유틸리티

파일 관련 유틸리티

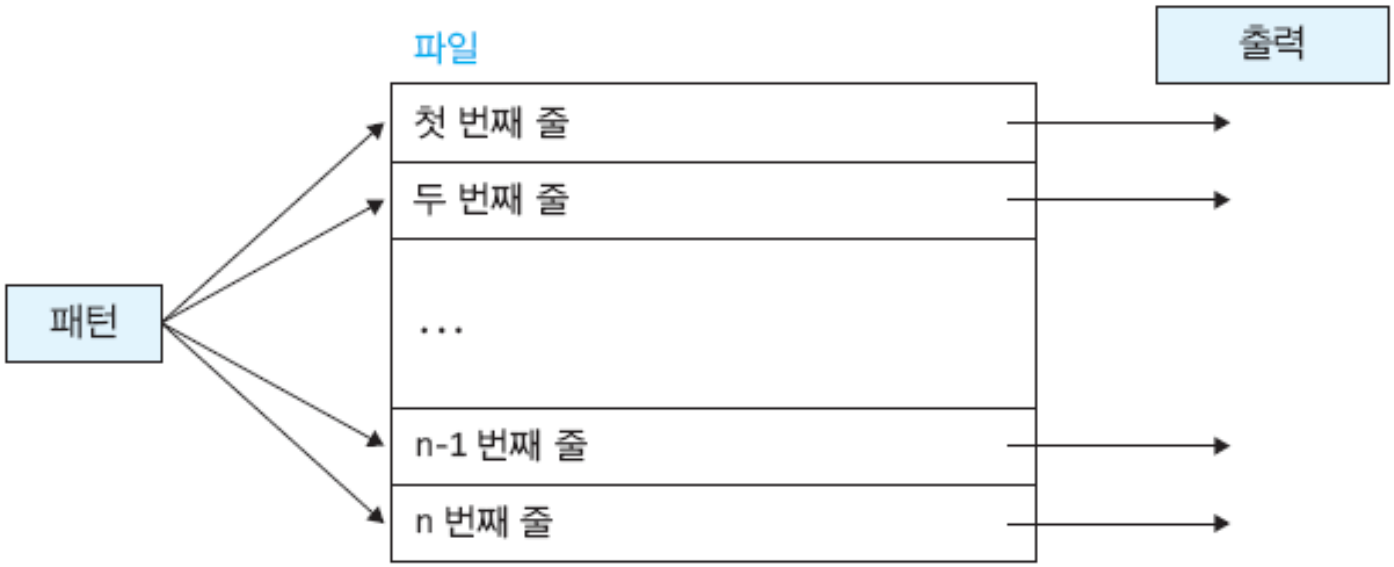
grep 명령어

- 사용법

0개 이상

```
$ grep 패턴 파일*
```

파일(들)을 대상으로 지정된 패턴의 문자열을 검색하고, 해당 문자열을 포함하는 줄들을 출력한다.



grep 명령어의 옵션

옵션	기능
-i	대소문자를 무시하고 검색한다.
-l	해당 패턴이 들어있는 파일 이름을 출력한다.
-n	각 줄의 줄번호도 함께 출력한다.
-v	명시된 패턴을 포함하지 않는 줄을 출력한다.
-c	패턴과 일치하는 줄 수를 출력한다.
-w	패턴이 하나의 단어로 된 것만 검색한다.

필터링: grep 명령어

- $ grep with you.txt

  Until you come and sit awhile with me

  There is no life - no life without its hunger;

  But when you come and I am filled with wonder,

- $grep -w with you.txt (-w 독립적인 단어 하나로 구분)

  Until you come and sit awhile with me

  But when you come and I am filled with wonder,

- $grep -n with you.txt (-n 줄 번호 표시)

  4:Until you come and sit awhile with me

  15:There is no life - no life without its hunger;

  17:But when you come and I am filled with wonder,

grep 명령어

- $grep -i when you.txt (-i 대소문자 무시)

```
When I am down and, oh my soul, so weary
When troubles come and my heart burdened be
I am strong, when I am on your shoulders
But when you come and I am filled with wonder,
```

- $grep -v raise you.txt (-v 해당 패턴을 포함하지 않는 줄 출력)

```
When I am down and, oh my soul, so weary
When troubles come and my heart burdened be
Then, I am still and wait here in the silence
Until you come and sit awhile with me
I am strong, when I am on your shoulders
There is no life - no life without its hunger;
Each restless heart beats so imperfectly;
But when you come and I am filled with wonder,
Sometimes, I think I glimpse eternity
```

- 고정된 문자열 대신 정규 표현식을 이용하여 패턴 명세 가능 (다음 페이지)

정규표현식(regular expression)

- 정규표현식을 이용한 패턴 기술
  - . : 임의의 한 문자
  - * : 0개 이상 예) a*b는 b, ab, aab, aaab, ...
  - [ ] : [과] 사이의 문자 중 하나 a가 0번 이상, b는 한 번
  - [^...] : [^ 과] 사이의 문자를 제외한 나머지 문자 중 하나
  - ^, $: 각각 줄의 시작과 끝을 의미한다.

  - ^ 의 의미 주의
    - 시작 vs. Not 구별

정규표현식 요약

문자	의미	예	결과
^	**라인의 시작**	'^문자열'	문자열로 시작하는 모든 행
$	**라인의 끝**	'문자열$'	문자열로 끝나는 모든 행
.	한 글자	'a...b'	한글자 대응, a로 시작해서 b로 끝나는 5글자 검색
?	없거나 한글자	'patter?'	patter 또는 patter과 한 문자 더 있는 문자열 검색, 즉 없거나 한글자 (pattern, pattera 등)
*	앞의 항목이 없거나 여러 번 반복	'ab*'	a다음에 b가 없거나 반복적으로 나타나는 라인 검색 a가 한 번 나타나고 b는 0번 이상 나타나야 함
[ ]	괄호안의 글자중 하나	'[Pp]atter n'	Pattern 또는 pattern이 나타나는 라인 검색
[^]	괄호 안에 있는 글자가 아닌 글자	'[^a-m]att' a~m	att앞에 a부터 m까지 나오지 않는 라인 검색

grep *.txt //정규식이나 워드단위

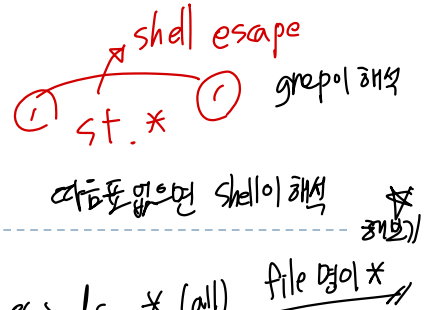
정규표현식(regular expression)

시험

- $ grep -w 'st.*' you.txt (st로 시작하는 모든 단어)

  Then, I am still and wait here in the silence → 아래 참고
  You raise me up, so I can stand on mountains
  You raise me up, to walk on stormy seas
  I am strong, when I am on your shoulders

  참고, 실제 결과는 단어 단위(-w 옵션)로 개행문자까지 확인하여 출력
  특정 단어 검색이 아니라 **패턴을 찾아서** 해당 문장을 출력하는 개념
  Then, I am still and wait here in the silence



grep -w 'st*' you.txt
s 한번 t는 0번 이상

grep 'st.*' you.txt
st가 단어의 중간에 있더라도 출력

최소 2글자

- $ grep -w 'w.*t' you.txt (w로 시작하여 t로 끝나는 모든 단어)

  Then, I am still and wait here in the silence
  There is no life - no life without its hunger;

정규식 사용 예

- `$ grep 'st.?' you.txt`

  두글자

  Then, I am still and wait here in the silence
  You raise me up, so I can stand on mountains
  You raise me up, to walk on stormy seas
  I am strong, when I am on your shoulders
  Each restless heart beats so imperfectly; ←단어중간

- `$ grep 'st.*e' you.txt`

  최소 3글자

  두줄일때는?

  기본적으로 한줄에서 본다. 개행문자를 끝으로 취급

  Then, I am still and wait here in the silence ←전체문장
  You raise me up, to walk on stormy seas
  I am strong, when I am on your shoulders
  Each restless heart beats so imperfectly; ←단어중간

- `$ grep -w 'st.*e' you.txt` ←독립단어 : -w

  Then, I am still and wait here in the silence

(참고) grep 명령어의 anchor 기호

기호	설명	예시	예시 결과
\b	단어 단위로 문자열에서 일치	\b(word)	단어 'word'를 포함하는 문장
\B	단어의 내부에서도 일치 여부 확인	\Bion	'condition'에서 'ion' (이를 포함하는 문장)
\<	단어 시작 부분의 문자열과 일치	\<cat	'catalog'에서 'cat'
\>	단어 끝 부분의 문자열과 일치	cat\>	'snowcat'에서 'cat'
\w	단어 구성 요소(알파벳)와 일치	\w+	'abc?'에서 'abc'
\W	단어 구성 요소가 아닌 것과 일치	\W	\w의 반대 'abc?' 에서 '?'
\s	공백 문자까지 확인 후 패턴 까지만 출력	hello\s	'hello()world'에서 'hello'
\S	공백이 아닌 문자와 일치	\S+	'()hello()'에서 'hello'

한 개의 단어 문자

한 개 이상의 연속된 단어 문자

문자열 내에서 공백 문자 하나만 찾음

hello 문자열 다음에 공백 문자가 있는지 확인

- anchor 기호는 Linux 유형이나 버전에 따라 작동하지 않을 수 있다.
- 자세한 설명은 GNU 공식 사이트를 참조

공부(알아보기)

파이프와 함께 grep 명령어 사용

- 파이프와 함께 grep 명령어 사용
  - 어떤 명령어를 실행하고 그 실행 결과 중에서 원하는 단어 혹은 문자열 패턴을 찾고자 할 때 사용함.

- 예

```
$ ls -l | grep chang
$ ps -ef | grep chang
```

정렬: sort 명령어

- sort 명령어

0개 이상 정렬 됨

```
$ sort [-옵션] 파일*
```

텍스트 파일(들)의 내용을 줄 단위로 정렬한다. 옵션에 따라 다양한 형태로 정렬한다.

  - 텍스트 파일을 정렬 필드를 기준으로 줄 단위로 오름차순 정렬 (알파벳 a to z)
  - -r 옵션을 사용하여 내림차순으로 정렬 (reverse)
  - 기본적으로 각 줄의 첫번째 필드(컬럼)이 정렬 필드
  - 정렬 필드는 0부터 시작
  - 정렬 필드 지정 가능
    - 형식: +시작필드 -종료필드 (e.g., +2 -3 : 세번째 필드로 정렬)

$ sort [-옵션] 파일*

정렬 예1

- $ sort you.txt //기본적으로 첫번째(0th) 필드 a to z로 정렬

```
But when you come and I am filled with wonder,
Each restless heart beats so imperfectly;
I am strong, when I am on your shoulders
Sometimes, I think I glimpse eternity
Then, I am still and wait here in the silence
There is no life - no life without its hunger;
Until you come and sit awhile with me
When I am down and, oh my soul, so weary
When troubles come and my heart burdened be
You raise me up, so I can stand on mountains
You raise me up, to more than I can be
You raise me up, to walk on stormy seas
```

sort 명령어 예

reverse

```
$ sort -r you.txt
You raise me up, to walk on stormy seas
You raise me up, to more than I can be
You raise me up, so I can stand on mountains
When troubles come and my heart burdened be
When I am down and, oh my soul, so weary
Until you come and sit awhile with me
There is no life - no life without its hunger;
Then, I am still and wait here in the silence
Sometimes, I think I glimpse eternity
I am strong, when I am on your shoulders
Each restless heart beats so imperfectly;
But when you come and I am filled with wonder,
```

정렬 필드 지정

시험문제

헷갈리지 않기

필드 지정	기능
-k 필드번호	필드번호에 해당하는 필드를 기준으로 정렬한다. 이 옵션에서 필드번호는 1부터 시작된다.
+시작필드 –종료필드	시작필드부터 종료필드-1까지의 필드들을 기준으로 정렬한다. 이 때 필드 번호는 0부터 시작된다.

정렬 예2

sort -k 3 you.txt

- $ sort +2 -3 you.txt (세번째 필드로 정렬)

Then, I am still and wait here in the silence
When I am down and, oh my soul, so weary
Until you come and sit awhile with me
When troubles come and my heart burdened be
Each restless heart beats so imperfectly;
You raise me up, so I can stand on mountains
You raise me up, to more than I can be
You raise me up, to walk on stormy seas
There is no life - no life without its hunger;
I am strong, when I am on your shoulders
Sometimes, I think I glimpse eternity
But when you come and I am filled with wonder,

sort 명령어의 옵션

옵션	기능
-b	앞에 붙는 공백은 무시한다.
-c	정렬이 되지 않은 상태로 출력한다. 파일이 정렬되어 있는지 확인 -> 아무것도 안뜸 정렬이 안 되어 있을 경우 오류 메시지를 출력
-d	숫자, 문자, 공백만 비교하여 사전식 순서로 정렬한다.
-f	대소문자를 구분하지 않고 정렬한다. a, A 이면 대문자가 먼저 출력
-n	숫자 문자열의 숫자 값에 따라 비교하여 정렬한다.
-r	역순(내림차순)으로 정렬한다.
-t 문자	지정한 문자를 필드 구분자로 사용한다. 기본은 스페이스로 나눔

```
예) sort -t "," -k 2 file.csv
예) sort -t "," -k 3 -n -r file.csv
```

sort 명령어의 옵션 예

- -o 출력파일 옵션
  - 정렬된 내용을 지정된 파일에 저장할 수 있다.

```
$ sort -o sort.txt you.txt
```

- -n 옵션
  - 숫자 문자열의 경우에 숫자가 나타내는 값의 크기에 따라 비교하여 정렬할 수 있다.
  - 예: "49"와 "100": 문자열로는 100이 우선 (1이 4보다 우선)
  - 숫자로는 49가 우선

필드 구분 문자 지정

```
$ 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
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
...
```

필드 구분자를 : 로 지정하고 (-t:) 세번재 필드로 정렬 (-k 3), 숫자 크기대로 (-n)

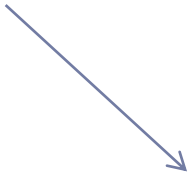
파일 자르기: split 명령어

- 하나의 파일을 일정한 크기의 여러 개 작은 파일로 분할

  $ split [-옵션] 입력파일 [출력파일]

  1000줄씩 분할하여 xaa, xab, ... 형태의 파일명으로 저장

  -l n 옵션: 분할할 줄 수(n)를 지정



- 예

```
$ split -l 10 you.txt        (10줄 단위로 분할)
$ ls -l
-rw-r--r-- 1 chang faculty 341 2월 16일 14:36 xaa
-rw-r--r-- 1 chang faculty 177 2월 16일 14:36 xab
-rw-r--r-- 1 chang faculty 518 2월 15일 19:33 you.txt
```

파일 합병: cat

- cat 명령어를 이용한 파일 합병 (파일 단위)

```
$ cat 파일1 파일2 > 파일3
```

파일1과 파일2의 내용을 붙여서 새로운 파일3을 만들어 준다.

- 예

```
$ cat xaa xab > xmerge
```

파일 합병: paste

- paste 명령어를 이용한 줄 단위 파일 합병

```
$ paste [ -s ][ -d구분문자 ] 파일*
```

여러 파일들을 줄 단위로 합병하여 하나의 파일을 만들어 준다.

-s : 한 파일 끝에 다른 파일 내용을 덧붙인다.

- 예

```
$ paste -s xaa xab > xmerge
```

split 명령어로 분리된 명령어를 다시 합할 경우 사용

cat 명령어로도 활용 가능 (== cat a b > c의 기본 동작)

파일 합병: paste 예 가로로 붙임

```
yu22312072@acslab-146:~$ paste a b
a   a
b   b
c   c
d   d
e   e
    f
yu22312072@acslab-146:~$ paste -s a b > res
yu22312072@acslab-146:~$ cat res
a   b   c   d   e
a   b   c   d   e   f
```

1번 파일의 첫줄과 2번 파일의 첫줄을 합하여 새로운 하나의 첫줄로 생성

- line.txt
  line 1:
  line 2:
  ...
  line 13:
  line 14:

내용

- $ paste line.txt you.txt
  line 1: When I am down and, oh my soul, so weary
  line 2: When troubles come and my heart burdened be
  ...
  line 13: But when you come and I am filled with wonder,
  line 14: Sometimes, I think I glimpse eternity

- $ paste line.txt you.txt > lineyou.txt

파일 비교: cmp 명령어

```
yu22312072@acslab-146:~$ cat a
a
b
c
d
e
yu22312072@acslab-146:~$ cat b
a
b
c
d
e
f
yu22312072@acslab-146:~$ cmp a b
cmp: EOF on a after byte 10, line 5

yu22312072@acslab-146:~$ diff a b
5a6
```

- $ cmp 파일1 파일2
  - 두 파일이 같은지 비교한다.
  - 두 파일이 같으면 아무 것도 출력하지 않음.
  - 두 파일이 서로 다르면 서로 달라지는 위치 출력

- $ cmp you.txt me.txt

  you.txt me.txt differ : byte 340, line 10

  (10번째 행 340번째 부터 문자가 다르다는 것을 의미)

보안 관련 설정 파일 검토(수정 여부 파악)용으로 사용 가능

• 보안 관련 설정 파일이 변경되었는지 확인하려면,

초기 파일을 저장한 후 나중에 이 파일이 변경되었는지 **cmp**를 통해 비교

diff 명령어

- $ diff 파일1 파일2
  - 두 파일을 줄 단위로 비교하여 그 차이를 출력
  - -i 옵션은 대소문자를 무시하여 비교
  - 출력은 첫 번째 파일을 두 번째 파일 내용과 같도록 바꿀 수 있는 편집 명령어 형태

- $ diff you.txt me.txt

  9a10,13 (첫번째 파일의 9줄 이후에 두번째 파일의 10~13줄 추가하면 두 파일이 같아짐을 의미, 아래는 추가해야할 두번째 파일의 줄들)

  >

  > You raise me up, so I can stand on mountains

  > You raise me up, to walk on stormy seas

  > I am strong, when I am on your shoulders (이상 10~13의 4줄)

```
yu22312072@acslab-146:~$ diff a b
1,2c1,2

4,6c4,6
```

diff 출력: 편집 명령어

- 추가(a)

  첫 번째 파일의 줄 n1 이후에 두 번째 파일의 n3부터 n4까지의 줄들을 추가

  n1 a n3,n4

  > 추가할 두 번째 파일의 줄들

- 삭제(d)

  첫 번째 파일의 n1부터 n2까지의 줄들을 삭제하면 두 번째 파일의 줄 n3 이후와 서로 같다.

  n1,n2 d n3

  < 삭제할 첫 번째 파일의 줄들

  파일의 앞부분이 같은 상황에서 n1과 n3 내용이 달라지는 시작 라인.
  1번 파일의 n1~n2를 삭제하면,
  n1과 같았던 2번 파일의 n3 이후로 계속 같아진다는 의미.

- 변경(c)

  첫 번째 파일의 n1부터 n2까지의 줄들을 두 번째 파일의 n3부터 n4까지의 줄들로 대치하면 두 파일은 서로 같다.

  n1,n2 c n3,n4

  < 첫 번째 파일의 대치될 줄들

  --

  > 두 번째 파일의 대치할 줄들

링크

- 링크
  - 기존 파일에 대한 또 하나의 새로운 이름

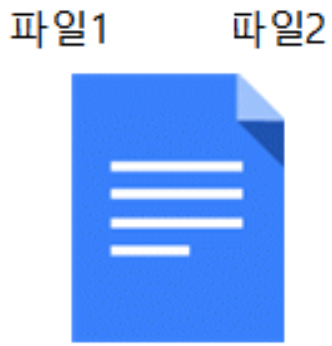
- 사용법

```
$ ln [-s] 파일1 파일2
```

파일1에 대한 새로운 이름(링크)로 파일2를 만들어 준다. -s 옵션은 심볼릭 링크

```
$ ln [-s] 파일1 디렉터리
```

파일1에 대한 링크를 지정된 디렉터리에 같은 이름으로 만들어 준다.



심볼릭 링크(symbolic link)

window - hard link 없음

- 심볼릭 링크
  - 다른 파일을 가리키고 있는 별개의 파일이다.
  - 실제 파일의 경로명을 저장하고 있는 일종의 특수 파일이다.
  - 이 경로명이 다른 파일에 대한 간접적인 포인터 역할을 한다.

  - 원래 파일의 위치에 대한 정보가 들어 있어서,
  - 심볼릭 링크를 참조하면, 가리키고 있는 파일을 대신 참조
  - 심볼릭 링크의 원본이 삭제되면, 참조 불가

```
$ ln -s hello.txt hi.txt
$ ls -l
-rw------- 1 chang faculty 15 11월 7일 15:31 hello.txt
lrwxrwxrwx 1 chang faculty 9 1월 24일 12:56 hi.txt -> hello.txt
```

hard는 l 생기지 않음

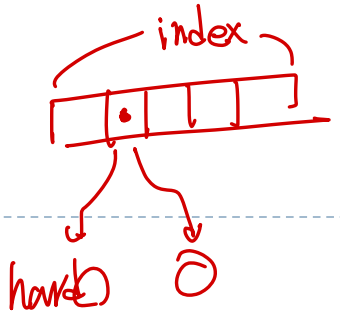
심볼릭 링크(symbolic link)

- 예

```
$ ln -s hello.txt hi.txt
$ ls -l
-rw------- 1 chang cs 15 11월 7일 15:31 hello.txt
lrwxrwxrwx 1 chang cs 9 1월 24일 12:56 hi.txt -> hello.txt

$ ln -s /usr/bin/gcc cc
$ ls -l cc
lrwxrwxrwx. 1 chang chang 12 7월 21 20:09 cc -> /usr/bin/gcc
```

하드 링크(hard link)



- 하드 링크
  - 기존 파일에 대한 새로운 이름이라고 생각할 수 있다.
  - 실제로 기존 파일을 대표하는 i-노드를 가리켜 구현한다.

index-노드

Symbolic 원본 sym

hard

- 예

```
$ ln hello.txt hi.txt
$ ls -l
-rw------- 2 chang cs 15 11월 7일 15:31 hello.txt
-rw------- 2 chang cs 15 11월 7일 15:31 hi.txt
```

hard link의 갯수 증가하면 숫자수

- 질문
  - 이 중에 한 파일의 내용을 수정하면 어떻게 될까?
  - 이 둘 중에 한 파일을 삭제하면 어떻게 될까?

symbol, hard, 원본 수정하면 모두 변경
(원본, sym, hard)

심볼릭 못봄
다시 원본 만들면 심볼릭 볼수 있다, 다시 이어진다
하지만 새로 생긴 원본가 기전에 있던 hard는 따로 나눠진다

링크와 복사

- 복사(cp)와 링크(ln)의 차이는?

하드링크

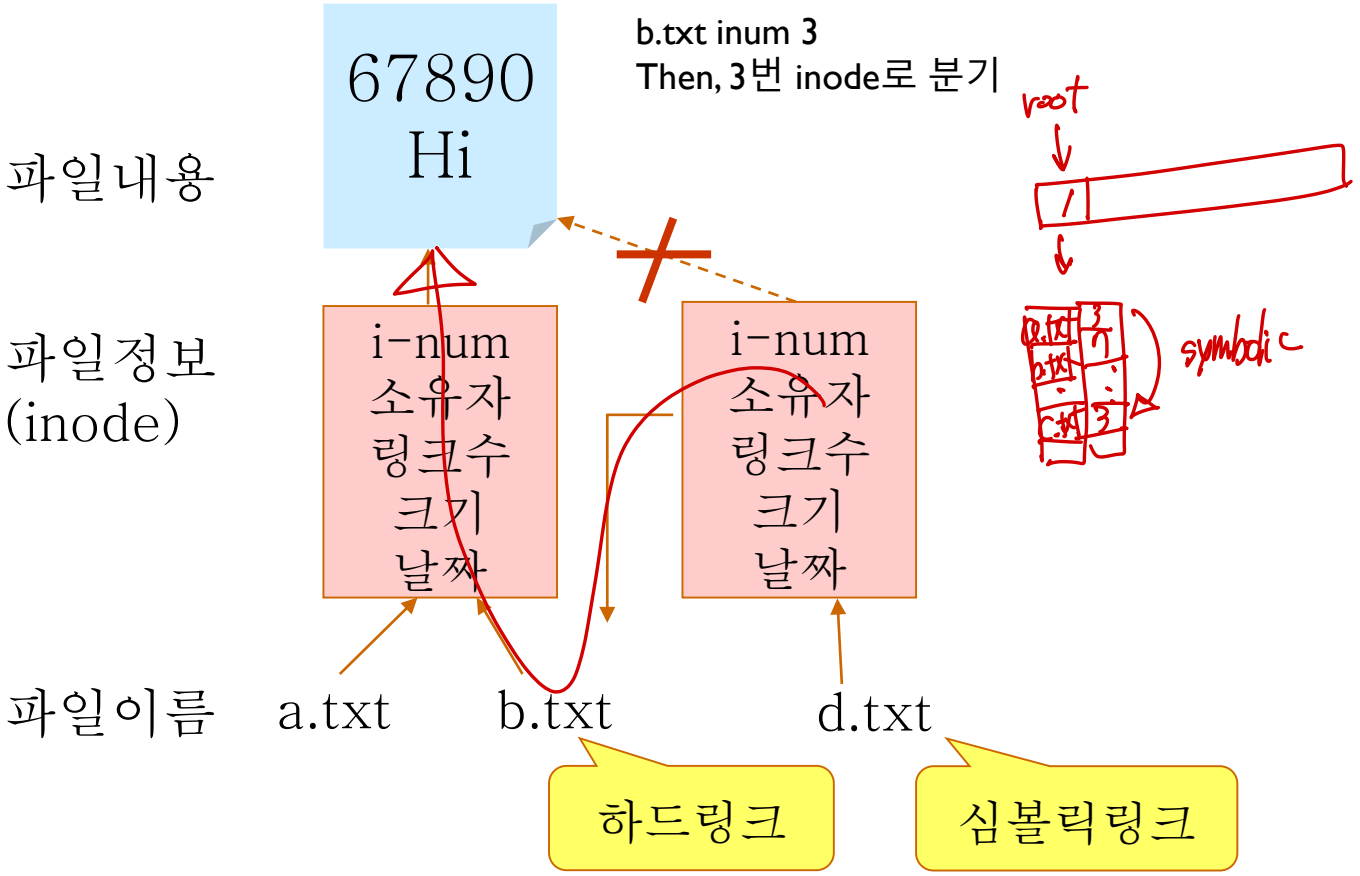
cp	ln
– 완전 별도 파일 생성 – 둘 중 하나를 수정해도 다른 파일에 영향 없음 – 같은 파일을 별도로 수정하여 작업할 때 사용	– 이름만 다르고 내용은 동일 – 둘 중 하나를 수정하면 두 파일이 같이 수정됨 – 파일을 공동으로 관리해야 할 때 사용

심볼링 링크
다른 파일에 대한 경로를 가리키는 별도의 파일
파일의 위치를 참조
원본 파일이 삭제되거나 이동되면 깨진 링크

링크 비교

- 하드링크와 심볼릭 링크

가령, /a.txt, /b.txt 가 존재하며 하드 링크라 하자.
Root inode number 0
a.txt inum 3
b.txt inum 3
Then, 3번 inode로 분기



링크 비교

- 디렉토리에 대한 하드 링크는 슈퍼 유저만 가능
  - 디렉토리에 대한 심볼릭 링크는 일반 사용자도 가능
- 물리적으로 서로 다른 파일 시스템 사이에는 하드링크 불가능
  - 심볼릭 링크는 가능
- 하드링크는 원본이 삭제되어도 접근 가능
  - 심볼릭 링크는 원본이 삭제되면 접근 불가
- 임의 파일에 대한 모든 하드 링크 찾기
  - find /dir_name -inum inode_number

파일 찾기: find 명령어

- find 명령어
  - 옵션의 검색 조건에 따라 대상 디렉터리 밑에서 해당되는 파일을 모두 찾아 출력

```
$ find 디렉터리 [-옵션]
```

옵션의 검색 조건에 따라 지정된 디렉터리 아래에서 해당되는 파일들을 모두 찾아 출력한다.

$ find 범위 표현식 동작

find 명령어 범위 표현

경로 표현	찾기 시작 위치
~	홈 디렉토리에서 찾기 시작
.	현재 디렉토리에서 찾기 시작
/etc	/etc 디렉토리에서 찾기 시작 (절대경로)
/	/(root) 디렉토리에서 찾기 시작 (전체파일 시스템 검색)
unix	unix 디렉토리에서 찾기 시작 (상대경로)

find 명령어 표현식 종류

시험 / 어떤 역할인지 알아두기

find . -name '*.txt' -type f

find . -name '*.txt' -size +10M
(10M이상) //MB는 안됨

검색조건표현	의미	기능
-name filename	파일이름	특정파일명에 일치하는 파일 검색 메타문자(*,?)사용도 가능하나 " "안에 있어야 함
-type	파일종류	특정파일종류에 일치하는 파일 검색 (f : file, d : directory, b, c, l, p, s)
-mtime [+|-]n -atime [+|-]n -ctime [+|-] n	수정(m), 접근 혹은 읽기 (a), 소유자 혹은 권한(c) 변경시간	m/a/c 시간이 +n일보다 오래되거나, -n일보다 짧거나 정확히 n일에 일치하는 파일 검색
-user loginID -group gname	사용자ID	loginID가 소유한 파일 모든 파일 검색 소유 그룹 또는 gid
-size [+|-]n [bck]	파일크기	+n보다 크거나, -n보다 작거나, 정확히 크기가 n인 파일 검색 b(512byte), c (byte), k (kilobyte) M:(1024^2) G:(1024^3)
-newer	기준시간	기준시간보다 이후에 생성된 파일 검색
-prune	검색 범위	서브 디렉토리로 내려가지 않고 현재 디렉토리에서만 검색
-perm	사용권한	사용권한과 일치하는 파일검색(8진수)

- find 경로명 ".." :

경로명 디렉터리에서 이름이 ..파일을 찾기

수정: 변경하고 저장했을 때
접근: 파일이 읽힌 시간 (cat)
권한: chmod

find 명령어: 검색 조건

검색 조건 및 처리 방법	설명
-name 파일명	파일명으로 찾는다.
-atime +n	접근 시간이 n일 이전인 파일을 찾는다.
-atime -n	접근 시간이 n일 이내인 파일을 찾는다.
-mtime +n	n일 이전에 수정된 파일을 찾는다.
-mtime -n	n일 이내에 수정된 파일을 찾는다.
-perm nnn	접근권한이 nnn인 파일을 찾는다.
-type x	파일 종류가 x인 파일들을 찾는다.
-size n	크기가 n 블록(512바이트)인 파일들을 찾는다.
-links n	링크 개수가 n인 파일들을 찾는다.
-user 사용자명	파일의 소유자가 사용자명인 파일을 찾는다.
-group 그룹명	그룹명을 갖는 그룹에 속한 파일을 찾는다.
-print	찾은 파일의 절대 경로명을 화면에 출력한다.
-ls	찾은 파일에 대해 ls -dils 명령어 실행 결과를 출력한다.
-exec cmd {};	찾은 파일들에 대해 cmd 명령어를 실행한다.

0 : 오늘 수정된 파일 (24시간 이내)

8진수 형식으로 입력

find 명령어 동작 종류

find . -name '*.txt' -exec rm {} \;
명령어에 따라 괄호 위치가 바뀜

동작	정의
-exec 명령 {} ₩;	exec 옵션은 ₩;으로 끝남 (명령은 ;로 끝나야 함. 특수 문자로 처리되지 않기 위해 shell escape ₩와 함께 사용) 검색된 파일은 {} 위치에 적용됨
-ok 명령 {} ₩;	exec의 확인모드 형태 사용자의 확인을 받아야 명령을 적용 (-i 옵션과 유사)
-print	화면에 경로명을 출력 (default 동작)
-ls	긴 목록형식으로 검색결과를 출력

표현식의 결합 기호

-a : and (기본) , -o : or , ! : not

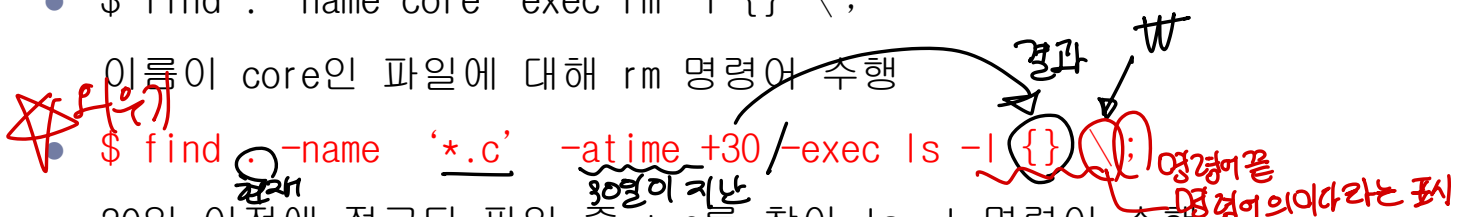
find 명령어 예

- $ find /usr -name '*.c' -print /usr 밑에 .c 파일들을 찾아 경로명 출력
- $ find . -name ping -ls 이름이 ping인 파일을 찾아 ls 명령어 수행
- $ find . -type d -print 디렉터리(d) 파일을 찾아 경로명을 출력
- $ find . -perm 700 -ls 사용권한 700인 파일 찾아 ls 명령어 수행
- $ find . -size +1024 -print 1024 블럭 이상인 파일을 찾아 출력
- $ find . -name core -size +2048 -ls

  파일 이름이 core이고 크기가 2048 블럭 이상인 파일을 찾아 ls 명령어 수행
- $ find . -user chang -print 파일 소유자가 chang인 파일을 찾아 출력
- $ find . -atime +30 -print 30일 이전에 접근되었던 파일을 찾아 출력
- $ find . -mtime -7 -print 7일 이내에 수정된 적이 있는 파일들 출력
- $ find . -name core -exec rm -i {} \;

  이름이 core인 파일에 대해 rm 명령어 수행
- $ find . -name '*.c' -atime +30 -exec ls -l {} \;

  30일 이전에 접근된 파일 중 *.c를 찾아 ls -l 명령어 수행

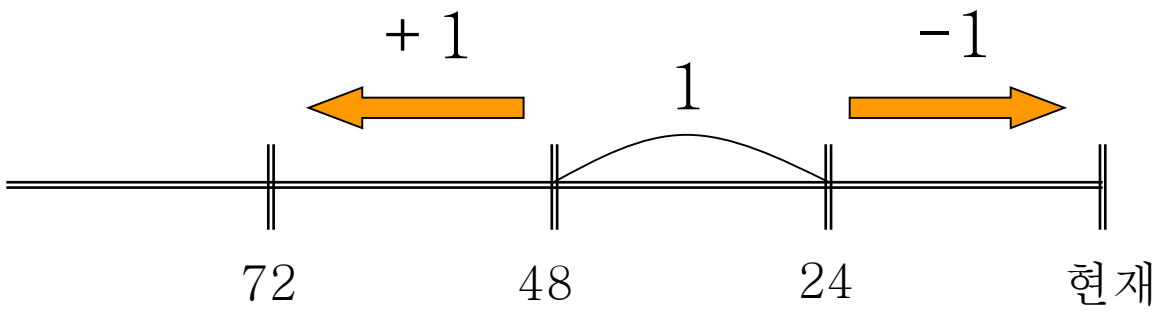


find 명령어 예

- -mtime (+/- : 24시간 기준)

telnet hanbit.co.kr

```
$ find . -mtime -1
./eg.dat
./eh.dat
./fg.dat
```



find 명령어: 검색 조건

- 파일의 접근권한(-perm)으로 검색

```
$ find . -perm 700 -ls
```

권한 찾아라

- 파일의 접근 시간(-atime) 혹은 수정 시간(-mtime)으로 검색

```
+n: 현재 시각을 기준으로 n일 이상 전
 n: 현재 시각을 기준으로 n일 전
-n: 현재 시각을 기준으로 n일 이내
```

```
$ find . -atime +30 -print
$ find . -mtime -7 -print
```

find 명령어: 검색 조건

- 파일의 소유자(-user)로 검색

```
$ find . -user chang -print
```

- 파일 크기(-size)로 검색

```
$ find . -size +1024 -print
```

기본 : 바이트
다른 용량으로 바꿀수 있음

- 파일 종류(-type)로 검색

```
d : 디렉터리        f: 일반 파일          l: 심볼릭 링크
b: 블록 장치 파일   c: 문자 장치 파일     s: 소켓 파일

$ find ~ -type d -print
```

사용자 홈 디렉토리

find 명령어: 검색 조건 조합

- find 명령어는 여러 검색 옵션을 조합해서 사용할 수 있다.

- 예

같은 명령어

```
$ find . -type d -perm 700 -print
$ find . -name core -size +2048 -ls
```

print 생략 가능

find 명령어: 검색된 파일 처리

- find 명령어의 -exec 옵션
  - 검색한 모든 파일을 대상으로 동일한 작업(명령어)을 수행

- 예

```
$ find . -name core -exec rm -i {} \;
$ find . -name *.c -atime +30 -exec ls -l {} \;
```

무조건 써야함

따음표를 안써도 됨
따음표를 쓸때는 띄어쓰기가 있을때 사용

명령어 스케줄링

cron/at 명령어

cron 시스템

- cron 시스템
  - 유닉스의 명령어 스케줄링 시스템으로
  - crontab 파일에 명시된 대로 주기적으로 명령을 수행한다.

- crontab 명령어
  - $ crontab 파일 — crontab 파일을 cron 시스템에 등록
  - $ crontab -l [사용자] — 사용자의 등록된 crontab 파일 리스트
  - $ crontab -e [사용자] — 사용자의 등록된 crontab 파일 수정/생성
  - $ crontab -r [사용자] — 사용자의 등록된 crontab 파일 삭제

- crontab 파일
  - 7개의 필드로 구성
  - 분 시 일 월 요일 [사용자] 명령

주기적 실행 cron

$ crontab 파일

crontab 파일을 cron 시스템에 등록한다.

$ crontab -l [사용자]

사용자의 등록된 crontab 파일 리스트를 보여준다.

$ crontab -e [사용자]

사용자의 등록된 crontab 파일을 수정 혹은 생성한다.

$ crontab -r [사용자]

사용자의 등록된 crontab 파일을 삭제한다.

crontab 파일 예

- chang.cron : 매일 18시30분에 파일 삭제

---

30 18 * * * rm /home/chang/tmp/* ← 분 시 일 월 요일 명령

- $ crontab chang.cron : cron 작업 등록

- 다른 crontab 파일의 예

  20 1 * * * find /tmp -atime +3 -exec rm -f {} ₩;

  매일 새벽 1시 20분에 3일간 접근하지 않은 /tmp 내의 파일을 삭제

  wall message 받기,안받기 설정하는 방법

  30 1 * 2,4,6,8,10,12 3-5 /usr/bin/wall /var/tmp/message

  2개월마다 수요일부터 금요일까지 새벽 1시 30분에 wall 명령을 사용해서 시스템의 모든 사용자에게 메시지를 전송

at 명령어 보안

- at 명령어
  - 미래의 특정 시간에 지정한 명령어가 한 번만 실행되도록 한다.
  - 실행할 명령은 표준입력을 통해서 받는다.
  - $ at [-옵션] 시간

- 예

```
$ at 1145 jan 31        1월 31일 11시 45분 다음 명령 실행
at> sort infile > outfile              ← 표준 입력으로 명령어 등록
at> <EOT>                (control-D 입력)
```

- 추가 기능
  - atq : at 시스템 큐에 등록된 작업 확인 (atq : at queue)
  - -f : 등록 명령을 파일로 부터 입력
  - -r : 등록 작업 삭제 (atq로 확인 후, 작업 번호 확인하여 이를 제거)

at 명령어

어떤 명령어가 저장되어있는지 확인

- atq 명령어
  - at 시스템의 큐에 등록되어 있는 at 작업을 확인
- 사용예

```
$ atq
Rank Execution Date Owner  Job Queue Job Name
1st  Jan 31, 2012 11:45 chang 1327977900.a a stdin
```

- at -r 옵션

$ at -r 작업번호 지정된 작업번호에 해당하는 작업을 제거한다.

- 사용 예

```
$ at -r 1327977900.a
```

작업제어 명령 – nohup

No hang up 전화를 끊다

후면처리와 같이 사용 가능
append한다 확인하기

nohup 백그라운드명령

- 백그라운드 작업을 실행시킨 단말기가 종료되거나 사용자가 로그아웃하면 실행중이던 백그라운드 작업은 함께 종료
  - 로그아웃한 다음에도 백그라운드 작업은 작업이 완료될 때까지 실행하도록 해야 할 때 nohup 명령 사용
- 명령의 실행결과와 오류메시지는 현재 디렉토리에 nohup.out파일로 자동적으로 저장
- 사용예

nohup find ~ -name passwd &

ls | grep nohup.out
== ls nohup.out

cat nohup.out

telnet hanbit.co.kr

```
$ nohup find / -name passwd  &
[1]   16454
$ exit
```

nohup.out파일에 결과저장

디스크 및 아카이브

df 유틸리티 (disk free)

- $ df 파일시스템* 0개 이상
  - 파일시스템에 대한 정보(사용 중/사용 가능한 디스크 공간)
  - 디스크의 여유 공간 확인

- $ df (df -h : 메가, 기가 단위로 표시)

```
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/sda3 49594228 7352576 39681696 16% /
/dev/sda5 86495548 66226168 15875608 81% /home
/dev/sdb1 141122196 103870536 30083060 78% /home1
/dev/sda1 101086 11455 84412 12% /boot
tmpfs 1037652 0 1037652 0% /dev/shm
```

이름 전체크기 사용중 가용공간 사용비율 마운트위치

du 유틸리티 (disk usage)

- $ du [-s] 파일명* (0개 이상)
  - 특정한 (혹은 현재) 디렉터리 내의 모든 파일들의 사용량 표시
  - 파일의 디스크 사용량(블록수)을 보여준다.
  - 파일을 명시하면 해당 파일의 사용량을 아니면

- $ du (현재 디렉토리의 모든 파일의 사용량 정보 표시)

```
12 ./htdocs/graphics
258 ./htdocs/images
42 ./htdocs/lecture/math
2582 ./htdocs/lecture/sp/lab
33196 ./htdocs/lecture/sp
...
```

- $ du -s -s(sum) 각 파일에 대한 전체 합계 표시

```
22164 .
```

아카이브

1. c (create): 새로운 tar 파일을 생성합니다. 여러 파일을 하나의 tar 파일로 묶을 때 사용합니다.
2. v (verbose): tar 명령어의 실행 과정을 자세히 출력합니다. 묶거나 추출하는 동안 진행 상태를 보여줍니다.
3. x (extract): tar 파일에서 파일을 추출(복원)합니다.
4. t (table of contents): tar 파일의 내용을 확인할 때 사용됩니다. 파일을 실제로 추출하지 않고 목록만 출력합니다.
5. f (file): tar 파일을 처리할 때 tar 파일의 이름을 지정합니다. f는 무조건 있어야 함

- 아카이브
  - 백업 또는 다른 장소로 이동을 위해 여러 파일들을 하나로 묶어놓은 묶음
  - 아카이브를 만들거나 푸는데 tar(tape archive) 명령어 사용

- tar 명령어
  - 옵션: c(create), v(verbose), x(extract), t(table of contents), f(file)
  - $ tar -cvf 타르파일 파일+

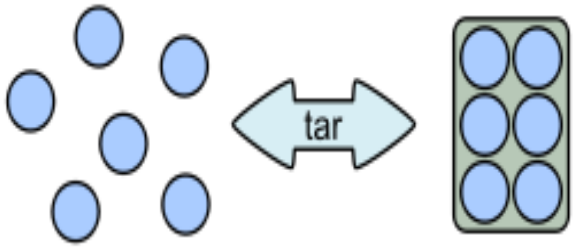
    압축할 파일 이름, 1개 이상

    파일 위치 헷갈릴 수 있음

    여러 파일들을 하나의 타르파일로 묶으며 보통 확장자로 .tar 사용
  - $ tar -xvf 타르파일

    하나의 타르파일을 풀어서 원래 파일들을 복원
  - $ tar -tvf 타르파일

    타르파일의 내용을 확인



압축 후
원본 파일 삭제
압축 풀기

```
yu22312072@acslab-146:~/ex5$ tar -tvf disk.tar
-rw-rw-r-- yu22312072/yu22312072 92 2024-10-14 16:40 xaa
-rw-rw-r-- yu22312072/yu22312072 87 2024-10-14 16:40 xab
-rw-rw-r-- yu22312072/yu22312072 112 2024-10-14 16:40 xac
-rw-rw-r-- yu22312072/yu22312072 101 2024-10-14 16:40 xad
-rw-rw-r-- yu22312072/yu22312072 106 2024-10-14 16:40 xae
```

아카이브: 예

- 현재 디렉터리에 있는 모든 파일을 다른 곳으로 옮기기

  $ tar -cvf src.tar * 모든 파일을 묶어놓음

  확장자명

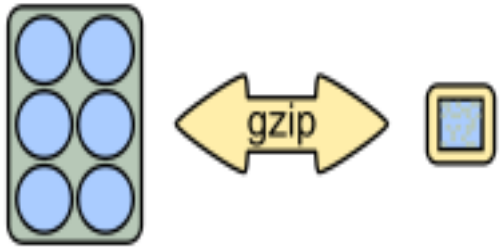
  src.tar를 다른 곳으로 이동

  $ tar -tvf src.tar
  $ tar -xvf src.tar
  $ tar -rvf src.tar test (기존 tar 파일에 파일 추가하기)

압축

- gzip 명령어

  ```
  $ gzip 파일*
  $ gzip -d 파일.gz*
  ```



- 사용 방법
  - 파일들을 하나의 타르파일로 묶은 후 compress/gzip을 사용해 압축
  - 파일 복원: 압축을 해제한 후, 타르파일을 풀어서 원래 파일들을 복원

파일 압축: gzip

0개 이상

$ gzip [옵션] 파일*

파일(들)을 압축하여 .gz 파일을 만든다.

-d : 압축을 해제한다.

-l : 압축파일 안에 있는 파일 정보(압축된 크기, 압축률) 출력한다.

-r : 하위 디렉터리까지 모두 압축한다.

-v : 압축하거나 풀 때 압축률, 파일명을 출력한다.

1개 이상

파일.gz+

$ gzip –d 파일.gz*

gzip으로 압축된 파일들을 복원한다.

$ gunzip 파일.gz*

gzip으로 압축된 파일들을 복원한다.

압축 예

- 사용예

```
$ tar -cvf src.tar *
$ gzip src.tar
```

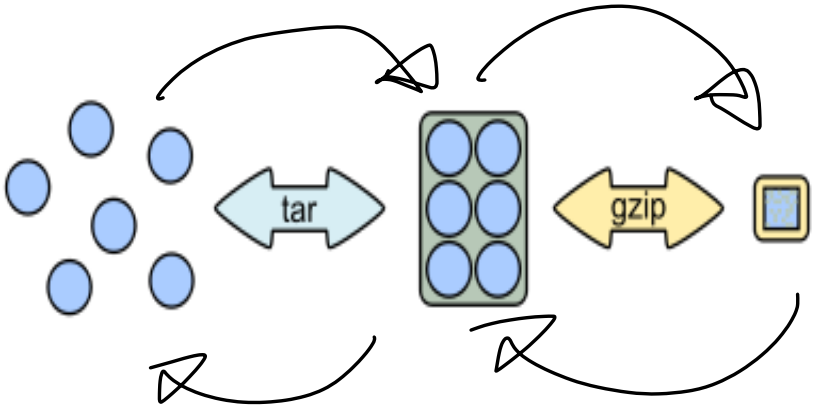
-> src.tar.gz

이 파일을 원하는 곳으로 이동

```
$ gzip -d src.tar.gz
$ tar -xvf src.tar
```

풀기

풀기



파일 압축: compress

있구나 정도 알기
많이 사용 안함

- 명령어 compress/ uncompress 명령어

> $ compress 파일*
>
> 파일(들)을 압축하여 .Z 파일을 만든다.
>
> $ uncompress 파일.Z*
>
> 압축된 파일(들)을 복원한다.

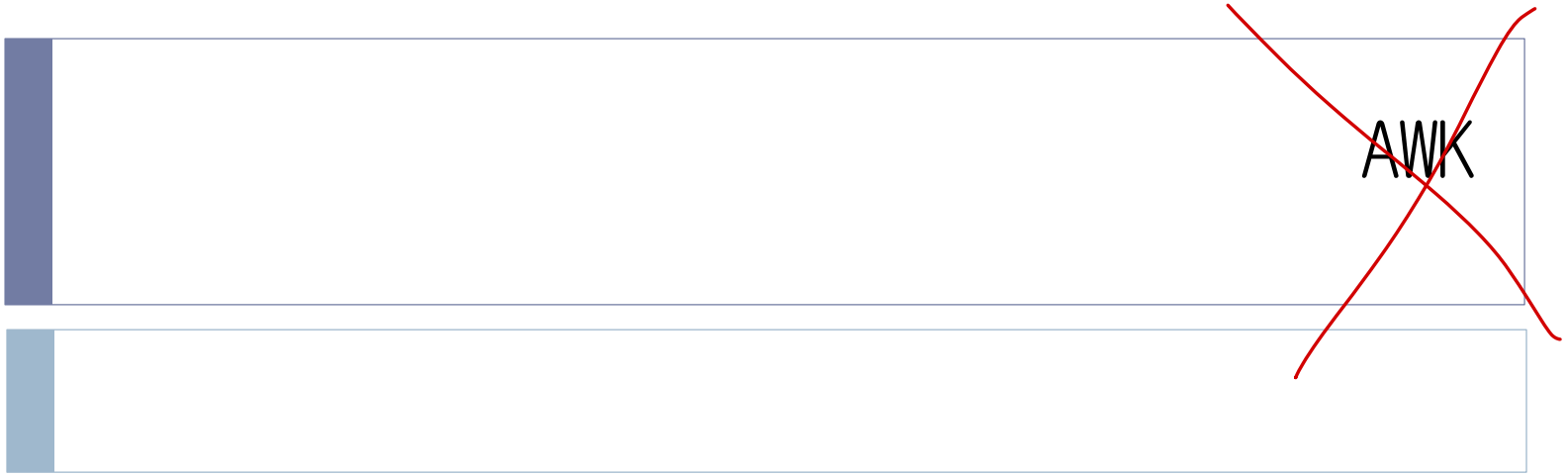
- 사용 예

```
$ ls
304 -rw-r--r-- 1 chang faculty 143360 Oct 8 2012 src.tar
$ compress src.tar
$ ls
54 -rw-r--r-- 1 chang faculty 27422 Oct 8 2012 src.tar.Z
$ uncompress src.tar.Z
$ ls
304 -rw-r--r-- 1 chang faculty 143360 Oct 8 2012 src.tar
```

압축 예

외우지 않기

파일 확장자	압축하기	압축풀기	파일내용보기
.z	pack file	unpack file	pcat file
.Z	compress file	uncompress file	zcat file
.gz	gzip file	gunzip file	gzcat file
.zip	zip file	unzip file	-


AWK

AWK

	Field1	Field2	Field3	Field4
Record1	박중훈	100	70	80
Record2	안성기	98	90	90
Record3	이영애	100	98	70
Record4	손예진	90	80	90
Record5	배용준	80	90	88

- AWK
  - 일반 스크립트 언어
  - AWK(Aho, Weinberger, Kernighan)
  - 텍스트 형태로 되어있는 줄을 필드로 구분하여 처리한다.
  - 필드는 줄을 구성하는 각각의 단어

- awk 프로그램
  - 간단한 프로그램은 명령줄에 직접 작성하여 수행
  - awk 스크립트를 별도의 파일로 작성하여 -f 옵션을 이용하여 수행
  - -F 옵션으로 필드 구분자를 c로 정할 수 있음 (default : 공백)

```
$ awk [-Fc] 프로그램 파일*
$ awk [-Fc] [-f 프로그램파일] 파일*
```

awk 프로그램

- awk 프로그램
  - 조건과 액션을 기술하는 명령어들로 구성됨
  - [조건] [{ 액션 }]
  - 대상 파일의 줄들을 스캔하여 조건을 만족하는 줄에 액션 수행

(1) 조건 없이 액션만 존재하는 경우,
$ awk '{ print NF, $0 }' you.txt (필드의 수와 줄 출력)
NF (number of field) 빌트인 변수, $0: 줄 전체

$ awk '{ print $1, $3, $NF }' you.txt (1, 3, 마지막 필드 출력)

(2) 조건과 액션이 같이 있는 경우,
$ awk 'NR > 1 && NR < 4 { print NR, $1, $3, $NF }' you.txt
NR (현재 줄의 줄 번호) 빌트인 변수
조건 (두번째 세번째 줄에 대해), 액션 (줄번호, 1,3,마지막 필드 출력)

핵심 개념 (고급 명령어 및 유틸리티)

- grep 명령어는 대상 파일들을 읽어서, 해당 패턴을 검색하고, 패턴을 포함하는 줄의 내용을 출력한다.
- sort 명령어는 텍스트 파일을 줄 단위로 정렬한다.
- cron은 유닉스의 명령어 스케줄링 시스템으로 crontab 파일에 명시된 대로 주기적으로 명령을 수행한다.
- 유닉스에서는 tar 명령어를 사용하여 여러 파일을 하나로 묶은 후에 compress 혹은 gzip 명령어를 이용하여 압축한다.
- awk 프로그램은 조건과 액션을 기술하는 명령어들로 구성되며 텍스트 파일의 줄들을 스캔하여 조건을 만족하는 각 줄에 대해 액션을 수행한다.