

기본 명령어

Basic Commands

- 명령어 구조
- 도움말 보기: man
- 파일 시스템 구조
- 파일 시스템 작업 명령
- 파일 내용 보기
- 프로세스 관련 명령
- vi 편집기

명령어 구조

□ 명령 실행 구조

- 명령행 (Command line)에서 명령어와 옵션, 인자 입력

```
$ ls -la /usr/bin/a*
```

prompt command options arguments

- 옵션: '-' 기호 뒤에 문자를 붙인다. 하나씩 써도 되고, 한꺼번에 여러 문자를 써도 된다.
- 인자: 주로 file 또는 directory 이름, 숫자 등 해당 명령 또는 프로그램에서 요구하는 명령행 입력
 - 메타문자 사용 가능
 - ? : 1개의 임의의 character
 - * : 0개 이상의 임의의 character
- 셸이 명령을 해석하여 해당 프로그램을 실행시킨다.

□ 특정 명령어나 표준 API에 대한 매뉴얼 출력 명령

```

cespc1.kumoh.ac.kr - PuTTY
[cespc1 : /juyoon]% man man
페이지를 다시 포맷 중입니다. 기다려 주십시오... 완료

User Commands  이름의 종류                                man(1)
NAME           이름과 기능 요약
man - find and display reference manual pages

SYNOPSIS      사용법
man [ - ] [ -adFlrt ] [ -M path ] [ -T macro-package ] [
-s section ] name ...

man [ -M path ] -k keyword ...

man [ -M path ] -f file ...

DESCRIPTION  설명
The man command displays information from the reference
manuals. It displays complete manual pages that you select
by name, or one-line summaries selected either by keyword
(-k), or by the name of an associated file (-f). If no
manual page is located, man prints an error message.

Source Format

```

□ man 페이지 구성 항목

항 목	의 미
NAME	해당 명령어에 대한 이름과 사용 목적을 간단히 설명
SYNOPSIS	해당 명령어에 대한 일반적 사용 형식
DESCRIPTION	해당 명령어에 대한 자세한 설명
FILES	해당 명령어가 사용하는 파일
SEE ALSO	해당 명령어에 대한 보다 더 많은 정보를 얻기 위해 참조해야 할 부분
DIAGNOSTICS	예상할 수 있는 에러에 대한 설명과 명령어가 실행했을 때 되돌려주는 에러코드 목록

□ 매뉴얼 섹션 (BSD UNIX와 Linux)

Section	Description
1	General commands
2	System calls
3	C library functions
4	Special files (devices) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellanea
8	System administration commands and daemons

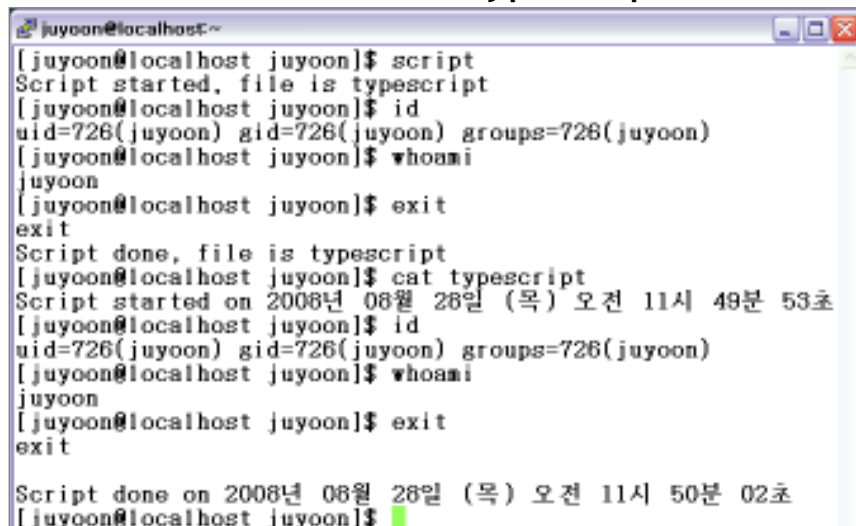
□ 옵션

- -a: 섹션별 모든 매뉴얼 페이지 출력

□ 현재 실행 중인 내용을 그대로 기록

```
script [-a] [filename]
```

- -a: 기존 파일에 덧붙임
- filename: 이름이 없으면 'typescript'로 만들어짐



```

juyoon@localhost~
[juyoon@localhost juyoon]$ script
Script started, file is typescript
[juyoon@localhost juyoon]$ id
uid=726(juyoon) gid=726(juyoon) groups=726(juyoon)
[juyoon@localhost juyoon]$ whoami
juyoon
[juyoon@localhost juyoon]$ exit
exit
Script done, file is typescript
[juyoon@localhost juyoon]$ cat typescript
Script started on 2008년 08월 28일 (목) 오전 11시 49분 53초
[juyoon@localhost juyoon]$ id
uid=726(juyoon) gid=726(juyoon) groups=726(juyoon)
[juyoon@localhost juyoon]$ whoami
juyoon
[juyoon@localhost juyoon]$ exit
exit
Script done on 2008년 08월 28일 (목) 오전 11시 50분 02초
[juyoon@localhost juyoon]$
  
```

- 파일 시스템 구조
- 파일의 속성과 유형
- 파일시스템 명령: ls, cd, mkdir, rm, cp, ...

File System

파일 시스템 구조

□ 파일 시스템

- 사용자가 볼 수 있는 자료는 모두 하드디스크에 있다.
- 하드디스크 자료의 논리적 구조 → 파일 시스템

□ UNIX/Linux 파일 시스템

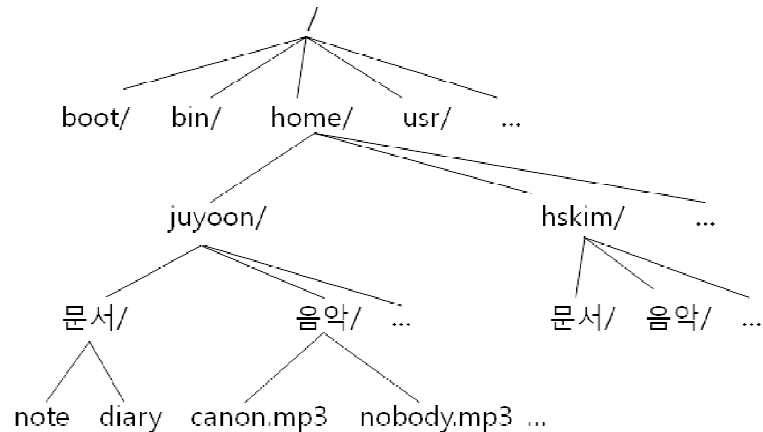
- 역 트리 구조
- **File**: 보조 저장 매체에 존재하는 정보의 논리적 저장 단위
- **Directory**: 많은 파일의 논리적 집합
 - 사실은 **목록 파일**: UNIX에서는 directory도 하나의 file
- 로그인 시 위치 → 자신의 **home** directory
 - 시스템 관리자가 계정을 만들 때 정해진다.

□ 역 트리 (reverse tree) 구조

□ 경로 (path)

- 절대경로
 - /로부터의 경로
- 상대경로
 - 현재 위치로부터 경로
- 특별한 경로 이름

.	현재 디렉터리
..	부모 디렉터리
~	홈 디렉터리



예)

현재 위치: /home/juyoon

diary의 절대경로: /home/juyoon/문서/diary

diary의 상대경로: 문서/diary

□ UNIX/Linux 파일시스템 구조의 관습적 표준

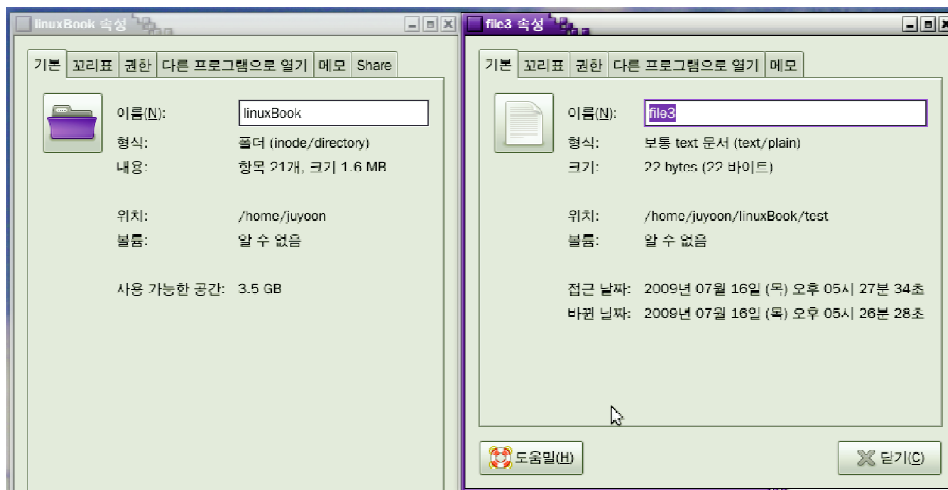
/boot	부팅에 필요한 모든 파일
/bin	핵심적인 사용자 명령 (binary) 프로그램 파일
/dev	장치 파일
/etc	텍스트로 된 설정 (configuration) 파일. 실행 파일은 올 수 없다
/home	각 사용자별 저장 공간. 로그인 시 시작 위치
/lib	핵심적인 공유 라이브러리와 커널 모듈
/media	플로피, CD/DVD 드라이브 등 탈부착 가능한 장치 파일
/mnt	임시로 연결할 파일시스템에 대한 마운트 지점
/opt	추가로 설치할 응용 소프트웨어 패키지
/root	시스템 관리자 (root)의 홈 디렉터리
/sbin	시스템 관리에 사용되는 프로그램
/usr	읽기 전용의 공유 데이터 저장

□ UNIX/Linux 파일시스템 구조의 관습적 표준

/usr/bin	거의 모든 사용자 명령 프로그램
/usr/include	C 프로그래밍에 사용되는 헤더 파일
/usr/lib	프로그래밍을 위한 라이브러리 및 패키지
/usr/local	시스템 관리자가 별도로 설치하는 소프트웨어
/usr/src	리눅스 커널의 소스 코드. 커널 헤더 파일도 여기
/var	시간에 따라 변하는 데이터
/proc	실행 중인 프로세스 및 시스템 상태에 대한 실시간 정보

□ 파일 관리의 기본

- 파일의 메타정보 (metadata)를 관리해야 한다.
 - 이름, 소유자, 생성날짜, 크기, 접근권한, ...



Directory 속성

일반 파일 속성

□ 일반 파일

- 문자(text)로 된 파일을 비롯하여 응용 프로그램에 따라 다양한 형식의 파일 존재
- 이름이나 확장자로 구분하지 않는다.

□ 특수 파일

- 시스템의 다른 자원을 파일로 취급

directory	여러 파일의 논리적 집합 단위. 계층 구조 형성
symbolic link	다른 파일에 대한 참조 표시 (파일을 찾아갈 수 있는 경로 정보 저장)
named pipe	실행 중인 프로세스 간에 정보를 전달하는 통로
socket	프로세스 간 정보 교환을 위한 통로
device	하드웨어 장치. character형과 block형으로 구분

□ 디렉터리

- 디렉터리에 속하는 파일들의 고유번호와 이름 기록

```
$ ls -ail
총 10
901531 drwxr-xr-x  3 juyoon  faculty    512  7월 21일  11:50 ./
2225129 drwxr-xr-x  4 juyoon  faculty    512  7월 21일  11:51 ../
957539  drwxr-xr-x  2 juyoon  faculty    512  7월 20일  11:32 dir1/
901532 -rw-r--r--   1 juyoon  faculty     6  7월 20일  11:32 file1
901533 -rw-r--r--   1 juyoon  faculty    10  7월 20일  11:32 file2
```

901531	.	\0				
2225129	.	.	\0			
901532	f	i	l	e	1	\0
901533	f	i	l	e	2	\0
957539	d	i	r	1	\0	

파일의 속성과 유형

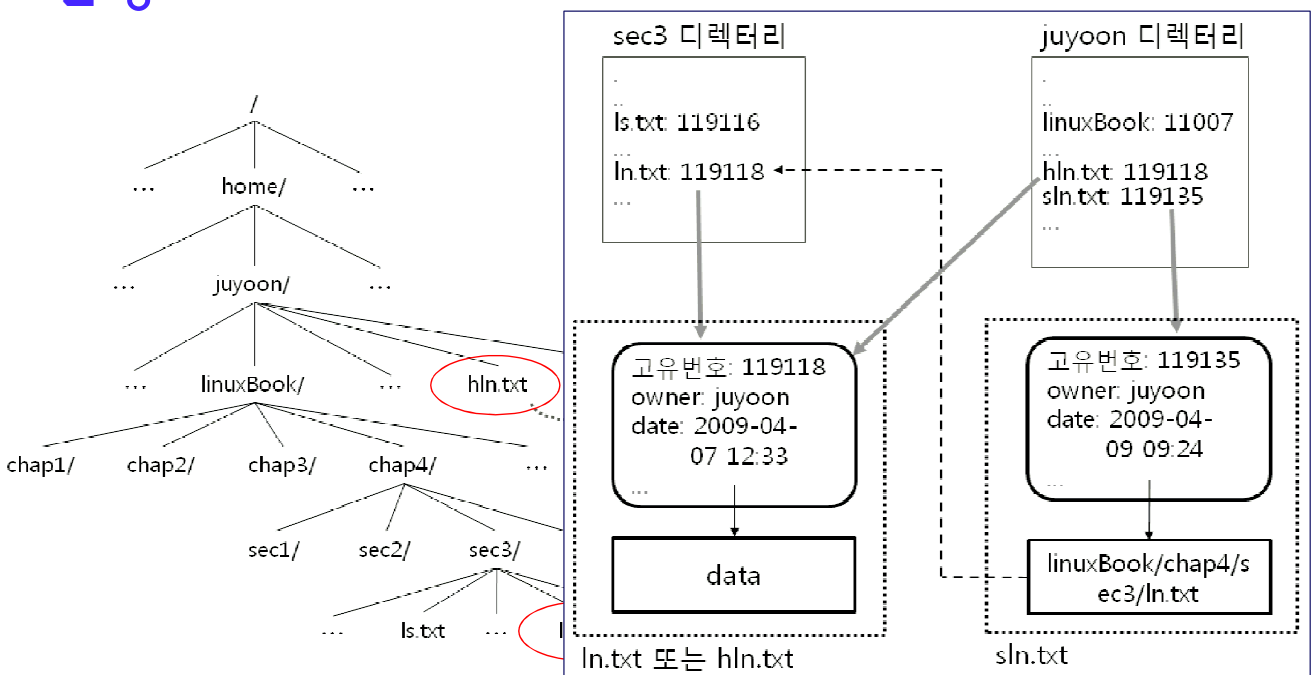
□ 링크 (link)

- 하나의 파일 실체에 대해 여러 개의 접근 경로 가능
 - hard link: 새로운 이름 (메타데이터)과 파일 실체 연결
 - soft(symbolic) link: 새로운 파일을 만들어 경로 정보 저장

상이한 요소	하드 링크	심볼릭 링크
디렉토리에 대한 링크	만들 수 없다	만들 수 있다
다른 파일시스템으로의 링크	만들 수 없다	만들 수 있다
target 존재 여부	반드시 존재해야 한다	존재하지 않아도 된다
파일 삭제 시 동작	참조 계수가 0이 될 때까지는 이름만 삭제되고 파일은 보존	링크 삭제 시 파일은 삭제되지 않으나 파일 삭제 시 링크 무효화

파일의 속성과 유형

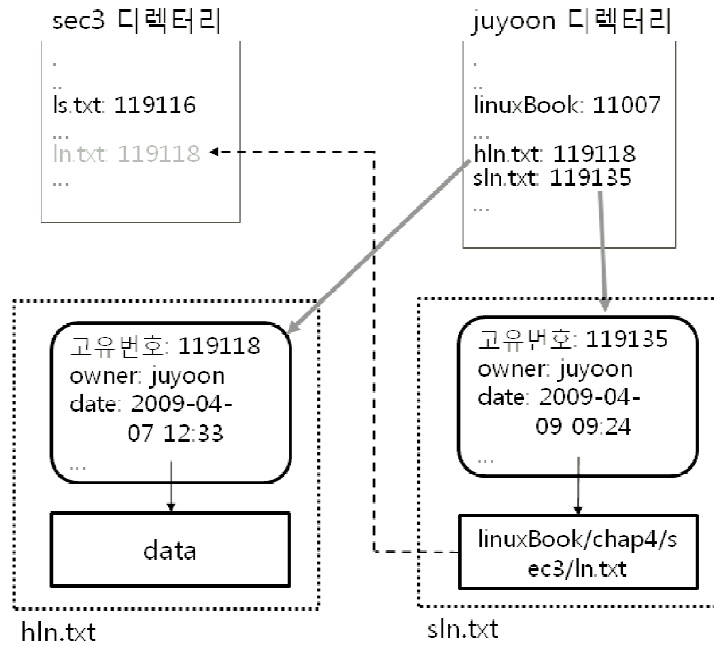
□ 링크



파일의 속성과 유형

□ 링크

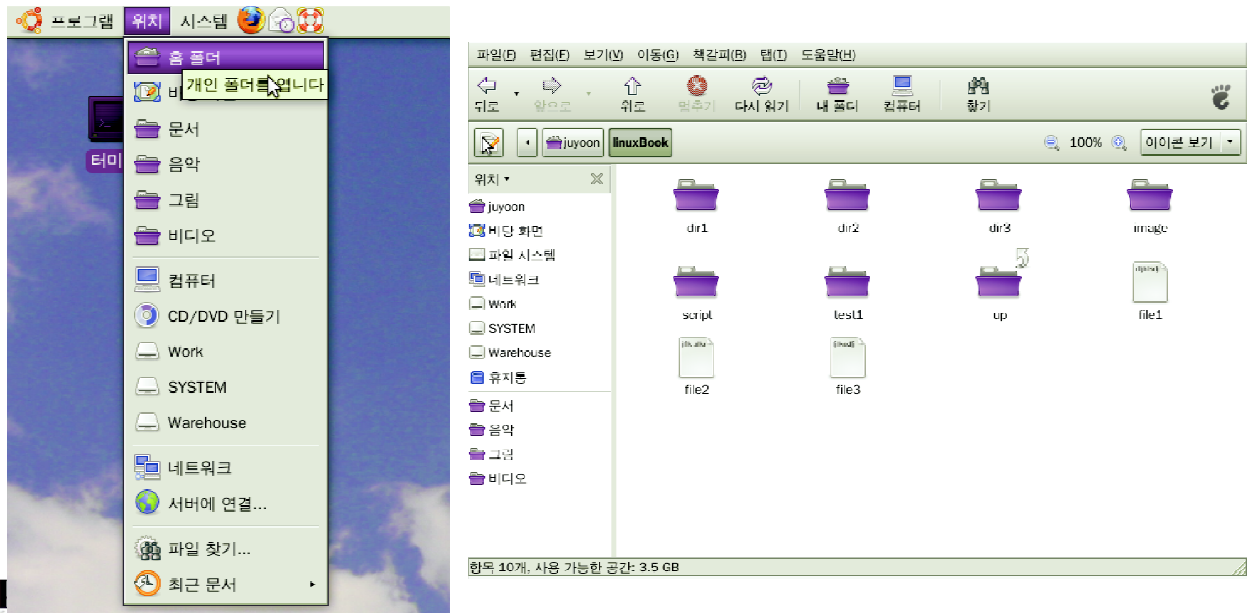
- 파일 삭제 시 하드 링크와 심볼릭 링크



파일 시스템 작업

□ 파일시스템 구조를 따라 파일 목록 보고 관련 작업하기

- GUI - 위치 메뉴 (파일 탐색기 nautilus) 실행



□ 명령어: pwd

- 현재의 작업 디렉터리 확인 (print working directory)

```

cespc1.kumoh.ac.kr - PuTTY
SunOS 5.8

login: juyoon
Password:
Last login: Fri Aug 22 21:21:16 from 59.23.235.107
Sun Microsystems Inc. SunOS 5.8 Generic Patch October 2001
[cespc1 : /juyoon]% pwd
/home1/faculty/juyoon
[cespc1 : /juyoon]%
  
```

□ 명령어: ls (list)

- 현재 디렉터리의 목록 보기
 - Subdirectory와 file들의 목록이 출력됨
- 주요 옵션
 - -a : hidden file도 모두 나열 (보통 .으로 시작한다.)
 - -F : 끝부분에 항목의 유형을 표시

/	Directory
=	Socket
@	Symbolic link
*	Executable file

- -l : 상세 정보를 보여 줌
 - 파일 모드
 - 링크 수
 - 소유자, 그룹, 크기 (bytes), 최종 수정 일시

□ 명령어: ls

■ 주요 옵션 (계속)

- -R : subdirectory의 목록을 recursive하게 보여 줌.
- -s : 파일의 크기를 block 수로 나타냄. (1 block = 512 bytes)
- 그 외에도 많음.

■ 인자: file 또는 directory 이름

- 생략: 현재 directory
- file 이름: 해당 file의 목록 정보만 나열
- directory 이름: 해당 directory에 수록된 모든 정보 나열

```

juyoon@linda: ~/test
juyoon@linda:~/test$ ls -asF system
합계 40
4 ./      12 cpC*   12 file*   0 home@
4 ../     4 cpC.c  4 file.c
juyoon@linda:~/test$
  
```

□ 파일의 모드 (mode)

```

juyoon@linda: ~/test
juyoon@linda:~/test$ ls -la system
합계 40 ②
drwxr-xr-x 2 juyoon juyoon 4096 2008-08-25 14:48 .
drwxr-xr-x 5 juyoon juyoon 4096 2008-08-25 14:47 ..
-rwxr-xr-x 1 juyoon juyoon 9547 2008-08-25 14:47 cpC
① -rw-r--r-- 1 juyoon juyoon 754 2008-08-25 14:47 cpC.c
-rwxr-xr-x 1 juyoon juyoon 8751 2008-08-25 14:48 file
-rw-r--r-- 1 juyoon juyoon 105 2008-08-25 14:48 file.c
lrwxrwxrwx 1 juyoon juyoon 12 2008-08-25 14:48 home -
> /home/juyoon
  
```

① 항목의 유형 (type)

d: directory

- : 일반 파일

b: 블록 유형의 특수 파일

c: 문자 유형의 특수 파일

l: 심볼릭 링크

s: 소켓

② 파일 접근 권한

□ 파일의 모드

▪ 접근 권한 표시

- owner, group, others로 구분

type owner group all
 ↓ ↓ ↓ ↓
drwxr-xr-x

r: read w: write
 x: execute -: no right

□ 명령어: cd

▪ Directory의 변경 (change directory)

▪ 인자: directory 이름

- 절대경로, 상대경로, 특수경로 이름 모두 사용
- 인자가 생략되면?
→ 자신의 home directory

```
juyoon@linda: /home
juyoon@linda: /usr/bin$ cd
juyoon@linda: ~$ cd /usr/bin
juyoon@linda: /usr/bin$ cd ..
juyoon@linda: /usr$ cd ~
juyoon@linda: ~$ cd test
juyoon@linda: ~/test$ cd ../../
juyoon@linda: /home$
```

□ 명령어: mkdir

- 새로운 directory 생성 (make directory)

```
mkdir [options] directory ...
```

- directory는 절대경로명 또는 상대경로명 모두 가능
- 하나 이상의 directory를 한꺼번에 생성 가능

```
cespc1.kumoh.ac.kr - PuTTY
[cespc1 : /juyoon]% mkdir /usr/juyoon
mkdir: 디렉토리 "/usr/juyoon" 작성 실패. 사용 권한이 거부됨
[cespc1 : /juyoon]% ls -dl /usr
drwxr-xr-x 31 root sys 1024 3월 6일 20:39 /usr/
[cespc1 : /juyoon]%
```

- 옵션

- -p : 지정된 directory의 중간 directory도 같이 생성
- -m *mode* : 접근권한을 부여해서 directory 생성

□ 명령어: rmdir

- Directory 삭제 (remove directory)

- 빈 directory만 삭제할 수 있다.

```
cespc1.kumoh.ac.kr - PuTTY
[cespc1 : /a]% ls
./ ../ sys/ test
[cespc1 : /a]% cd ..
[cespc1 : /test]% rmdir a
rmdir: 디렉토리 "a": 디렉토리가 비어있지 않음
[cespc1 : /test]% rmdir a/sys
[cespc1 : /test]% rmdir a
rmdir: 디렉토리 "a": 디렉토리가 비어있지 않음
[cespc1 : /test]%
```

- 옵션

- -p: directory 삭제 결과로 현재 directory가 비게 되면 상위 directory도 함께 삭제

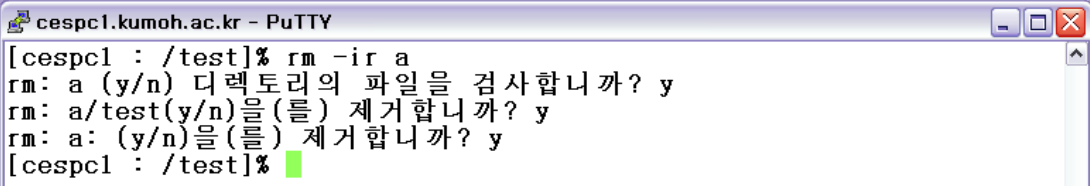
□ 명령어: rm

■ 파일 삭제 (remove)

```
rm [options] file ...
```

■ 옵션

- -i : 삭제 전에 사용자에게 확인 (interactive)
- -r 또는 -R : 순환적으로 directory 삭제
- -f : 확인하지 않고 모든 파일 삭제. 쓰기 금지된 파일도 삭제. 쓰기 금지된 directory 및 그 하위 파일은 삭제하지 않음.



```
cespc1.kumoh.ac.kr - PuTTY
[cespc1 : /test]% rm -ir a
rm: a (y/n) 디렉토리의 파일을 검사합니까? y
rm: a/test(y/n)을(를) 제거합니까? y
rm: a: (y/n)을(를) 제거합니까? y
[cespc1 : /test]% █
```

□ 명령어: cp

■ 파일 복사 (copy files)

```
cp [options] file1 file2
```

- file1을 file2로 복사

```
cp [options] file directory
```

- file을 directory 내부로 복사 (이름은 보존됨)

```
cp -r|R [options] directory1 directory2
```

- directory1의 내용을 모두 directory2로 복사
- 원본과 복사본은 별개의 파일이다.

■ 옵션

- -i : 복사 전에 사용자에게 확인 (interactive)
- -p : mode, 소유자, 그룹 등 속성을 보존하며 복사 (preserve)

□ 명령어: mv

- 파일 이름 바꾸기 또는 이동하기 (move files)

```
mv [options] file1 file2
```

```
mv [options] directory1 directory2
```

```
mv [options] file ... directory
```

- 여러 파일을 하나의 directory로 이동하기 가능
- 파일 본체는 동일하게 존재하고 이름과 위치만 바뀜
- 옵션
 - -i : 복사 전에 사용자에게 확인 (interactive)
 - -f : 이동할 새 이름의 파일이 이미 존재해도 무시하고 overwrite

□ 명령어: ln

- 링크 (link) 만들기

```
ln [options] target ... [link_name]
```

- link_name이란 새 이름으로 대상 파일 target에 대한 링크를 생성
- link_name이 생략되면 target과 같은 이름 사용 (절대 경로가 같으면 안 됨)
- link_name이 디렉터리고 target이 여러 개일 때 각 target에 대한 링크를 link_name 디렉터리 내에 생성
- 옵션
 - -s : 심볼릭 링크를 만든다.
 - -i : link_name과 같은 이름의 파일이 있으면 덮어쓰기 여부를 물어 본다.

□ 명령어: find

▪ 파일 찾기 (find files)

```
find [path ...] [expression]
```

- 지정된 path 이하의 모든 directory를 순환적으로 검색
- 여러 path를 한꺼번에 지정할 수 있다.
- path가 없으면? → 현재 directory (시스템마다 다르다.)
- expression이 없으면? → 지정된 directory 내의 모든 파일

▪ expression

- ‘_’로 시작하는 부분부터 Boolean expression으로 취급
 - ‘(’ 또는 ‘!’ 등으로 시작할 수도 있다.
- 파일을 찾기 위한 조건을 명시
- 여러 조건이 나열되면 ‘AND’의 의미

□ 명령어: find - 대표적 검색 조건

옵션	의미
- name 파일명	지정한 파일명이 찾은 파일명과 일치하면 출력된다. 이때 파일명으로 메타문자([, ?, *)와 함께 사용할 수 있다.
- type 파일형	지정한 파일형이 찾은 파일명과 일치하면 출력된다. 이때 사용되는 파일형은 다음과 같다. b : 블록 특수파일 c : 문자 특수파일 d : 디렉토리 p : 파이프인 fifo 파일 f : 일반파일 e : 연결파일
- user 로그인명	지정한 사용자 ID가 찾은 사용자 ID와 일치하면 출력한다.
- size 수	파일의 크기를 이용해서 찾는다. 파일 크기는 블록단위(1블록 = 512바이트)이다. 지정한 파일 크기보다 작은 파일을 찾으려면 파일 크기 앞에 - 기호를 붙이고 큰 파일을 찾으려면 + 기호를 붙인다.
- ls	파일의 속성 정보를 함께 출력한다. (ls 명령의 -l 옵션과 유사)
- atime 수	24시간 중 지정한 시간에 접근(access)된 적이 있는 파일을 찾는다.
- mtime 수	24시간 중 지정한 시간에 변경된 파일을 찾는다. (0을 지정하면 지난 24시간 동안을 의미한다.)
- exec 명령어	명령어를 실행한다. 이 명령어는 \ ; 으로 끝을 맺으며 명령어 인수 {}는 현재의 경로명으로 대체한다.

□ 명령어: find

```

juyoon@linda: ~/test/system
juyoon@linda:~/test/system$ ls -l
합계 144
drwxr-xr-x 2 juyoon juyoon 4096 2008-08-25 17:09 a
drwxr-xr-x 2 juyoon juyoon 4096 2008-08-25 17:09 b
drwxr-xr-x 2 juyoon juyoon 4096 2008-08-25 17:09 c
-rwxr-xr-x 1 juyoon juyoon 9547 2008-08-25 14:47 cpC
-rw-r--r-- 1 juyoon juyoon 754 2008-08-25 14:47 cpC.c
-rwxr-xr-x 1 juyoon juyoon 8751 2008-08-25 14:48 file
-rw-r--r-- 1 juyoon juyoon 105 2008-08-25 14:48 file.c
-rw-r--r-- 1 juyoon juyoon 105 2008-08-26 00:21 file1.c
-rw-r--r-- 1 juyoon juyoon 27 2008-08-26 00:22 file2.c
lrwxrwxrwx 1 juyoon juyoon 12 2008-08-25 14:48 home -> /home/juyoon
-rw-r--r-- 1 juyoon juyoon 74882 2008-08-26 00:10 less.man
-rw-r--r-- 1 juyoon juyoon 4756 2008-08-26 00:48 more.man
-rw-r--r-- 1 juyoon juyoon 103 2008-08-26 00:32 myfile
juyoon@linda:~/test/system$ find . -name "file*" -exec chmod 644 {} \; -ls
16680821 12 -rwxr-xr-x 1 juyoon juyoon 8751 8월 25 14:48 ./file
16680822 4 -rw-r--r-- 1 juyoon juyoon 105 8월 25 14:48 ./file.c
16680828 4 -rw-r--r-- 1 juyoon juyoon 105 8월 26 00:21 ./file1.c
16680829 4 -rw-r--r-- 1 juyoon juyoon 27 8월 26 00:22 ./file2.c
juyoon@linda:~/test/system$

```

□ 명령어: locate

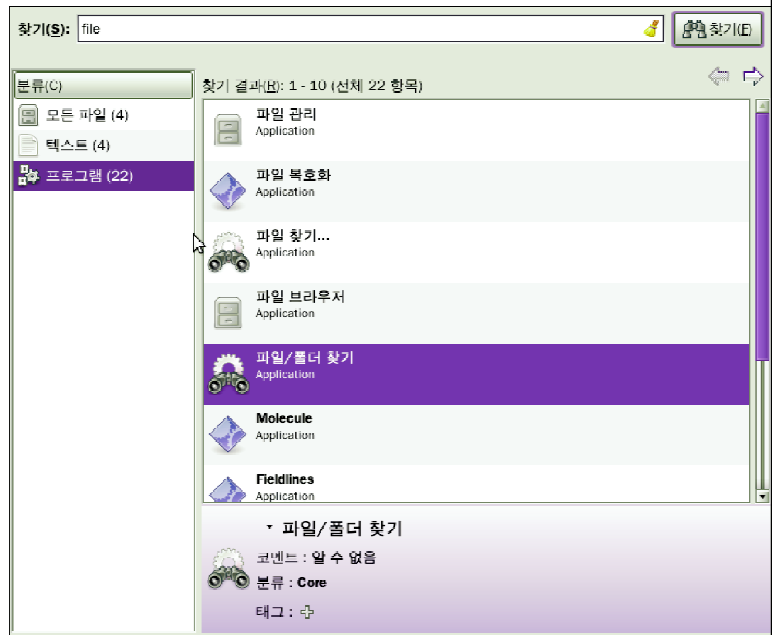
- 빠른 속도의 검색 프로그램 (**locate** files)
 - 파일 이름 등의 정보를 데이터베이스로 미리 생성
 - 검색 조건이 단순

```
locate [option] string
```

- 인자로 주어진 string과 일치하는 이름을 가진 파일을 검색해 위치 출력
 - 사용자가 접근 권한이 없을 때는 출력하지 않는다.
 - string에 메타 문자를 써서 표현할 수 있다.
- 옵션
 - -i : 대소문자를 구분하지 않음 (ignore case)
 - -n num : 결과 출력을 num개만 한다.

□ GUI 파일 검색 도구

- 파일 탐색기 (nautilus)에 연결된 “찾기” - 트래커 찾기 도구
 - 미리 인덱싱해야 함
시스템 → 기본설정
→ 찾기와인덱싱
- GUI find
 - 명령어 find, locate, grep의 시각화
 - GNOME/GTK
- kfind
 - KDE find



- cat
- more/less
- head/tail

파일 내용 보기

□ 기본적으로 text 파일 내용을 본다.

- UNIX 종류에 따라 binary 파일이나 device 등 특수 파일을 보기도 한다.

□ 명령어: cat

- 파일의 내용을 보여 주거나 연결 (concatenate)
- 결과는 표준 출력으로 내보냄

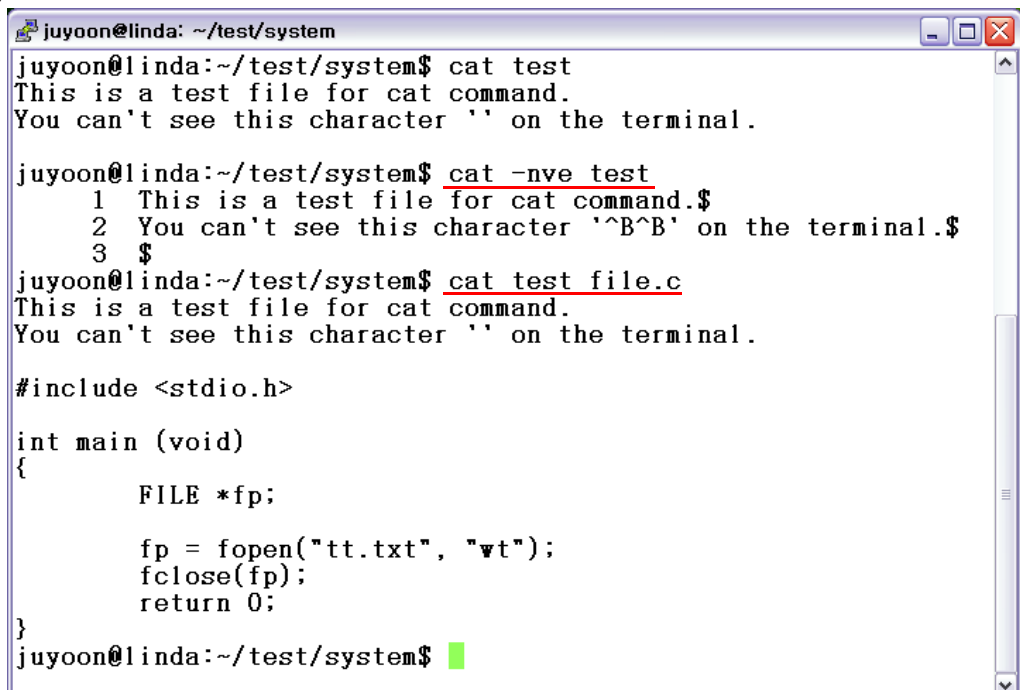
```
cat [options] [file ...]
```

▪ 주요 옵션

- -n : 행 번호를 앞에 붙인다.
- -v : tab, newline, form-feed를 제외하고 출력할 수 없는 문자를 보여 준다.
- -e : 각 행의 끝에 \$를 보여 준다. (-v와 함께 사용)

□ 명령어: cat

▪ 사용 예



```

juyoon@linda: ~/test/system
juyoon@linda:~/test/system$ cat test
This is a test file for cat command.
You can't see this character '' on the terminal.

juyoon@linda:~/test/system$ cat -nve test
 1 This is a test file for cat command.$
 2 You can't see this character '^B^B' on the terminal.$
 3 $
juyoon@linda:~/test/system$ cat test file.c
This is a test file for cat command.
You can't see this character '' on the terminal.

#include <stdio.h>

int main (void)
{
    FILE *fp;

    fp = fopen("tt.txt", "wt");
    fclose(fp);
    return 0;
}
juyoon@linda:~/test/system$ █
  
```

□ 명령어: more

- 한 화면에 맞추어 한 페이지씩 출력된다.

```
more [options] [+pattern] file ...
```


■ 주요 옵션

- -c : 스크롤하지 않고 화면을 지운 다음 출력
- -w : 출력이 끝난 후 빠져 나가지 않고 사용자 입력을 기다림.
- -lines# : 한 번에 # 줄씩 출력
- +/pattern: pattern에 해당하는 문자열을 찾아 거기서부터 출력

□ 명령어: more

- 이동 명령 - more로 출력 중인 파일 내에서 이동

- ? : 이동 명령에 대한 도움말
- <space> : 다음 페이지
- #f, #b : #개의 다음 또는 이전 페이지 건너뛰기 (default는 1)
- #<return> : 다음 #개 행의 텍스트 표시 (default는 1)
- = : 현재 행 번호 표시
- #s : 다음 n개 행의 텍스트 건너뛰기
- /pattern : 다음 pattern이 있는 곳으로 점프
- q : 마침



#는 임의의 숫자

□ 명령어: less

- more보다 옵션이 풍부하고 다양한 형태의 이동 가능
- 전체 파일을 적재하지 않고 보여주므로 속도가 빠르다.
- 한 페이지를 보여준 후의 프롬프트 - ‘.’
 - 다양한 이동 /검색 명령을 입력할 수 있다.
 - h : 도움말
- 자세한 내용은 매뉴얼을 참조하자! (A4 15pages ^^)

□ more, less -- 기본 명령어가 아닌 utility

- 시스템 종류와 버전에 따라 구현이 다르다.
- ‘man’ 명령어 수행 시 보여 주는 방식이 more 또는 less (시스템에 따라 다르다.)

□ 명령어: head, tail

- 파일의 시작 (head) 부분 / 끝 (tail) 부분 보기

```
head|tail [options] [file ...]
```

- 시작 또는 끝의 10줄을 출력
- 주요 옵션
 - -# 또는 -n # : 시작(끝) # 줄을 출력
 - -c# : # 바이트만큼 출력
- file이 생략되면?
 - 표준 입력

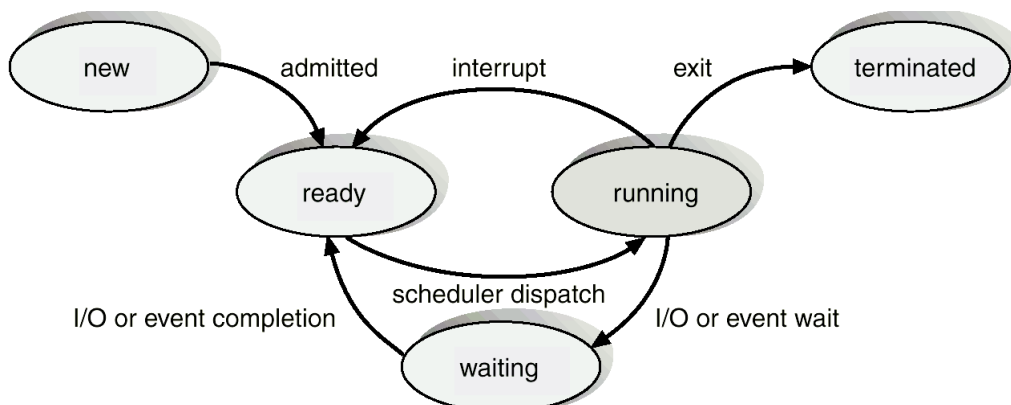
- ps
- background/foreground
- kill
- top

Process

프로세스

□ 프로세스에 대해 뭘 알지?

- 프로세스: 실행 중인 프로그램
⇔ 프로그램: 파일로 저장된 실행 가능한 코드
- 운영체제가 프로세스를 관리한다.
 - CPU 상태 저장, 메모리 사용 현황, 실행 상태 등
 - 프로세스 생명 주기(life cycle)



□ 사용자의 프로세스 생성

- 셸 또는 데스크톱에서 프로그램 실행
- 실행 중인 프로세스에서 프로세스 생성
 - 프로그램 내에 프로세스를 생성하는 코드를 넣는다.

□ 알고 보면?

- 모든 프로세스는 프로세스에서 생성한다.

- 셸도 프로세스
- 그러면 셸은 누가...?
 - 운영체제가.
- 그러면 운영체제는 누가...?
 - 부팅 시 모든 프로세스의 시조인 0번 (idle), 1번 (init) 프로세스 생성

□ 현재 실행 중인 프로세스 보기

□ 주요 옵션

- -e: 실행 중인 모든 프로세스
- -a: 로그인 중인 터미널과 관련된 모든 프로세스
- -f: 프로세스에 대한 모든 정보
- -l: 프로세스에 대한 상세 정보
- -H: 프로세스 트리 구조
- -t tty_no: 지정 단말 tty_no에서 실행 중인 프로세스
- -p PID: 지정하는 프로세스 번호(PID)의 정보
- -u UID: 지정하는 사용자 번호(UID)의 프로세스
- -g GID: 지정하는 그룹 번호(GID)의 프로세스

□ 프로세스 정보

```

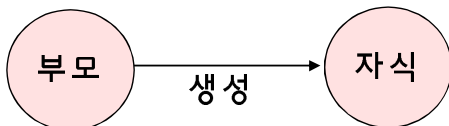
juyoon@ce:~/system
[juyoon:system/31]$ ps -l
F S  UID  PID  PPID  C PRI  NI ADDR  SZ WCHAN  TTY          TIME CMD
4 S  726  2100 2099  0  75   0  - 1363 wait4 pts/2      00:00:00 bash
0 R  726  3796 2100  0  75   0  -  775 -      pts/2      00:00:00 ps

```

- F: 플래그 또는 옵션
- S: 상태. S (Sleeping), R(Runnable), I (Idle), D(uninterruptable sleep), T(traced or stopped), Z(zombie)
- UID, PID, PPID: 사용자, 프로세스, 부모 프로세스 번호
- C: 스케줄링을 위한 프로세스 소모량
- STIME: 프로세스가 시작된 시간, 분, 초
- PRI: 프로세스 우선 순위. 값이 작을수록 우선순위가 높다.
- NI: nice 값 (우선 순위 계산에 사용)
- SZ: 프로세스의 자료와 스택 크기 (kbytes)
- WCHAN: 프로세스가 대상이 되는 주소
- TIME: 프로세스 실행에 걸린 시간
- CMD: 프로세스를 실행시킨 명령어

□ 프로세스 계통도

- ps -eH
- 프로세스 관계



- PPID를 거슬러 추적하면 "시조" 를 만난다!

```

juyoon@ce:~/system
PID TTY          TIME CMD
  1 ?             00:00:04 init
  2 ?             00:00:00 keventd
  3 ?             00:00:00 ksoftirqd_CPU0
  8 ?             00:00:00 bdflush
  4 ?             00:00:00 kswapd
  5 ?             00:00:00 kscand/DMA
  6 ?             00:00:00 kscand/Normal
  7 ?             00:13:34 kscand/HighMem
  9 ?             00:00:00 kupdated
 10 ?             00:00:00 mdrecoveryd
 16 ?             00:00:00 scsi_eh_0
 17 ?             00:00:00 scsi_eh_1
 20 ?             00:00:05 kjournald
 78 ?             00:00:00 khubd
184 ?             00:00:02 kjournald
185 ?             00:00:00 kjournald
186 ?             00:00:00 kjournald
187 ?             00:00:00 kjournald
188 ?             00:00:00 kjournald
189 ?             00:00:00 kjournald
190 ?             00:00:10 kjournald
688 ?             00:00:10 syslogd

```


□ 실시간으로 프로세스 실행 상황 모니터링

```

top - 10:20:33 up 4 min, 3 users, load average: 0.87, 0.81, 0.36
Tasks: 119 total, 2 running, 117 sleeping, 0 stopped, 0 zombie
Cpu(s): 2.6%us, 14.6%sy, 0.0%ni, 81.5%id, 0.0%wa, 1.3%hi, 0.0%si, 0.0%st
Mem: 380056k total, 297212k used, 82844k free, 10616k buffers
Swap: 875500k total, 5572k used, 869928k free, 129980k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM     TIME+  COMMAND
 2561 root        20   0 56576  21m 8128  S   8.6   5.9   0:16.97 Xorg
 3001 juyoon      20   0 6128  1740 1336  S   1.7   0.5   0:00.89 VBoxClient
 3049 juyoon      20   0 6000  1196  816  S   1.3   0.3   0:00.34 VBoxClient
 3179 juyoon      20   0 21540  10m 8396  S   1.3   2.7   0:02.75 metacity
 3273 juyoon      20   0 16908 2668 1448  S   1.3   0.7   0:01.78 gnome-screensav
 3275 juyoon      20   0 37956  14m 9.9m  R   1.3   3.9   0:01.93 gnometerminal
 3012 juyoon      20   0 6000  1148  808  S   0.7   0.3   0:00.40 VBoxClient
 3180 juyoon      20   0 44216  21m 15m  S   0.7   5.9   0:04.68 gnome-panel
  14 root        15  -5    0    0    0  S   0.3   0.0   0:00.48 ata/0
  30 root        15  -5    0    0    0  S   0.3   0.0   0:00.31 scsi_eh_1
2500 root        20   0 5100  1004 1592  S   0.3   0.5   0:00.10 hald-addon-stor
3181 juyoon      20   0 64012  24m 16m  S   0.3   6.5   0:05.14 nautilus
3298 juyoon      20   0 2448  1184  908  R   0.3   0.3   0:00.61 top
  1 root        20   0 3084   616  564  S   0.0   0.2   0:01.72 init
  2 root        15  -5    0    0    0  S   0.0   0.0   0:00.03 kthreadd
  3 root        RT  -5    0    0    0  S   0.0   0.0   0:00.00 migration/0
  4 root        15  -5    0    0    0  S   0.0   0.0   0:00.44 ksoftirqd/0
  
```

□ 프로세스 실행 제어 (셸 내장명령)

- **&**: 실행 시 background로 실행하도록 명령
- **bg**: 실행을 일단 중단 후 background로 전환
 - 표준입력은 받을 수 없으며 표준출력은 가능
- **fg**: background 프로세스를 foreground로 전환
 - 한 번에 하나의 프로세스만 foreground에서 실행
- **jobs**: 셸 내장 명령으로서 후면 프로세스를 보여 준다.
 - 별도의 job 번호를 붙임
 - bg → fg 선택 시 job 번호로 선택 가능

```

juyoon@ce:~/system
[juyoon:system/122]$ jobs
[2]- Stopped                loop
[3]+ Stopped                vim test
[juyoon:system/123]$ fg %2
loop
2
3
4
[2]+ Stopped                loop
  
```

□ 프로세스 강제 종료: kill

\$ kill [options] %job_no 또는 PID

- 셸 내장 명령 kill의 경우 작업 번호로 프로세스 종료
- 리눅스 명령어 kill
 - 명시된 PID를 가진 프로세스에게 **시그널(signal)**을 보낸다.
- 옵션
 - -l: 시그널의 목록 확인
 - -signal_number 또는 -signal_name
: 해당 시그널을 보낸다

\$ kill -KILL 4301

- default는 TERM (15): terminate

- pid가 -1: 실행 중인 셸만 제외하고 모두 종료

```
juyoon@ce:~/system
[juyoon:system/143]$ ps
  PID TTY          TIME CMD
 2100 pts/2        00:00:00 bash
 4301 pts/2        00:00:00 loop
 4419 pts/2        00:00:00 vim
 4431 pts/2        00:00:00 loop
 4433 pts/2        00:00:00 ps
[juyoon:system/144]$ kill 4301
[juyoon:system/145]$ kill -9 4419
[juyoon:system/146]$ jobs
[2]  Stopped                  loop
[3]-  죽었음                    vim test
[4]+  Stopped                  loop
[juyoon:system/147]$ kill %4
[4]+  Stopped                  loop
[juyoon:system/148]$ jobs
[2]-  Stopped                  loop
[4]+  종료됨                    loop
```

□ 정해진 시간에 프로세스 실행: at

\$ at time [date][+increment]

- 시간 지정
 - HH:MM - 같은 날의 시간 지정. AM, PM을 붙일 수 있다.
 - MM/DD/YY 또는 DD.MM.YY - 날짜 지정
 - + n {minutes, hours, days, weeks}
 - now, today, tomorrow, midnight, noon 등의 키워드

□ 관련 명령어

- atq: 등록된 작업 목록 보기 (= at -l)
- atrm: 등록된 작업 삭제 (= at -d)
- batch: 정해진 시간이 아니라 시스템 부하가 낮을 때 등록된 작업 수행

□ 기타 명령: sleep n

- n seconds 휴식

- 입력 모드
- 명령 모드
- 라인 편집 모드

텍스트 편집기: vi

vi 편집기

□ 텍스트 편집

- 짧을 때는 cat 명령을 사용할 수 있다.
- 긴 파일의 편집은 전문 편집기 필요

□ vi (Visual editor)

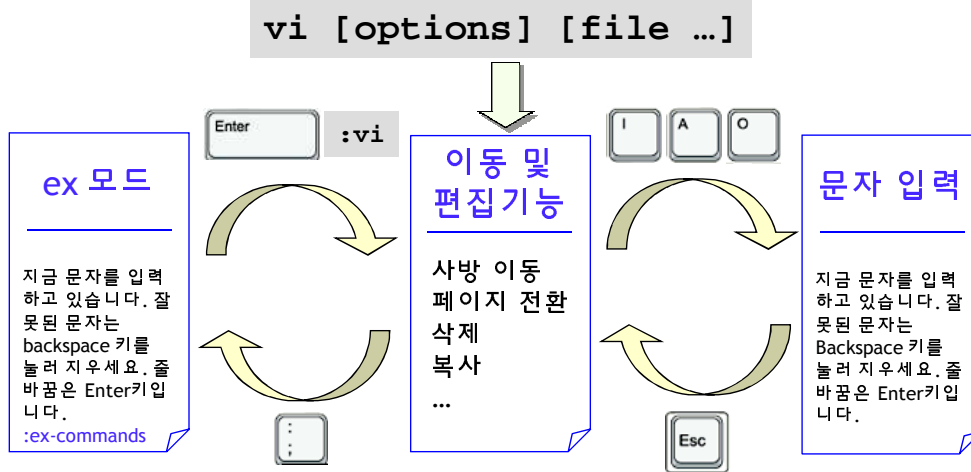
- vi 이전 편집기: ed, ex와 같은 line editor
- vi: 커서가 사방으로 돌아다니는 '획기적' 인 편집기
- 확장 키가 없던 환경에서 만들어져 기본 자판만으로 사용함.
- vim (vi improved) - syntax-oriented 기능 등 개선, 리눅스의 기본 편집기

□ 문서 편집기는 많다.

- CLI 환경의 emacs
- GUI 환경에서 OpenOffice를 사용하면 MS Office보다 강력한 기능을 활용한 문서 작성 가능

□ vi는 modal editor

- 두 가지 모드에서 행동이 다르다.
 - 입력 모드 (insert mode): 문자를 입력한다.
 - 명령 모드 (command mode): 편집 기능 수행. Escape 모드라고도 한다.



□ 입력 모드로 들어가기

- `i`: 현재 커서 위치부터 입력 (insert)
- `I`: 현재 행의 가장 앞에서부터 입력
- `a`: 커서 다음 위치부터 입력 (append)
- `A`: 현재 행의 가장 끝에서부터 입력
- `o`: 현재 커서 아래에 새 행을 만들어 입력 (open)
- `O`: 현재 커서 위에 새 행을 만들어 입력
- `R`: 현재 커서 위치부터 덮어쓰면서 입력 (replace)
- `s`: 현재 커서 위치의 한 글자를 다른 문자열로 대체 (substitute)
- `S`: 현재 커서 위치에서 라인 끝까지 지우고 새로운 텍스트 입력

□ 명령 모드

■ 커서 이동

- h(←), j(↓), k(↑), l(→) : 한 칸씩 이동
- w, b, e : 한 단어씩 이동. 다음 단어, 앞 단어, 다음 단어의 끝
- 각 명령 앞에 숫자를 붙이면 그만큼 반복
- ^, \$: 행의 맨 앞, 맨 뒤
- #G : #번째 행으로 이동
- G : 맨 마지막 행
- ' ' : 이전에 커서가 있었던 행
- ^f, ^b : 다음 페이지, 이전 페이지
- ^d, ^u : 다음 반 페이지, 이전 반 페이지
- H, M, L : 화면의 맨 앞, 중간, 마지막
- /string, ?string : string과 일치하는 다음 (/), 이전 (?) 문자열

□ 명령 모드

■ 삭제

- #x : #개의 글자 삭제 (default=1)
- #dd : #개의 행 삭제 (default=1)
- #dw : #개의 단어 삭제 (default=1)
- D : 현재 커서 위치부터 그 줄 끝까지 삭제
- J : 줄바꿈 문자 삭제. 즉, 두 행을 연결함.

■ 되돌리기

- u : 마지막 변경 명령에 대한 되돌리기. vim에서는 다중 되돌리기도 가능
- U : 커서가 위치한 행에 대한 변경 되돌리기.
- #. : 마지막 명령 #번 반복

□ 명령 모드

■ 대체

- #r : 한 문자 대체
- R : 현재 위치부터 ESC를 누를 때까지 대체
- s : 현재 문자 하나를 대체하고 입력 모드로 전환

■ 복사

- #yy : #개의 행을 복사 (default=1)
- #p : 복사 또는 삭제한 내용을 현재 커서 위치에 #번 붙이기

■ 기타

- ZZ : 변경 내용을 저장하고 셸로 돌아가기
- ^L : 화면 refresh

□ 라인 편집 모드

- 명령 모드에서 'Q' 또는 ':'을 입력하면 라인 편집 모드로 간다. ex 편집기 명령들을 사용할 수 있다.

■ 필수 명령

명령	결과
w[rite] [file]	지금까지 변경한 내용(버퍼에 저장되어 있음)을 파일에 저장한다. 파일 이름이 주어지지 않으면 vi 시작 시 열었던 파일에 저장한다.
w! [file]	쓰기 금지가 되어 있거나 기타 시스템의 이유로 파일에 저장할 수 없을 때 강제로 덮어쓰기한다. 다만 해당 파일에 대하여 쓰기 권한을 가진 사용자여야 한다.
r[ead] [file]	file을 읽어 현재 커서가 있는 줄 아래 붙여 넣기한다.
e[dit] [file]	편집 중인 파일을 빠져 나와 인자로 준 file 편집을 시작한다.
q[uit]	편집기를 마치고 셸로 나간다(quit). 파일이 변경되었는데 저장을 하지 않은 경우 여러 메시지를 출력하고 편집기에 머문다.
q!	저장하지 않고 셸로 나간다.
wq, x[it]	파일의 변경 사항을 저장하고 셸로 나간다. 명령 모드에서 ZZ와 같은 효과다.
Enter, vi	vi 명령 모드로 돌아 간다.
he[[p]	온라인 매뉴얼을 출력한다.
!bash	하위 셸을 실행한다. 셸 실행이 끝난 후 exit를 입력하면 vi로 되돌아온다.
se[t]	여러 가지 환경 설정을 한다.

□ 라인 편집 모드

▪ 편집 명령

- vi 명령 모드와 유사한 명령이 모두 존재
- 명령어 양식: [range]명령[인자]
- range : 주소, 주소

주소	의미	주소	의미
.	현재 커서가 위치한 줄	\$	파일의 마지막 줄
num	num 번째 줄	+num	현재 위치에서 다음 num번째 줄
-num	현재 위치에서 이전 num번째 줄	%	파일 전체

- 명령: 매우 많으나 자주 사용되는 것은...
 - d[ele]te, g[lobal]/pattern/[명령], v/pattern/[명령], s[ubstitute]/[pattern/replace/][options][#]

`:1,$s/text/test/g` ← 1행에서 끝 행까지 모든 text를 test로 치환 (substitute)

□ 저렇게 많고 복잡한 명령들을 어떻게 외우나?

- 자주 사용하면 머리와 손에 '각인' 된다!
- 매일매일 사용하자!

마우스 vs. 키보드

어느 쪽이 손/손가락/손목/팔꿈치에 더 나쁠까?
어느 쪽이 두뇌 개발에 더 좋을까?

vi 같은 고전적 편집기를 아직도 써야 하나?



이거 배워서 어디다 써먹나?
노력 대비 효과가 너무 낮은 거 아닌가?