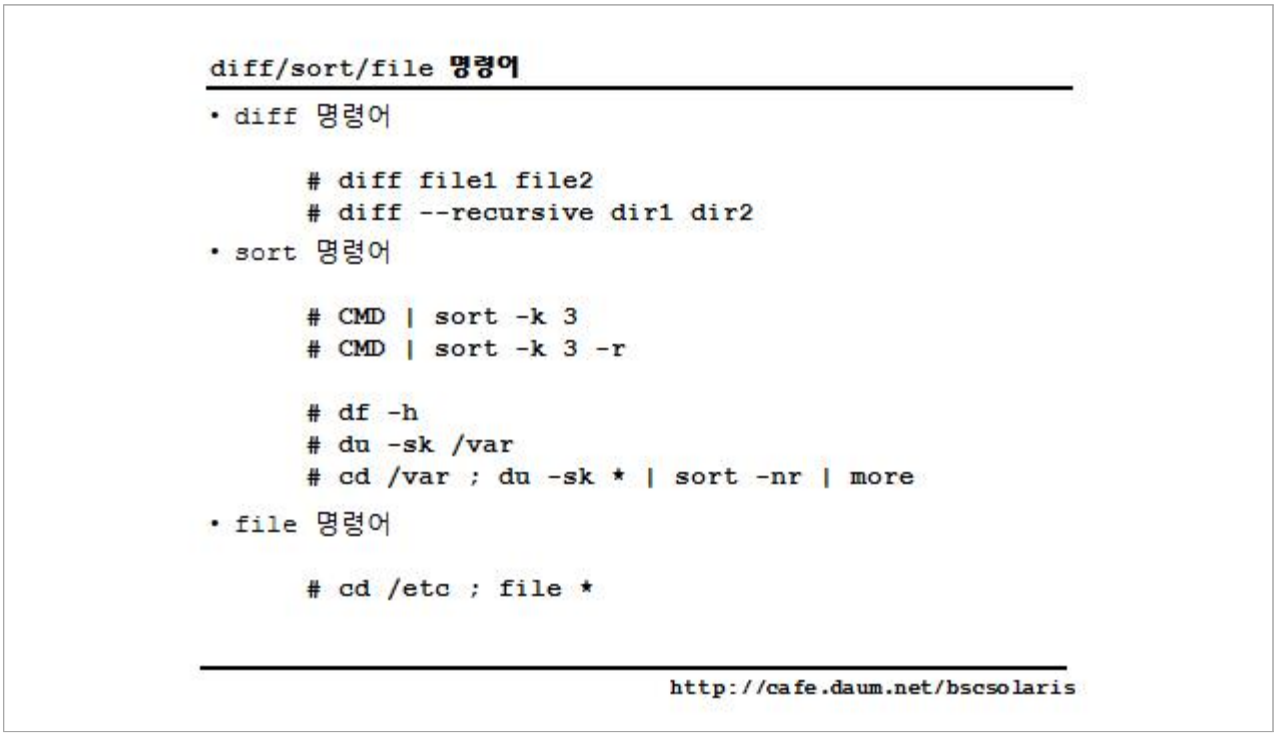
Unit 10
Usefull Commands

http://cafe.daum.net/bscsolaris

단원 목표

- cmp/diff 명령어
- sort 명령어
- file 명령어

http://cafe.daum.net/bscsolaris



1 cmp / diff CMD

(1) cmp 명령어

```
NAME
       cmp - compare two files byte by byte

DESCRIPTION
       Compare two files byte by byte.

       The optional SKIP1 and SKIP2 specify the number of bytes to skip at the
       beginning of each file (zero by default).

       Mandatory arguments to long options are  mandatory  for  short  options
       too.
       -s, --quiet, --silent
              suppress all normal output
```

서로 다른 파일을 비교하여 다른 내용을 출력해 주는 명령어이다. cmp(compare), diff(different)

[명령어 형식]

```
# cmp file1 file2          /* 두 개의 파일에 대한 비교점(차이점의 시작 정도) 확인 */
```

[명령어 옵션]

옵 션	설 명
-l	두 화일 내용을 비교함에 있어 틀린곳마다 byte 수 (10진수)와 틀린 byte 수(8진수)를 출력
-s	틀린 화일의 내용을 출력하지 않고 return code 만 변환한다

[실습] cmp 명령어 실습

■ 실습 시스템
- server1

■ 실습시나리오
- cmp 명령어를 사용한 파일 비교 실습

[EX1] cmp 명령어를 사용한 파일 비교 실습
① 3개의 실습용 파일 생성
[root@server1 ~]# cd /test
[root@server1 /test]# vi cmpfile1

```
11111
22222
33333
```

[root@server1 /test]# vi cmpfile2

```
11111
22222
33333
```

[root@server1 /test]# vi cmpfile3

```
11111
22222
44444
```

② 파일 비교
[root@server1 /test]# cmp cmpfile1 cmpfile2
[root@server1 /test]#
-> 파일의 내용이 동일하면 특별한 출력 결과가 없다.

[root@server1 /test]# cmp cmpfile1 cmpfile3

```
cmpfile1 cmpfile3 differ: byte 13, line 3
```

-> 파일의 내용이 틀린 경우 틀린 부분의 시작부분을 알려준다.
(3번째 라인에 13번째 문자부터 틀린 부분이 시작된다.)

(2) diff 명령어

```
이름
       diff(different) - 두 파일에서 차이점을 찾는다.

사용법
       diff [options] from-file to-file

설명
       간단한 예로, diff 는 from-file 와 to-file 의 두파일의 내
       용을 비교한다.  - 의 파일명은 표준입력으로부터 읽어들 여
       내용을 나타낸다.  특별한 경우로, diff - - 는 자기 자신을
       표준입력으로 비교한다.

       If from-file 이 디렉토리이고, to-file 이 아니라 면  diff
       는 to-file의 파일과 from-file안의 파일을 비교한다.

       from-file  과 to-file 이 모두 디렉토리라면, diff 는 알파
       벳 순서로 두 디렉토리 안의 상응하는 파일을 비교한다;  이
       비 교는 -r이나 -recursive옵션이 주어지지 않으면 재귀적이
       아니다.  diff 는 파일인양 실제 디렉토리를 비교하지는  않
       는 다.   표준입력은 같은 이름을 가진 파일개념을 적용하지
       않기때문에 꽉찬 파일은표준입력되지 않을 수 있다.

       diff 은 -, 로 시작된다, 그래서 대게 from-file 과 to-file
       은 - 로 시작되지 않을런지 모른다.

   Options
       아 래는 그뉴 diff에서 쓰이는 모든 옵션들의 요약이다.  대
       부분의 옵션은 두개의 이름을 갖는다. 하나는 -에 앞서는 단
       일 문자이고, 다른 하나는 --에 앞서는 긴 이름이다. 복수의
       단일 문자 옵션은 단일 명령행에 결합될 수  있 다:  -ac 는
       -a-c와 같다. 긴 이름의 옵션은 그 이름의특정부분만으로 줄
       여 쓸수가 있다. 대괄호 ([ 와 ]) 은 옵션이 임의의 인자 를
       갖는다는 것을 가리킨다.

       -c     내용 출력폼을 사용한다.

       -i     Ignore  changes  in  case;  consider  upper-  and
              lower-case letters equivalent.

       --recursive
              두  디렉토리를 비교할때, 모든 서브디렉토리는 재귀
              적으로(recursively) 비교한다.
```

[명령어 형식]

```
# diff file1 file2        /* 두 개의 파일에 대한 간략한 차이점 */
# diff -c file1 file2     /* 두 개의 파일에 대한 자세한 차이점 확인 */
# diff -i file1 file2     /* -i : 대소문자를 구분하지 말아라. A와 a는 같은 것으로 간주 */
# diff -r dir1 dir2
```

[명령어 옵션]

옵션	설 명
--brief	두 파일의 내용이 같은지 다른지 알아봄
-c	파일의 이름, 날짜 등 파일의 차이점을 상세히 출력
-d	두 디렉토리간의 차이점 출력
-H	용량이 큰 파일 비교 시 속도를 빠르게 비교
-i	대소문자 구분하지 않음

[실습] diff 명령어 실습

■ 실습 시스템
- server1

■ 실습 시나리오
- diff 명령어를 사용하여 2개의 파일의 내용을 비교한다.
- diff 명령어를 사용하여 2개의 디렉토리 안의 모든 파일을 비교한다.

[EX1] 파일 비교

① 3개의 파일 확인
```
[root@server1 ~]# cd /test
[root@server1 /test]# cat cmpfile1
```
```
11111
22222
33333
```

```
[root@server1 /test]# cat cmpfile2
```
```
11111
22222
33333
```

```
[root@server1 /test]# cat cmpfile3
```
```
11111
22222
44444
```

② 파일 비교
```
[root@server1 /test]# diff cmpfile1 cmpfile2
[root@server1 /test]#
```
- 2개의 파일 내용이 같으면 특별한 출력 결과가 없다.

```
[root@server1 /test]# diff cmpfile1 cmpfile3
```
```
3c3
< 33333
---
> 44444
```
- 2개의 파일의 틀린 내용 부분만을 보여준다.

```
[root@server1 /test]# diff -c cmpfile1 cmpfile3
```
```
*** cmpfile1    Wed Jan 27 00:54:39 2010
--- cmpfile3    Wed Jan 27 00:55:05 2010
***************
*** 1,3 ****
  11111
  22222
! 33333
--- 1,3 ----
  11111
  22222
! 44444
```
- 2개 파일의 전체 내용을 두고, 틀린 부분을 보여준다.

[EX2] 대소문자 구별 없이 파일 비교

① 2개의 파일 생성
[root@server1 /test]# vi file1

```
11111
22222
iiiii
```

[root@server1 /test]# vi file2

```
11111
22222
IIIII
```

② 2개의 파일 비교
[root@server1 /test]# diff file1 file2

```
3c3
< iiii
---
> IIII
```

[root@server1 /test]# diff -i file1 file2 (-i : ignore case, 대소문자 구분하지 않음)
[root@server1 /test]#

[EX3] 디렉토리 비교시(--recursive, -r 옵션 사용)

① 실습 준비
[root@server1 /test]# cd /test && rm -rf /test/*

[root@server1 /test]# mkdir dir1
[root@server1 /test]# vi dir1/file1

```
1111
```

[root@server1 /test]# mkdir dir2
[root@server1 /test]# vi dir2/file1

```
2222
```

[root@server1 /test]# tree

```
.
|-- dir1/
|   `-- file1
`-- dir2/
    `-- file1
```

② 2개의 디렉토리 비교
[root@server1 /test]# diff -r dir1 dir2 (# diff --recursive dir1 dir2)

```
diff --recursive dir1/file1 dir2/file1
1c1
< 1111
---
> 2222
```

③ dir2/file1 내용 변경 후 2개의 디렉토리 비교
[root@server1 /test]# vi dir2/file1

```
1111                    <----- '2222 -> 1111' 변경
```

[root@server1 /test]# touch dir2/file2
[root@server1 /test]# diff -r dir1 dir2

```
Only in dir2: file2
```

[실무예] 파일 비교

```
(원 본) ------------------> (백업 본)
/source/* (server.xml)       /source/* (server.xml.old)
```

```
# diff server.xml server.xml.OLD
```

[EX] /etc/httpd/conf/httpd.conf 파일

```
# cd /test && rm -rf /test/*
# cp -p /etc/httpd/conf/httpd.conf /test

# cp -p httpd.conf httpd.conf.OLD
# vi httpd.conf
/ServerName -> n
```

[수정전]

```
#ServerName www.example.com:80
```

[수정후]

```
ServerName 192.168.10.20:80
# diff httpd.conf httpd.conf.OLD
```

[실무예] 디렉토리 마이그레이션(이관작업, Migration) 작업

```
/was1/*  ---- Migration ----> /was2/*
            (백업&복구)
    # rsync -a --delete /was1/ /was2/
```

디렉토리 마이그레이션 작업, 이후에 확인 작업

(ㄱ) 파일의 비교하는 경우

```
# find /was1 | wc -l
# find /was2 | wc -l
```

(ㄴ) 파일 각각을 비교하는 경우

```
# diff -r /was1 /was2 (# diff --recursive /was1 /was2)
```

2 sort CMD

```
NAME
       sort - sort lines of text files

DESCRIPTION
       Write  sorted  concatenation  of all FILE(s) to standard
       output.

       Mandatory arguments to long options  are  mandatory  for
       short options too.  Ordering options:

       -d, --dictionary-order
              consider only blanks and alphanumeric characters

       -f, --ignore-case
              fold lower case to upper case characters

       -n, --numeric-sort
              compare according to string numerical value

       -r, --reverse
              reverse the result of comparisons

       -k, --key=POS1[,POS2]
              start a key at POS1, end it at POS2 (origin 1)
```

파일의 정렬. 하나 또는 그 이상의 파일의 텍스트 줄을 스크린 상에서 정렬 하고자 할 때 사용한다. 즉 출력 내용을 정렬하여 표현하고자 할 때 사용한다. sort 명령어는 아무런 옵션 없이 사용되면 숫자나 알파벳 순으로 정렬하여 준다. 기본 적으로 sort 명령어는 공백 문자(white space: space, Tab)를 필드 구분자로 인식한다.

■ 정렬 기준(default)
- 오름차순 정렬 <---> (-r) 내림차순 정렬(reverse)
- 1필드/2필드/3필드 순으로 정렬 <---> (-k) 필드 지정 정렬(key)
- 문자열 정렬 <---> (-n) 숫자열 정렬(numeric)

[명령어 형식]

```
# sort /etc/passwd
# sort -r /etc/passwd              /* -r : reverse sort */
# sort -k 3 filename               /* -k : key */
# sort -t : -k 3 -n /etc/passwd    /* -t : seperate, -n : numeric */
```

[명령어 옵션]

옵션	설 명
-n	숫자로 정열한다.
-r	내림차순으로 정렬한다. 기본은 오름차순으로 정렬하는 것이다.
-o	출력 결과를 파일에 저장한다.
-t	필드 구분자를 지정한다. (기본값은 공백이 기준이 된다)
-k	정렬할 필드를 지정한다.

[실습] sort 명령어를 사용한 출력 결과 정렬

■ 실습 시스템
- server1

■ 실습 시나리오
- sort 명령어를 사용한 출력결과 정렬하기
- sort 명령어를 사용한 응용 예

[EX1] sort 명령어를 사용한 출력결과 정렬하기

① 실습 준비
```
[root@server1 ~]# cd /test
[root@server1 /test]# vi sortfile1
```
```
linux100 10 20 31 50
linux200 20 25 31 20
linux300 30 20 30 40
linux400 50 20 30 80
```

② sortfile1 정렬/역정렬

■ **정렬 기준(default)**
- 1필드->2필드->3필드-> ... 정렬 <---> (-k) 필드 지정 정렬
- 오름차순 정렬 <---> (-r) 내림차순 정렬
- 문자열 정렬 <---> (-n) 숫자열 정렬

```
[root@server1 /test]# sort sortfile1          /* 첫 번째 필드를 기준으로 정렬 */
```
```
linux100 10 20 31 50
linux200 20 25 31 20
linux300 30 20 30 40
linux400 50 20 30 80
```

```
[root@server1 /test]# sort -r sortfile1       /* -r: reverse sort, 내림차순으로 정렬 */
```
```
linux400 50 20 30 80
linux300 30 20 30 40
linux200 20 25 31 20
linux100 10 20 31 50
```

③ 3번째 필드를 기준으로 정렬하기
```
[root@server1 /test]# sort -k 3 -n sortfile1    /* 세 번째 필드를 기준으로 숫자로 정렬 */
```
```
linux100 10 20 31 50
linux300 30 20 30 40
linux400 50 20 30 80
linux200 20 25 31 20
```

④ /etc/passwd 파일의 첫번째/세번째 필드를 중심으로 정렬하기

* 기본적으로 정렬시 필드와 필드의 구분: 공백 문자(White Space)
* 필드 구분자를 직접 지정: -t :

```
[root@server1 /test]# sort -t : -k 1 /etc/passwd /* -t (seperate) 구분자로 :사용, 첫번째 행 정렬 */
```
```
adm:x:3:4:adm:/var/adm:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
bin:x:1:1:bin:/bin:/sbin/nologin
chrony:x:992:986::/var/lib/chrony:/sbin/nologin
clevis:x:977:976:Clevis Decryption Framework unprivileged user:/var/cache/clevis:/sbin/nologin
..... (중략) .....
```

```
[root@server1 /test]# sort -t : -k 3 -n /etc/passwd
```
```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

```
sync:x:5:0:sync:/sbin:/bin/sync
..... (중략) .....
```

[EX2] sort 명령어 활용

일반적인 sort 명령어 사용

```
# CMD | sort
# CMD | sort -r
# CMD | sort -nr

# ps -ef | head | sort
# ps -ef | head | sort -r
# ps -ef | head | sort -k 3
# ps -ef | head | sort -k 3 -r

# df -h | sort -k 4
# df -h | sort -k 4 -r
```

① 파일 시스템 디스크 사용량 점검

```
[root@server1 ~]# df -h              /* df: disk free space, -h: human readable */
```

```
Filesystem           1K-blocks    Used Available Use% Mounted on
devtmpfs               886040       0    886040   0% /dev
tmpfs                  916500       0    916500   0% /dev/shm
tmpfs                  916500    9828    906672   2% /run
tmpfs                  916500       0    916500   0% /sys/fs/cgroup
/dev/mapper/cl-root  40117108 5143000  34974108  13% /
/dev/mapper/cl-home  19585024  183192  19401832   1% /home
/dev/sda1             1038336  245156    793180  24% /boot
tmpfs                  183300    1180    182120   1% /run/user/42
tmpfs                  183300    4660    178640   3% /run/user/0
```

② /var 디렉토리 용량 확인

```
[root@server1 ~]# du -sk /var        /* du: disk usage, -s: sum, -k: Kbytes */
```

```
656832    /var
```

③ /var 디렉토리 안의 내용에 대한 각각의 용량 확인

```
[root@server1 ~]# cd /var
[root@server1 ~]# du -sk * | sort
```

```
.... (중략) ....
0    run
0    www
0    yp
12   spool
15180    log
171872   lib
469756   cache
4    tmp
```

```
[root@server1 ~]# du -sk * | sort -n          /* 첫번째 열을 숫자로 정렬 */
```

```
.... (중략) ....0 lock
0    run
0    www
0    yp
4    tmp
12   spool
15180    log
171872   lib
469756   cache
```

```
[root@server1 ~]# du -sk * | sort -nr | more      /* 첫번째 필드열 역순으로 정렬 */
```

```
469756   cache
171872   lib
15180    log
12   spool
4    tmp
0    yp
0    www
.... (중략) ....
```

3 file CMD

```
NAME
       file - determine file type

SYNOPSIS
       file [ -bchikLnNprsvz ] [ -f namefile ] [ -F separator ]
       [ -m magicfiles ] file ...
       file -C [ -m magicfile ]

DESCRIPTION
     This manual page documents version 5.33 of the file command.

     file tests each argument in an attempt to classify it.  There are three
     sets of tests, performed in this order: filesystem tests, magic tests,
     and language tests.  The first test that succeeds causes the file type to
     be printed.

     The type printed will usually contain one of the words text (the file
     contains only printing characters and a few common control characters and
     is probably safe to read on an ASCII terminal), executable (the file con-
     tains the result of compiling a program in a form understandable to some
     UNIX kernel or another), or data meaning anything else (data is usually
      "binary"  or non-printable).  Exceptions are well-known file formats (core
     files, tar archives) that are known to contain binary data.  When modify-
     ing magic files or the program itself, make sure to preserve these
     keywords.  Users depend on knowing that all the readable files in a
     directory have the word  "text"  printed.  Don't do as Berkeley did and
     change  "shell commands text"  to  "shell script" .

     The magic tests are used to check for files with data in particular fixed
     formats.  The canonical example of this is a binary executable (compiled
     program) a.out file, whose format is defined in <elf.h>, <a.out.h> and
     possibly <exec.h> in the standard include directory.  These files have a
      "magic number"  stored in a particular place near the beginning of the
     file that tells the UNIX operating system that the file is a binary exe-
     cutable, and which of several types thereof.  The concept of a  "magic"
     has been applied by extension to data files.  Any file with some invari-
     ant identifier at a small fixed offset into the file can usually be
     described in this way.  The information identifying these files is read
     from the compiled magic file /usr/share/misc/magic.mgc, or the files in
     the directory /usr/share/misc/magic if the compiled file does not exist.
     In addition, if $HOME/.magic.mgc or $HOME/.magic exists, it will be used
     in preference to the system magic files.

     If a file does not match any of the entries in the magic file, it is
     examined to see if it seems to be a text file.  ASCII, ISO-8859-x, non-
     ISO 8-bit extended-ASCII character sets (such as those used on Macintosh
     and IBM PC systems), UTF-8-encoded Unicode, UTF-16-encoded Unicode, and
     EBCDIC character sets can be distinguished by the different ranges and
     sequences of bytes that constitute printable text in each set.  If a file
     passes any of these tests, its character set is reported.  ASCII,
     ISO-8859-x, UTF-8, and extended-ASCII files are identified as  "text"
     because they will be mostly readable on nearly any terminal; UTF-16 and
     EBCDIC are only  "character data"  because, while they contain text, it is
     text that will require translation before it can be read.  In addition,
     file will attempt to determine other characteristics of text-type files.
     If the lines of a file are terminated by CR, CRLF, or NEL, instead of the
     Unix-standard LF, this will be reported.  Files that contain embedded
     escape sequences or overstriking will also be identified.

     Once file has determined the character set used in a text-type file, it
     will attempt to determine in what language the file is written.  The lan-
     guage tests look for particular strings (cf.  <names.h>) that can appear
     anywhere in the first few blocks of a file.  For example, the keyword .br
     indicates that the file is most likely a troff(1) input file, just as the
     keyword struct indicates a C program.  These tests are less reliable than
     the previous two groups, so they are performed last.  The language test
     routines also test for some miscellany (such as tar(1) archives).

     Any file that cannot be identified as having been written in any of the
     character sets listed above is simply said to be  "data" .
```

file 명령어는 파일의 종류(File Type)을 알 수 있는 명령어이다. 많이 사용되는 명령어는 아니지만 특별한 경우에 사용 될 수 있다. 예를 들어 인터넷 상에서 다운로드 받은 파일이 정확히 어떤 종류인지를 확인할 때 사용한다.

Unix와 리눅스에서는 확장자(Extention)는 특별한 의미를 가지고 있지 않은 경우가 대부분이기 때문이다. 인터넷상에서 다운로드 받는 대부분의 프로그램 파일들은 filename.tar.gz 형태로 되어져 있는 경우가 많다. filename.tar.gz 라는 이름은 만든 사람이 잘못 배포하는 경우 filename.tar로 배포 되는 경우가 발생할 수 있다. 이런 경우 다운로드 받은 파일의 형태를 정확히 알지 못하는 경우 프로그램을 사용할 수 없게 된다. file명령어를 사용하여 파일의 형식(File Type)을 정확히 알아서 원본 확인의 확장자(Extention)로 복구한 후 압축을 풀고 아카이브(Archive)을 풀어 사용하게 되면 해결된다.

[참고] Windos / Unix 계열의 파일 확장자에 대해서
Windows -> 파일에 확장자(예: file.txt, hwp.exe, test.bat, ...) Unix,Linux -> 파일의 확장자가 거의 의미가 없다.

[명령어 형식]

```
# file /etc/passwd
# file /etc/passwd /var
# file *
```

[실습] file 명령어 실습

■ 실습 시스템
- server1

■ 실습 시나리오
- file 명령어의 기본 사용법

[EX1] 파일의 종류 확인

① file 명령어를 사용하여 다양한 종류의 파일에 대해서 확인한다.

```
[root@server1 ~]# file /etc/passwd          /* ASCII파일 (# cat /etc/passwd) */
/etc/passwd: ASCII text

[root@server1 ~]# file /bin/ls              /* Binary 파일 (# strings /bin/ls) */
/bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter
/lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0,
BuildID[sha1]=056dda3f1b77919163a7de5563a2b9d9d245554c, stripped
```

- PE 포맷 : windows 실행 파일 형식(예: test.exe)
- ELF 포맷 : linux/unix 실행 파일 형식

```
[root@server1 ~]# file /var/log/wtmp            /* Data 파일(# last -f /var/log/wtmp) */
/var/log/wtmp: data

[root@server1 ~]# file /usr/sbin/service        /* BASH Shell Script 파일(# cat /usr/sbin/service) */
/usr/sbin/service: Bourne-Again shell script, ASCII text executable

[root@server1 ~]# file /usr/bin/lsusb.py        /* Python Script 파일(# cat /usr/bin/lsusb.py) */
/usr/bin/lsusb.py: a /usr/libexec/platform-python script, ASCII text executable
```

② 여러개의 파일 지정하기

```
[root@server1 ~]# file /etc/hosts /etc
/etc/hosts: ASCII text
/etc:       directory

[root@server1 ~]# cd /etc
[root@server1 /etc]# file *
adjtime:                    ASCII text
```

```
aliases:                ASCII text
alsa:                   directory
alternatives:           directory
anaconda:               directory
anacrontab:             ASCII text
asound.conf:            ASCII text
at.deny:                very short file (no magic)
audit:                  directory
authselect:             directory
avahi:                  directory
bash_completion.d:      directory
..... (중략) .....
```

[실무예] 확장자가 변경되는 예에 대해서

다른 종류의 운영체제 간의 파일 전송이 많은 경우, 파일의 확장자가 변경되는 경우가 있다.

WIN(file1.txt) ------> LINUX(file1.txt)

LINUX(file1.txt) ----> WIN(file1.txt)

UNIX(file1.txt) -----> WIN(file1.txt)

■ (파일 제공자)

```
# cd /test
# rm -rf /test/*

# cp -p /etc/passwd file1
# cp -p file1 file2
# cp -p file1 file3
# ls -lh

# zip all.zip file1 file2 file3
# ls -lh

# unzip -l all.zip
# rm -f file?
```

(인터넷) all.zip ---- 다운로드 ----> all (?)

```
# mv all.zip all
```

■ (파일 받기(다운로드) - 파일 사용자)

```
# file all
# mv all all.zip
# unzip all.zip
# ls -lh
```