redhat CentOS

Unit 11
Searching Commands

http://cafe.daum.net/bscsolaris

단원 목표

- grep 명령어
- find 명령어

http://cafe.daum.net/bscsolaris

grep 명령어

• grep 명령어

```
# grep OPTIONS PATTERN
OPTIONS: -l, -n, -i, -v, -w
PATTERN: *  .  ^root  root$  [abc] [^a]

# CMD | grep xinetd
# ps -ef | grep xinetd
# rpm -qa | grep talk
# cat /etc/passwd | grep root
# cat /etc/group | grep root
# chkconfig --list | grep talk

# cat /var/log/messages | \
egrep '(warn|err|crit|alert|emerg)'
```

http://cafe.daum.net/bscsolaris

1 grep CMD

```
NAME
       grep, egrep, fgrep - print lines matching a pattern
       (g/re/p, Globally/Regular Expression/Print)

SYNOPSIS
       grep [options] PATTERN [FILE...]
       grep [options] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
       grep searches for PATTERN in each FILE. A FILE of “-” stands for
       standard input. If no FILE is given, recursive searches examine the
       working directory, and nonrecursive searches read standard input. By
       default, grep prints the matching lines.

       In addition, the variant programs egrep and fgrep are the same as
       grep -E and grep -F, respectively. These variants are deprecated, but
       are provided for backward compatibility.

OPTIONS
   Matcher Selection
       -E, --extended-regexp
              Interpret PATTERN as an extended regular expression (ERE, see
              below).

       -F, --fixed-strings
              Interpret PATTERN as a list of fixed strings (instead of regular
              expressions), separated by newlines, any of which is to be
              matched.

       -G, --basic-regexp
              Interpret PATTERN as a basic regular expression (BRE, see
              below). This is the default.

       -P, --perl-regexp
              Interpret the pattern as a Perl-compatible regular expression
              (PCRE). This is experimental and grep -P may warn of
              unimplemented features.

   Matching Control
       -i, --ignore-case
              Ignore case distinctions, so that characters that differ only in
              case match each other.

       -v, --invert-match
              Invert the sense of matching, to select non-matching lines.

       -e PATTERN, --regexp=PATTERN
              Use PATTERN as the pattern. If this option is used multiple
              times or is combined with the -f (--file) option, search for all
              patterns given. This option can be used to protect a pattern
              beginning with “-”.

       -w, --word-regexp
              Select only those lines containing matches that form whole
              words. The test is that the matching substring must either be
              at the beginning of the line, or preceded by a non-word
              constituent character. Similarly, it must be either at the end
              of the line or followed by a non-word constituent character.
              Word-constituent characters are letters, digits, and the
              underscore. This option has no effect if -x is also specified.

   General Output Control
       -c, --count
              Suppress normal output; instead print a count of matching lines
              for each input file. With the -v, --invert-match option (see
              below), count non-matching lines.

       --color[=WHEN], --colour[=WHEN]
              Surround the matched (non-empty) strings, matching lines,
              context lines, file names, line numbers, byte offsets, and
              separators (for fields and groups of context lines) with escape
              sequences to display them in color on the terminal. The colors
              are defined by the environment variable GREP_COLORS. The
              deprecated environment variable GREP_COLOR is still supported,
              but its setting does not have priority. WHEN is never, always,
```

```
              or auto.

       -l, --files-with-matches
              Suppress normal output; instead print the  name  of  each  input
              file  from  which  output would normally have been printed.  The
              scanning will stop on the first match.

       -q, --quiet, --silent
              Quiet;  do  not  write  anything  to  standard  output.    Exit
              immediately  with  zero status if any match is found, even if an
              error was detected.  Also see the -s or --no-messages option.

   Output Line Prefix Control
       -n, --line-number
              Prefix each line of output with the 1-based line  number  within
              its input file.

   Context Line Control
       -A NUM, --after-context=NUM
              Print  NUM  lines  of  trailing  context  after  matching lines.
              Places a line containing  a  group  separator  (described  under
              --group-separator)  between  contiguous groups of matches.  With
              the -o or --only-matching option,  this  has  no  effect  and  a
              warning is given.

   File and Directory Selection
       -r, --recursive
              Read  all  files  under  each  directory, recursively, following
              symbolic links only if they are on the command line.  Note  that
              if   no  file  operand  is  given,  grep  searches  the  working
              directory.  This is equivalent to the -d recurse option.
```

파일 내에서 특정한 패턴을 검색하여 그 패턴을 포함하는 모든 줄을 화면에 출력하는 명령어이다. 파일 내에 특정한 패턴을 찾기 위해 많이 사용한다. 찾으려고 하는 패턴을 정규식(g/re/p, Globally/Regular Expression/Print)이라고 한다. grep 명령어의 약자에서 볼 수 있듯이 많은 패턴을 제공하고 있다.

[명령어 형식]

```
# grep OPTIONS PATTERN file1
```

(기본 사용법)

```
# grep 'root' /etc/passwd    (# cat /etc/passwd | grep root)

# CMD | grep root
# cat /etc/passwd | grep root
# rpm -qa | grep httpd
# ps -ef | grep rsyslogd
# netstat -antup | grep :22
# systemctl list-unit-files | grep ssh
```

(옵션 사용법) "# grep OPTIONS PATTERN file1"

```
여러개의 파일중에 root 문자열이 들어 있는 파일만 출력
# grep -l root /etc/hosts /etc/passwd /etc/group /* -l, --files-with-matches */
/etc/ 디렉토리안에 모든 파일 중 문자열 ens33 존재하는 파일이름과 내용 출력
# grep -r ens160 /etc/              /* -r, --recursive */
문자열 root가 들어 있는 라인번호와 내용 출력
# grep -n root /etc/passwd          /* -n, --line-number */
문자열 root가 들어 있지 않는 내용 출력(문자열 root만 제외하고 나머지 출력)
# grep -v root /etc/passwd          /* --invert-match, 해석: 제외하고~ */
대소문자 구분하지 않고 검색
# grep -i dns /etc/passwd           /* -i, --ignore-case, upper/lower case(대/소문자) */
파일에서 문자열 root가 별도의 워드로 된경우에만 검색
# grep -w root /etc/passwd          /* -w, --word-regexp, 해석: 정확하게~ */
검색된 내용과 3줄 더 출력
# grep -A 3 root /etc/group         /* -A, --after-context */
검색된 내용 색깔 표시
# grep --color root /etc/passwd     /* --colour[=WHEN] */
```

[명령어 옵션]

옵션	설 명
-l	(-l : list files) 패턴이 있는 파일이름만을 출력한다.
-n	(-n : number line) 패턴을 포함하는 줄을 출력할 때 줄번호와 함께 출력한다.
-v	(-v : inVerse, except) 패턴을 포함하는 줄을 제외하고 출력한다.
-c	(-c : count) 패턴을 찾은 줄의 수를 출력한다.
-i	(-i : ignore case, 대문자/소문자) 패턴을 찾을 때 대소문자를 구분하지 않는다.

(패턴 사용법) "# grep OPTIONS PATTERN file1"

```
*       # grep 'roo*t' /etc/passwd          /* 앞의 문제가 0회 이상 math(rot, root, rooot, roooot, ..) */
.       # grep 'no...y' /etc/passwd         /* 1개의 문자 매치(정확히 1개의 문자와 매치) */
^root   # grep '^root' /etc/passwd          /* 문자열의 라인의 처음 */
root$   # grep 'bash$' /etc/passwd          /* 문자열 라인의 마지막 */
[abc]   # grep 'user0[123]' /etc/passwd     /* 대괄호에 포함된 문자 중 한개와 매치 */
```

[참고] 정규 표현식(Regular Expression)에 대해서

```
# man grep
/REGULAR EXPRESSIONS
# man 7 glob
# man 7 regex
```

[실습] grep 명령어 실습

■ 사용 시스템
- server1

■ 실습 시나리오
- grep 명령어 옵션 실습
- grep 명령어 패턴 실습

[EX1] grep 명령어 옵션 실습

① "grep -i" 옵션 실습
- 일반적으로 grep 명령어의 -i 옵션은 alias 선언되어서 사용되는 경우가 많다.
- -i, --ignore-case ignore case distinctions

```
[root@server1 ~]# alias grep
alias grep='grep --color=auto'
```

- alias grep='grep --color'
- alias grep='grep --color=auto'

```
# alias | grep grep
```

```
[root@server1 ~]# useradd FEDORA
[root@server1 ~]# passwd FEDORA
-> 사용자 암호 입력
```

```
[root@server1 ~]# grep fedora /etc/passwd      (# cat /etc/passwd | grep fedora)
fedora:x:1000:1000:fedora:/home/fedora:/bin/bash
```

* case sensitive <-> ignore case

```
[root@server1 ~]# grep -i fedora /etc/passwd
fedora:x:1000:1000:fedora:/home/fedora:/bin/bash
FEDORA:x:1002:1002::/home/FEDORA:/bin/bash
```

(실무 예) 환경 파일에 등록

```
# vi ~/.bashrc
```

```
alias grep='grep --color=auto -i'
# . ~/.bashrc
# alias grep
```

② "grep -n" 옵션 실습

- grep 명령어의 -n 옵션은 쉘 스크립트 라인에서 사용되는 경우가 많다.
- -n, --line-number print line number with output lines

```
[root@server1 ~]# grep -n fedora /etc/passwd
47:fedora:x:1000:1000:fedora:/home/fedora:/bin/bash
```

```
[root@server1 ~]# grep -n root /etc/passwd
1:root:x:0:0:root:/root:/bin/bash
10:operator:x:11:0:operator:/root:/sbin/nologin
```

③ grep -l 옵션 실습

- grep 명령어의 -l 옵션은 명령어 프롬프트에서 주로 사용이 되며, 특정한 패턴을 가지고 있는 파일을 이름을 검색하는 용도로 사용이 된다.
- -l, --files-with-matches print only names of FILEs with selected lines

```
[root@server1 ~]# grep -l root /etc/group /etc/passwd /etc/hosts
/etc/group
/etc/passwd
```

④ "grep -w" 옵션 실습

- grep 명령어의 -w 옵션은 명령어 프롬프트에서 주로 사용이 되고, 특정 패턴에 대한 정확한 단어를 검색하기 위한 용도로 사용된다.
- -w, --word-regexp force PATTERN to match only whole words

```
[root@server1 ~]# cd /test
[root@server1 /test]# vi file.txt
```

```
root  hello
roothello
testroottest
helloroot
```

```
[root@server1 /test]# grep -w root file.txt
root hello
```

```
[root@server1 /test]# ps -ef | grep -w rsyslogd
root       1152       1  0 Feb21 ?        00:00:09 /usr/sbin/rsyslogd -n
root      39797   25320  0 11:46 pts/0    00:00:00 grep --color=auto -w rsyslogd
```

⑤ grep -v 옵션 실습

- -v, --invert-match select non-matching lines

```
[root@server1 /test]# ps -ef | grep rsyslogd
root       1152       1  0 Feb21 ?        00:00:09 /usr/sbin/rsyslogd -n
root      39624   25320  0 11:39 pts/0    00:00:00 grep --color=auto rsyslogd
```

```
[root@server1 /test]# ps -ef | grep rsyslogd | grep -v grep
root       1152       1  0 Feb21 ?        00:00:09 /usr/sbin/rsyslogd -n
```

⑥ "grep -r" 옵션 실습

- -r, --recursive like --directories=recurse

```
[root@server1 /test]# nmcli device
DEVICE  TYPE      STATE                   CONNECTION
ens160  ethernet  connected               eth0
ens192  ethernet  connected               eth1
lo      loopback  connected (externally)  lo
```

-> DEVICE: ens160, ens192, lo

```
[root@server1 /test]# grep -r ens160 /etc
/etc/NetworkManager/system-connections/ens160.nmconnection:interface-name=ens160

[root@server1 /test]# grep ens160 /etc/NetworkManager/system-connections/ens160.nmconnection
interface-name=ens160
```

⑦ grep -A 옵션 실습

- -A, --after-context=NUM print NUM lines of trailing context

```
[root@server1 /test]# ip addr | grep -A 6 ens160:
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:b4:2f:0a brd ff:ff:ff:ff:ff:ff
    altname enp3s0
    inet 192.168.10.20/24 brd 192.168.10.255 scope global noprefixroute ens160
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feb4:2f0a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

[EX2] grep 명령어 패턴 실습

① 로그 파일에서 시간을 기준으로 기록을 검색하는 경우

```
[root@server1 /test]# cat /var/log/messages | grep -i jan
Jan 26 01:26:02 localhost syslogd 1.4.1: restart.
Jan 26 01:26:02 localhost kernel: klogd 1.4.1, log source = /proc/kmsg started.
Jan 26 01:26:02 localhost kernel: Linux version 2.6.18-164.el5 (mockbuild@builde
r16.centos.org) (gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)) #1 SMP Thu Sep 3
 03:33:56 EDT 2009
Jan 26 01:26:02 localhost kernel: BIOS-provided physical RAM map:
Jan 26 01:26:02 localhost kernel:  BIOS-e820: 0000000000010000 - 000000000009f80
0 (usable)

[root@server1 /test]# cat /var/log/messages | grep -i 'Jan 26'
```

[실무예] 로그 파일를 시간에 기반해서 검색하기

```
# cat /var/log/messages | grep 'Aug 28'
# cat /var/log/messages | grep 'Aug 28 10:'
# cat /var/log/messages | grep 'Aug 28 1[01]:'
# cat /var/log/messages | grep 'Aug 28 1[0-5]:'
# cat /var/log/messages | grep 'Aug 28 10:12:'
```

② 현재 프로세스의 PID를 검색하는 경우

```
[root@server1 /test]# ps
    PID TTY          TIME CMD
  12266 pts/0    00:00:00 bash

[root@server1 /test]# ps -ef | grep 12266
root       12266   12261  0 11:59 pts/0    00:00:00 bash
root       14238   12266  0 13:20 pts/0    00:00:00 ps -ef
root       14239   12266  0 13:20 pts/0    00:00:00 grep -i --color=auto 12266

[root@server1 /test]# ps -ef | grep bash
root         971       1  0 Feb26 ?        00:00:01 /bin/bash /usr/sbin/ksmtuned
root       12266   12261  0 11:59 pts/0    00:00:00 bash
root       14273   12266  0 13:21 pts/0    00:00:00 grep -i --color=auto bash
```

③ 패키지 설치 유무 확인

```
[root@server1 /test]# rpm -qa | grep sendmail
sendmail-8.15.2-32.el8.x86_64
```

```
# rpm -q sendmail
# rpm -qa sendmail
# rpm -qa | grep sendmail
```

[EX3] 파일내의 특정 패턴 여러개 검색하기

fgrep/egrep/pcre(perl compatable regular expression)
* grep -F : -F, --fixed-strings(fgrep CMD)
* grep -E : -E, --extended-regexp(egrep CMD)
* grep -P : -P, --perl-regexp(PCRE)

■ egrep(Extended grep) CMD == grep -E
```
# cat /var/log/messages | egrep -i 'warn|error|fail|crit|alert|emerg'
```
■ fgrep(Fixed grep) CMD == grep -F
```
# fgrep '^root' file1
```

① egrep 명령어를 사용하여 /etc/passwd 파일의 fedora 또는 user01 사용자 정보 확인
```
[root@server1 /test]# egrep "fedora|user01" /etc/passwd
fedora:x:1000:1000:fedora:/home/fedora:/bin/bash
user01:x:1001:1001::/home/user01:/bin/bash
```

[실무예] 로그 파일에서 에러 메세지 검색
```
# cat /var/log/messages | egrep -i 'warn|error|fail|danger|crit|alert|emerg'
# alias chklog='cat "$1" | egrep -i "warn|error|fail|danger|crit|alert|emerg"'
# chklog /var/log/messages
# chklog /var/log/messages | egrep 'Dec 28'

# mkdir -p /root/bin && cd /root/bin
# vi chklog.sh
```
```
#!/bin/bash

export LANG=en_US.UTF-8


if [ $# -ne 1 ] ; then
    echo "Usage: $0 <Log filename>"
    exit 1
fi
LOGFILE="$1"

if [ ! -f "$LOGFILE" ]; then
    echo "Error: file not found: $LOGFILE"
    exit 2
fi

MONTH=$(date +'%b')
DAY=$(date +'%e')
grep -E "$MONTH $DAY" "$LOGFILE" | grep -E -n -i --color=auto 'warn|error|fail|crit|alert|emerg'
```
```
# chmod 700 chklog.sh
# chklog.sh /var/log/messages
# chklog.sh /var/log/secure
```

find 명령어

• find 명령어

```
# find / -name core -type f
# find / -user user01 -group class1
# find / -mtime [-7|7|+7]
# find / -perm [755|-755]
# find / -size [-300M|300M|+300M]
# find / -name core -type f -exec rm -f {} \;

# find / -name core -type -ls
# find / -name core -type f -ok rm {} \;

# find /Log_Dir -name "*.log" -type f \
-mtime +30 -exec rm -f {} \;
# find /Log_Dir -mtime -2 -size +1G -type f
```

http://cafe.daum.net/bscsolaris

2 find CMD

```
NAME
       find - search for files in a directory hierarchy

SYNOPSIS
       find [-H] [-L] [-P] [path...] [expression]

DESCRIPTION
       This manual page documents the GNU version of find.  GNU
       find searches the directory tree rooted  at  each  given
       file  name  by evaluating the given expression from left
       to right, according to the rules of precedence (see sec-
       tion  OPERATORS),  until  the outcome is known (the left
       hand side is false for and operations, true for or),  at
       which point find moves on to the next file name.

       If  you  are using find in an environment where security
       is important (for example if you are using it  to  seach
       directories  that  are  writable  by  other  users), you
       should read the "Security Considerations" chapter of the
       findutils  documentation,  which is called Finding Files
       and comes with findutils.  That document also  includes
       a  lot more detail and discussion than this manual page,
       so you may find it a more useful source of  information.

OPTIONS
       -inum n
              File has inode number n.  It is  normally  easier
              to use the -samefile test instead.

       -mtime n
              File's data was last  modified  n*24  hours  ago.
              See  the  comments  for  -atime to understand how
              rounding affects the interpretation of file modi-
              fication times.

       -name pattern
              Base  of  file  name  (the  path with the leading
              directories removed) matches shell  pattern  pat-
              tern.   The  metacharacters  ( `*' ,  `?' , and  `[]' )
```

```
              match a '.' at the start of the base name (this
              is a change in findutils-4.2.2; see section STAN-
              DARDS CONFORMANCE below). To ignore a directory
              and the files under it, use -prune; see an exam-
              ple in the description of -wholename. Braces are
              not recognised as being special, despite the fact
              that some shells including Bash imbue braces with
              a special meaning in shell patterns. The file-
              name matching is performed with the use of the
              fnmatch(3) library function. Don't forget to
              enclose the pattern in quotes in order to protect
              it from expansion by the shell.

       -newer file
              File was modified more recently than file. If
              file is a symbolic link and the -H option or the
              -L option is in effect, the modification time of
              the file it points to is always used.

       -nouser
              No user corresponds to file's numeric user ID.

       -perm mode
              File's permission bits are exactly mode (octal or
              symbolic). Since an exact match is required, if
              you want to use this form for symbolic modes, you
              may have to specify a rather complex mode string.
              For example '-perm g=w' will only match files
              which have mode 0020 (that is, ones for which
              group write permission is the only permission
              set). It is more likely that you will want to
              use the '/' or '-' forms, for example '-perm
              -g=w', which matches any file with group write
              permission. See the EXAMPLES section for some
              illustrative examples.

       -perm -mode
              All of the permission bits mode are set for the
              file. Symbolic modes are accepted in this form,
              and this is usually the way in which would want
              to use them. You must specify 'u', 'g' or 'o' if
              you use a symbolic mode. See the EXAMPLES sec-
              tion for some illustrative examples.

       -size n[cwbkMG]
              File uses n units of space. The following suf-
              fixes can be used:

               'b'     for 512-byte blocks (this is the default
                     if no suffix is used)

               'c'     for bytes

               'w'     for two-byte words

               'k'     for Kilobytes (units of 1024 bytes)

               'M'     for Megabytes (units of 1048576 bytes)

               'G'     for Gigabytes (units of 1073741824 bytes)

              The size does not count indirect blocks, but it
              does count blocks in sparse files that are not
              actually allocated. Bear in mind that the '%k'
              and '%b' format specifiers of -printf handle
              sparse files differently. The 'b' suffix always
              denotes 512-byte blocks and never 1 Kilobyte
              blocks, which is different to the behaviour of
              -ls.

       -type c
              File is of type c:

              b      block (buffered) special

              c      character (unbuffered) special

              d      directory
```

```
              p      named pipe (FIFO)

              f      regular file

              l      symbolic link; this is never true if the
                     -L option or the -follow option is in
                     effect, unless the symbolic link is bro-
                     ken.  If you want to search for symbolic
                     links when -L is in effect, use -xtype.

              s      socket

              D      door (Solaris)

       -user uname
              File is owned by user uname (numeric user ID
              allowed).

       -exec command ;
              Execute command; true if 0 status is returned.
              All following arguments to find are taken to be
              arguments to the command until an argument con-
              sisting of ';' is encountered. The string '{}'
              is replaced by the current file name being pro-
              cessed everywhere it occurs in the arguments to
              the command, not just in arguments where it is
              alone, as in some versions of find. Both of
              these constructions might need to be escaped
              (with a 'W') or quoted to protect them from
              expansion by the shell. See the EXAMPLES section
              for examples of the use of the '-exec' option.
              The specified command is run once for each
              matched file. The command is executed in the
              starting directory. There are unavoidable secu-
              rity problems surrounding use of the -exec
              option; you should use the -execdir option
              instead.

       -ok command ;
              Like -exec but ask the user first (on the stan-
              dard input); if the response does not start with
               'y' or 'Y', do not run the command, and return
              false. If the command is run, its standard input
              is redirected from /dev/null.

       -ls    True; list current file in 'ls -dils' format on
              standard output. The block counts are of 1K
              blocks, unless the environment variable
              POSIXLY_CORRECT is set, in which case 512-byte
              blocks are used. See the UNUSUAL FILENAMES sec-
              tion for information about how unusual characters
              in filenames are handled.
```

디렉토리안에서 원하는 파일을 찾고자 할 때 사용하는 명령어이다. find 명령 다음에 시작 디렉토리를 정해 주고 찾고자하는 파일 이름 앞에 옵션을 주면 된다.

[명령어 형식]
find [검색시작위치] [옵션1] [인자값1] [옵션2] [인자값2] ...

[명령어 옵션]

옵 션	설 명
-name	파일 이름을 기준으로 검색
-perm	파일 권한을 기준으로 검색한다.
-type	파일의 종류를 기준으로 검색 • b : 블록 파일 • c : 문자 • d : 디렉토리 • f : 파일 • l : 링크 • s : 소켓
-size	파일의 크기를 기준으로 검색 • +n : n보다 크다 • -n : n보다 작다 • n : n이다 • b : 512-byte • c : Bytes (Character = Byte) • k : Kilo Byte • M : Mega Bytes • G : Giga Bytes • w : 2-byte
-user	사용자 ID를 기준으로 검색
-mtime	특정 기간 이상 수정되지 않은 파일을 기준으로 검색
-inode	number 지정된 inode 번호와 파일을 찾는다.
-print	표준출력으로 검색된 파일 출력: GNU는 디폴트(default), Unix는 필수 입력
-exec command {} W;	찾은 각 파일에 대해 지정된 명령을 실행
-ok command {} W;	실행여부(실행 되어 있는지 아닌지)를 사용자에게 확인 후 명령을 실행

■ 자주 사용되는 find 명령어 형식들

```
(형식1) # find / -name file1 -type [f|d]
(형식2) # find / -user user01 -group class1
(형식3) # find / -mtime {-7|7|+7}
(형식4) # find / -perm {-755|755}
(형식5) # find / -size {-300M|300M|+300M}
(형식6) # find / -name core -type f -exec CMD {} W;       # find + CMD
        # find / -name core -type f -exec rm -f {} W;
```

[실습] find 명령어 형식 실습

■ 사용 시스템
- server1

■ 실습 시나리오
- 파일 이름 검색 (예: # find / -name core -type f)
- 소유자/그룹 검색 (예: # find / -user fedora -group fedora)
- 수정 시간 검색 (예: # find / -mtime -7|7|+7)
- 크기 검색 (예: # find / -size -300M|300M|+300M)
- 퍼미션 검색 (예: # find / -perm -755|755)
- 다른 명령어와 연결 (예: # find / -name core -type f -exec CMD {} W;)

[EX1] 파일 이름 검색(예: **# find / -name file1 -type f**)

① 실습용 구조 만들기
다음과 같은 구조를 만들어 본다.
- 디렉토리: **mkdir CMD 사용**
- 파일: **touch CMD 사용**

```
/test/
|-- dir1/
|   `-- file1
|-- dir2/
|   `-- file3
|-- file1
`-- file2
```

② 파일의 이름이 file1 인 파일들 검색하기
```
[root@server1 ~]# cd /test
[root@server1 /test]# find . -name file1
./dir1/file1
./file1
```

③ 파일의 이름이 file1 이면서, 일반 파일인 파일들 검색하기
* -type f : 일반이면
* -type d : 디렉토리 파일이면

```
[root@server1 /test]# find . -name file1 -type f
./dir1/file1
./file1
```

[참고] 여러 이름 검색하기
```
# find . -name file1 -o -name dir1      /* -a : AND, -o : OR */
# find . -name "*1"
```

④ /usr/share 디렉토리안에 '*.xml' 파일들 검색하기
```
[root@server1 /test]# find /usr/share -name '*.xml' -type f
..... (중략) .....
/usr/share/gedit/plugins/snippets/ruby.xml
/usr/share/gedit/plugins/snippets/sh.xml
/usr/share/gedit/plugins/snippets/snippets.xml
/usr/share/gedit/plugins/snippets/tcl.xml
/usr/share/gedit/plugins/snippets/xml.xml
/usr/share/gedit/plugins/snippets/xslt.xml
```

```
# find / -name '*.py' -type f
# find / -name '*.jar' -type f
```

[EX2] 사용자/그룹 검색(예: # find / -user user01 -group class1)

① /home 디렉토리 안에 fedora 사용자 소유이면서 그룹이 fedora 그룹으로 되어 있는 파일 검색하기

```
[root@server1 /test]# find /home -user fedora -group fedora
..... (중략) .....
/home/fedora/.cache
/home/fedora/.ssh
/home/fedora/.ssh/known_hosts
/home/fedora/dirtest
/home/fedora/dirtest/test2.txt
/home/fedora/.viminfo
```

```
[root@server1 /test]# find /home -user fedora -group fedora -ls
..... (중략) .....
 16777365      4 -rw-r--r--   1  fedora   fedora        175 Feb 19 09:50 /home/fedora/.ssh/known_hosts
      142      0 drwxrwxr-x   2  fedora   fedora         23 Feb 21 13:36 /home/fedora/dirtest
      143      0 -rw-rw-r--   1  fedora   fedora          0 Feb 21 13:36 /home/fedora/dirtest/test2.txt
      144      4 -rw-------   1  fedora   fedora        948 Feb 21 15:45 /home/fedora/.viminfo
```

[EX3] 날짜 검색(예: # find / -mtime -7|7|+7)

```
[root@server1 /test]# cd dir2
[root@server1 /test]# touch file1 file2 file3 file4 file5 file6
[root@server1 /test]# date
Mon Feb 22 13:04:22 KST 2021
```

```
[root@server1 /test# touch -t 02211304 file2
[root@server1 /test# touch -t 02201304 file3
[root@server1 /test# touch -t 02191304 file4
[root@server1 /test# touch -t 02181304 file5
[root@server1 /test# touch -t 02171304 file6
```

```
# ----- 다음을 복사해서 사용할 수도 있다. -----
mkdir -p /test/dir2 && cd /test/dir2
for i in {0..5}
do
    N=$(expr $i + 1)
    TIME=$(date -d "-${i} days" +'%m%d%H%M')
    FILE=file${N}
   touch -t $TIME $FILE
done
```

```
[root@server1 /test# ls -l file*
-rw-r--r-- 1 root root 0 Feb 22 13:04 file1
-rw-r--r-- 1 root root 0 Feb 21 13:04 file2
-rw-r--r-- 1 root root 0 Feb 20 13:04 file3
-rw-r--r-- 1 root root 0 Feb 19 13:04 file4
-rw-r--r-- 1 root root 0 Feb 18 13:04 file5
-rw-r--r-- 1 root root 0 Feb 17 13:04 file6
```

```
file1  file2  file3  file4  file5  file6
(오늘) (어제) (그제)
```

```
[root@server1 /test# find . -mtime 3 -type f     /* 수정 날짜가 3일전인 파일 */
./file4
```

```
[root@server1 /test# find . -mtime -3 -type f    /* 수정한 날짜가 3일이 안된 파일 */
./file3
./file2
./file1
```

```
[root@server1 /test# find . -mtime +3 -type f    /* 수정한 날짜가 3일이 지난 파일 */
./file6
./file5
```

[EX4] 퍼미션 검색(예: # find / -perm 755 -type f)

① 실습 준비

```
[root@server1 /test/dir2]# cd /test
[root@server1 /test]# mkdir dir3
[root@server1 /test]# cd dir3
[root@server1 /test/dir3]# touch file{1..8}

[root@server1 /test/dir3]# chmod 000 file1  (---)
[root@server1 /test/dir3]# chmod 100 file2  (--x)
[root@server1 /test/dir3]# chmod 200 file3  (-w-)
[root@server1 /test/dir3]# chmod 300 file4  (-wx)
[root@server1 /test/dir3]# chmod 400 file5  (r--)
[root@server1 /test/dir3]# chmod 500 file6  (r-x)
[root@server1 /test/dir3]# chmod 600 file7  (rw-)
[root@server1 /test/dir3]# chmod 700 file8  (rwx)
```

```
# ----- 다음을 복사해서 사용할 수도 있다. -----
mkdir -p /test/dir3 && cd /test/dir3
for i in {0..7}
do
    N=$(expr $i + 1)
    touch file${N}
    chmod ${i}00 file${N}
done
```

```
[root@server1 /test/dir3]# ls -l file*
----------  1 root root 0 Feb 22 13:14 file1
---x------  1 root root 0 Feb 22 13:14 file2*
--w-------  1 root root 0 Feb 22 13:14 file3
--wx------  1 root root 0 Feb 22 13:14 file4*
-r--------  1 root root 0 Feb 22 13:14 file5
-r-x------  1 root root 0 Feb 22 13:14 file6*
-rw-------  1 root root 0 Feb 22 13:14 file7
-rwx------  1 root root 0 Feb 22 13:14 file8*
```

② 600 퍼미션인 파일 검색하기

```
[root@server1 /test/dir3]# find . -perm 600 -type f -ls
 69663171      0 -rw-------   1  root     root            0 Feb 22 13:14 ./file7
```

③ 최소한 400 퍼미션인 파일 검색하기

```
[root@server1 /test/dir3]# find . -perm -400 -type f -ls
 69663169      0 -r--------   1  root     root            0 Feb 22 13:14 ./file5
 69663170      0 -r-x------   1  root     root            0 Feb 22 13:14 ./file6
 69663171      0 -rw-------   1  root     root            0 Feb 22 13:14 ./file7
 69663172      0 -rwx------   1  root     root            0 Feb 22 13:14 ./file8
```

[참고] find / ₩(-perm -4000 -o -2000 ₩) -type f

(질문) -perm -4000 versus -perm -2000

	-perm -4000	-perm -2000
(최소)	--s------	-----s---
	~	~
(최대)	rwsrwxrwx	rwxrwsrwx

[EX5] 파일 크기 검색(예: # find / -size 50k -type f)

```
# man find
'b'    for  512-byte  blocks  (this is the default if no suffix is used)
'c'    for bytes
'w'    for two-byte words
'k'    for Kilobytes (units of 1024 bytes)
'M'    for Megabytes (units of 1048576 bytes)
'G'    for Gigabytes (units of 1073741824 bytes)
```

① 실습 준비

```
[root@server1 /test/dir3]# cp /etc/services file10
[root@server1 /test/dir3]# cp /boot/vmlinuz-5.* file11
[root@server1 /test/dir3]# /bin/ls -l
```

```
---------- 1 root root        0 Aug 12 15:13 file1
-rw-r--r-- 1 root root   692252 Aug 12 15:17 file10
-rwxr-xr-x 1 root root 14424104 Aug 12 15:18 file11
---x------ 1 root root        0 Aug 12 15:13 file2
--w------- 1 root root        0 Aug 12 15:13 file3
--wx------ 1 root root        0 Aug 12 15:13 file4
-r-------- 1 root root        0 Aug 12 15:13 file5
-r-x------ 1 root root        0 Aug 12 15:13 file6
-rw------- 1 root root        0 Aug 12 15:13 file7
-rwx------ 1 root root        0 Aug 12 15:13 file8
```

* file1-8 0 Bytes
* file10 692252 Bytes
* file11 14424104 Bytes

② 2942 bytes 크기의 파일 검색

```
# find . -size 692252c -type f    /* 정확이 일치되는 용량만 검색 */
```

```
./file10
```

* c : bytes
* k : KB
* M : MB
* G : GB

③ 2942 bytes 미만 크기의 파일 검색

```
# find . -size -692252c -type f   /* 일치되는 것 제외한 파일 크기가 2942bytes 미만 */
```

```
./file1
./file2
./file3
./file4
./file5
./file6
./file7
./file8
```

④ 2942 bytes 초과되는 크기의 파일 검색

```
# find . -size +692252c -type f   /* 파일 크기가 2942bytes 보다 큰것 */
```

```
./file11
```

[EX6] 디렉토리안에 특정한 패턴을 가진 파일들을 삭제(# find / -name file -type f -exec CMD {} W;)

=> find + CMD <=

```
[root@server1 /test/dir3]# cd /test
[root@server1 /test]# find . -name file1 -type f
./file1
./dir2/file1
```

다음과 같은 명령어의 형식을 이해하고 수행 해 보자.

```
# find . -name file1 -type f -ls
# find . -name file1 -type f -exec chown user01 {} W;
# find . -name file1 -type f -ls

# find . -name file1 -type f -exec chmod 640 {} W;
# find . -name file1 -type f -ls

# find . -name file1 -type f -exec rm -f {} W;
# find . -name file1 -type f

# find /usr/share -name '*.py' -type f -exec cp {} /tmp W;
# find /tmp -name '*.py'
```

[실무예] 오래된 로그 기록 삭제하는 예

* OS 로그 파일 -> logrotate CMD
* 제품 로그 파일 ->
* SI 업체 -> 웹(?)

오래된 로그들은 일정한 기간이 지나면(ex: 시스템 생성일 ~ 30일) 그 의미를 상실하게 될수 있다. 따라서, 일정 시간이 지난 로그파일의 경우 find라는 명령어를 이용하여 파일을 주기적으로 삭제해 주도록 할 수 있다.

로그 파일 형식: /was/logs/server_0125.log
/was/logs/server_0126.log
/was/logs/server_0127.log
....

```
# find /Log_Dir1 -name "*.log" -type f -mtime +30 -exec rm -f {} W;
# find /Log_dir2 -name "*.log" -type f -mtime +60 -exec rm -f {} W;
```

-> find CMD + crontab CMD

```
# crontab -e
분  시  일  월  요일  CMD
0   4   1   *   *    find /zeuslogs -name '*.log' -type f -mtime +60 -exec rm -f {} W;
5   4   1   *   *    find /weblogs -name '*.log' -type f -mtime +90 -exec rm -f {} W;
```

[실무예] 파일시스템이 갑자기 풀(Full) 나는 경우 예
내가 관리하는 서버가 어제는 이상없는 경우, 오늘 IDC(통합 전산 센터) 출근 중에 데이터센터 연락이 왔다. 담당 서버의 특정 폴더가 풀(full) 났다고 하고 처리해 달라고 한다.
=> df + du + find + lsof CMD

```
# find /var -mtime -2 -size +1G -type f
# find /var -mtime -2 -size +512M -type f
/var/server/log/file.log
# > /var/server/log/file.log
```

[참고] lsof(list open file)

```
# lsof | grep /var/server/log/file.log
```

[참고] df + du + find + lsof CMD

```
# df -h
# du -sh /var
# cd /var ; du -sh * | sort -hr | more
# find /var -type f -size +1G -mtime -2
# lsof | grep FILE    (# lsof FILE)
```

[실무예] (소스 코드가 있는 경우) 에러메세지가 들어 있는 startup script 검색 예
```
# /was/bin/startup.sh
..... Server Error .....
# find /was -type f -exec grep -l 'Server Error' {} W;   (# grep -r 'Server Error' /was)
/was/conf/server.sh
# vi /was/conf/server.sh
/Server Error
......
if 조건 ; then
    정상 동작
else
    echo "Server Error"
fi
```

[실무예] (소스 코드가 없는 경우) 에러 메세지를 검색하는 방법에 대해서
```
http://www.google.co.kr
(ㄱ) [site:] 지시자 사용
    site:.redhat.com "Server Error"
(ㄴ) [AND] 사용
    "Server Error1" AND "Server Error2" (AND/OR)
    예) "Server Error" AND "Windows" AND "Oracle"
    예) "Error1" AND "Error2" AND "Error3"
(ㄷ) [큰 따옴표] 사용
    "Server Error1"
+
(ㄹ) 정보 검색
    가상화 .pdf
    가상화 .ppt

ChaptGPT
* https://chat.openai.com/

    (코드예제)
    다음 코드의 기능을 개선해줘:
    코드 블록

* (IDE) cursor, codieum
```

[참고] locate CMD

locate CMD
- 파일이름이나 경로를 기준으로 파일 검색
- mlocate 데이터베이스에서 정보를 조회하기 때문에 locate 명령이 빠르게 실행된다.
- 그러나 데이터베이스가 실시간으로 업데이트되지 않으므로 정확한 결과를 얻으려면 자주 업데이트를 한다.
  (관리자) # updatedb
- locate 데이터베이스는 매일 자동으로 업데이트된다.

```
$ sudo updatedb
$

$ locate passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd
/usr/bin/grub2-mkpasswd-pbkdf2
/usr/bin/passwd
/usr/lib/firewalld/services/kpasswd.xml
/usr/lib64/samba/pdb/smbpasswd.so
/usr/lib64/security/pam_unix_passwd.so
/usr/sbin/chgpasswd
/usr/sbin/chpasswd
.... (중략) .....

$ locate -i messages | head
/usr/lib/locale/C.utf8/LC_MESSAGES
/usr/lib/locale/C.utf8/LC_MESSAGES/SYS_LC_MESSAGES
/usr/lib/python3.12/site-packages/orca/messages.py
/usr/lib/python3.12/site-packages/orca/__pycache__/messages.cpython-312.opt-1.pyc
/usr/lib/python3.12/site-packages/orca/__pycache__/messages.cpython-312.pyc
/usr/lib64/samba/libmessages-dgm-private-samba.so
/usr/lib64/samba/libmessages-util-private-samba.so
/usr/share/locale/aa/LC_MESSAGES
/usr/share/locale/ab/LC_MESSAGES
/usr/share/locale/ab/LC_MESSAGES/at-spi2-core.mo

$ locate -n 5 passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd
```

3 sed CMD

```
NAME
       sed - stream editor for filtering and transforming text

SYNOPSIS
       sed [OPTION]... {script-only-if-no-other-script} [input-file]...

DESCRIPTION
       Sed  is a stream editor.  A stream editor is used to perform basic text trans-
       formations on an input stream (a file or input from  a  pipeline).   While  in
       some  ways similar to an editor which permits scripted edits (such as ed), sed
       works by making only one pass over the input(s), and is consequently more  ef-
       ficient.   But it is sed's ability to filter text in a pipeline which particu-
       larly distinguishes it from other types of editors.

       -n, --quiet, --silent
              suppress automatic printing of pattern space

       -e script, --expression=script
              add the script to the commands to be executed

       -f script-file, --file=script-file
              add the contents of script-file to the commands to be executed
```

sed(Stream Editor) 명령은 파일을 편집할 수 있는 기능을 제공한다. 편집기(vi/vim) 처럼 원하는 문자열을 치환하는 등 다양한 작업을 수행할 수 있다. 어떤 파일에서 원하는 내용을 추출하거나, 치환하는 작업을 수행할 수 있다.

[명령어 형식]

```
# sed [-n] 'addressCMD' file1
# sed [-n] '/pattern/CMD' file1
```

[명령어 옵션]

옵션	설 명
-e	조건식 스크립트를 직접 지정
-f	조건식 스크립트가 기재된 파일을 지정
-n	패턴이 일치하는 라인만 출력
-i	수정한 내용으로 파일 덮어쓰기

[하위 명령어 종류]

하위명령	설 명
a W	현재 행에 하나 이상의 새로운 행 추가
i W	현재 행의 내용을 새로운 내용으로 교체
d	행 삭제
p	행 출력
s	문자열 치환

[참고] 참고 사이트
https://velog.io/@inhwa1025/Linux-SED-명령어-사용법

■ sed 하위 명령 사용 예제
- p CMD(print)
- d CMD(delete)
- s CMD(substraction, substitution)
- a CMD(append)
- i CMD(insert)

(p CMD) 사용하는 방법 - print

```
1~3라인 출력
# sed -n '1,3p' /etc/hosts              (# head -n 3 /etc/hosts)
root 문자열 라인만 출력
# sed -n '/root/p' /etc/passwd          (# grep 'root' /etc/passwd)
첫번째 라인부터 adm 문자열 라인까지 출력
# sed -n '1,/^adm/p' /etc/passwd
라인 처음 root 문자열 부터 라인 처음 adm 라인까지 출력
# sed -n '/^root/,/^adm/p' /etc/passwd
10번째라인부터 마지막 라인까지 출력
# sed -n '10,$p' /etc/passwd            (# tail -n +10 /etc/passwd)
3번째 라인 출력
# sed -n '3p' /etc/passwd               (# sed -n '3,3p' /etc/passwd)
```

(d CMD) 사용하는 방법 - delete

```
1~3라인 삭제
# sed '1,3d' /etc/passwd
3번째 라인 삭제
# sed '3d' /etc/passwd                  (# sed '3,3d' /etc/passwd)
3번째 라인부터 마지막 라인까지 삭제
# sed '3,$d' /etc/passwd
마지막 라인 삭제
# sed '$d' /etc/passwd
root 문자열라인 삭제
# sed '/root/d' /etc/passwd             (# grep -v root /etc/passwd)

설정파일의 빈공백/주석라인을 제거
# mkdir -p /test && cd /test && rm -rf /test/*
# cp /etc/httpd/conf/httpd.conf httpd.conf
# sed -i -e '/^$/d' -e '/^#/d' -e '/.*#.*/d' httpd.conf
or
# cat httpd.conf | egrep -v '^$|^#|.*#.*' > httpd.conf.OLD
# mv -f httpd.conf.OLD httpd.conf

라인 처음이 127.0.0.1 시작하는 라인부터 172.16.6.2XX 시작하는 라인까지 삭제
# sed '/^127.0.0.1/,/^172.16.6.2XX/d' /etc/hosts
```

(s CMD) 사용하는 방법 - substitution

- vim (:1,3s/root/ROOT/g)

```
문자열 root를 ROOT로 치환
# sed 's/root/ROOT/' /etc/group         (# sed '1,$s/root/ROOT/' /etc/group)
1~3라인의 root를 ROOT로 치환
# sed '1,3s/root/ROOT/g' /etc/group

라인 앞 4글자 삭제하기
# sed 's/^....//' /etc/hosts
라인 끝 4글자 삭제하기
# sed 's/....$//' /etc/hosts

1~3라인의 처음 4칸 들여쓰기
# sed '1,3s/^/    /' /etc/hosts
```

```
1~3라인의 처음 4칸을 삭제하기
# sed '1,3s/^    //' /etc/hosts

구분자를 다른 문자로 변경하기(/ -> # ; ,)
# sed 's#/test/file.sh#/test/file.c#g' file1
# sed 's;/test/file.sh;/test/file.c;g' file1
# sed 's,/test/file.sh,/test/file.c,g' file1

지정된 파일의 root 단어를 ROOT로 치환(원본 파일 수정하는 경우)
# cp /etc/group group
# sed 's/root/ROOT/' group > group.tmp
# /bin/mv group.tmp group                    (# cat group.tmp > group)
or
# sed -i 's/root/ROOT/' group

linux200 단어를 가진 라인의 192.168.20.200을 172.16.20.200으로 변경
# cd /test
# cat << EOF > hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.10.10  main.example.com     main
192.168.10.20  server1.example.com  server1
192.168.10.30  server2.example.com  server2
EOF
# sed '/server1/s/192.168.10.20/192.168.10.21/' hosts

탭을 스페이스로 변환
# sed -e 's/<tab>/<space>/g' test.txt
```

(a CMD) 사용하는 방법 - append

```
라인 뒤에 추가하기
# cat << EOF > file1
1111
2222
3333
EOF
# sed -i '$a LastLine' file1    # 마지막 라인 뒤에 LastLine 라인 추가
# sed -i '3a 3rdLine' file1     # 3번째 라인 뒤에 3rdLine 라인 추가
```

(i CMD) 사용하는 방법 - insert

```
라인 앞에 추가하기
# sed -i '1i FirstLine' file1   # 첫번째 라인 앞에 FirstLine 추가
# sed -i '2i 2ndLine' file1     # 2번째 라인 앞에 2ndLine 추가
```

[실무예] SELinux 변경(enforcing -> permissive)

```
sed -i 's/^SELINUX=enforcing/SELINUX=permissive/' /etc/selinux/config
setenforce 0
```

[실무예] 환경 파일 ~/.bashrc 파일의 마지막 라인에 내용 추가

```
sed -i '$a alias c="clear"' ~/.bashrc
source ~/.bashrc
```

4 awk CMD

```
NAME
       gawk - pattern scanning and processing language

SYNOPSIS
       gawk [ POSIX or GNU style options ] -f program-file [ -- ] file ...
       gawk [ POSIX or GNU style options ] [ -- ] program-text file ...

DESCRIPTION
       Gawk  is  the  GNU Project's implementation of the AWK programming language.  It conforms to
       the definition of the language in the POSIX 1003.1 standard.  This version in turn is  based
       on the description in The AWK Programming Language, by Aho, Kernighan, and Weinberger.  Gawk
       provides the additional features found in the current version of Brian Kernighan's  awk  and
       numerous GNU-specific extensions.

       The  command  line consists of options to gawk itself, the AWK program text (if not supplied
       via the -f or --include options), and values to be made available in the ARGC and ARGV  pre-
       defined AWK variables.

       When  gawk  is  invoked  with the --profile option, it starts gathering profiling statistics
       from the execution of the program.  Gawk runs more slowly in this  mode,  and  automatically
       produces  an execution profile in the file awkprof.out when done.  See the --profile option,
       below.

       Gawk also has an integrated debugger. An interactive debugging session  can  be  started  by
       supplying  the --debug option to the command line. In this mode of execution, gawk loads the
       AWK source code and then prompts for debugging commands.  Gawk can only  debug  AWK  program
       source  provided with the -f and --include options.  The debugger is documented in GAWK: Ef-
       fective AWK Programming.
```

awk는 기능을 디자인한 사람들의 이니셜을 조합하여 만든 이름이다. Aho + Weinberger + Kernighan(A:Alfred V. Aho, W:Peter J. Weinberger, K:Brian W. Kernighan). awk는 파일로부터 레코드(record)를 선택하고, 선택된 레코드에 포함된 값을 조작하거나 데이터화하는 것을 목적으로 사용하는 프로그램이다. 즉, awk 명령의 입력으로 지정된 파일로부터 데이터를 분류한 다음, 분류된 텍스트 데이터를 바탕으로 패턴 매칭 여부를 검사하거나 데이터 조작 및 연산 등의 액션을 수행하고, 그 결과를 출력하는 기능을 수행한다.

[명령어 형식]
```
# awk 'statement' file1
# awk '{action}' file1
# awk 'statement {action}' file1
```

[명령어 옵션]

옵션	설 명
-F	필드 구분 문자 지정
-f	aws program 파일 경로 지정
-v	awk program에서 사용할 variable 값 지정

■ awk 명령 사용 예

```
-rw-r--r--  1  root  root  28  Jul  25  22:32  file1   => $0
   $1       $2  $3    $4   $5  $6   $7   $8     $9
```

```
awk '{Action}' file 형식
```

```
# awk '{ print $0 }' testfile
# awk '{ print $3 $5 $9 }' testfile
# awk '{ print $3 "     " $9 "     " $6, $7 }' testfile
# awk '{ print $3 "Wt" $5 "Wt" $9 }' testfile
# awk '{ print $9, "is using", $5, "bytes" }' testfile
```

```
[형식]
awk 'statement' file1
awk '{Action}' file1
awk 'statement {Action}' file1
```

```
# awk '/file/' testfile                    (# grep 'file' testfile)
# awk '{ print $1, $2, $3 }' testfile
# awk '/file/ { print $1, $2 }' testfile
```

[실무예] 추가된 일반 사용자 목록(1000 <= UID <= 60000)

```
awk -F: '$3 >=1000 && $3 <= 60000 {print $1}' /etc/passwd
```

[실무예] 파일시스템 사용량(%) 뽑아 보기

```
df -h / | tail -1 | awk '{print $6}' | awk -F% '{print $1}'
```

[실무예] ens160 NIC IP 정보 확인

```
ip addr show ens160 | grep 'inet ' | awk '{print $2}' | awk -F/ '{print $1}'
```

[실무예] 좀비 프로세스 확인

```
ps -elf | awk '$2 == "Z" {print $0}'
```