2장 리눅스 사용

리눅스 시스템 프로그래밍

청주대학교 전자공학과
한철수

목차

- 기본 명령어
- 파일 및 디렉터리
- 파일 관련 명령어
- 파일 속성
- 입출력 재지정 및 파이프
- 텍스트 편집기

기본 명령어

$ date	// 날짜 및 시간을 출력함.
$ hostname	// 호스트명을 출력함.
$ uname	// 시스템 정보를 출력함.
$ who	// 현재 로그인한 사용자 정보를 출력함.
$ ls	// 현재 디렉터리 내의 파일 목록을 출력함.
$ passwd	// 패스워드를 변경함.
$ clear	// 화면을 지움.
$ man *명령어*	// *명령어*에 대한 매뉴얼을 보여줌.

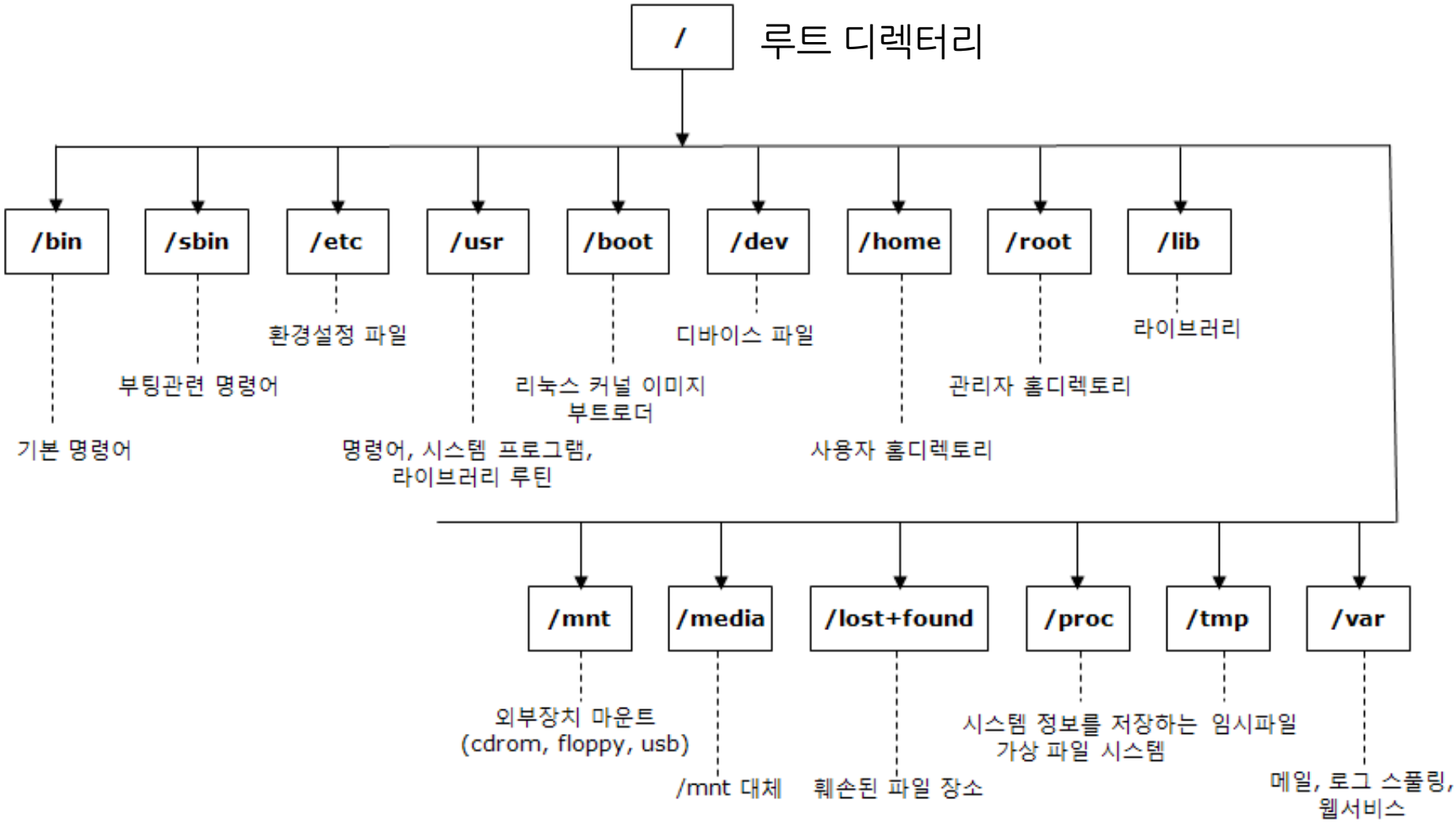
파일 및 디렉터리

- 리눅스에서는 데이터를 입출력 할 수 있는 대상은 모두 파일로 간주함.
  - 즉, 디스크에 저장된 파일 뿐만 아니라 입출력 장치들도 파일로 간주함. (교재 15쪽 유닉스의 설계 철학 중 단순성에 해당함.)
- 리눅스의 파일 종류
  - 일반 파일
    - 데이터를 가지면서 저장장치에 저장된 것.
  - 디렉터리(directory)
    - 데이터를 가지면서 저장장치에 저장된 것.
    - 다른 파일들을 조직하고 사용하는데 필요한 정보를 갖고 있음.
    - 폴더(folder)라고도 함.
  - 특수 파일
    - 물리적 장치에 대해서 파일로 간주한 것.
      - 키보드: stdin
      - 모니터: stdout

디렉터리 계층구조

- 한 디렉터리는 다른 디렉터리들을 포함함으로써 계층구조를 이룸.
  - 부모 디렉터리
    - 다른 디렉터리를 포함하는 디렉터리
  - 서브 디렉터리(하위 디렉터리)
    - 부모 디렉터리에 포함된 디렉터리들
- 리눅스 파일 시스템은 루트(root, /) 디렉터리부터 시작하여 하위 디렉터리들이 형성됨.

리눅스 주요 디렉터리


/
루트 디렉터리
/bin
/sbin
/etc
/usr
/boot
/dev
/home
/root
/lib
기본 명령어
부팅관련 명령어
환경설정 파일
명령어, 시스템 프로그램,
라이브러리 루틴
리눅스 커널 이미지
부트로더
디바이스 파일
사용자 홈디렉토리
관리자 홈디렉토리
라이브러리
/mnt
/media
/lost+found
/proc
/tmp
/var
외부장치 마운트
(cdrom, floppy, usb)
/mnt 대체
훼손된 파일 장소
시스템 정보를 저장하는 임시파일
가상 파일 시스템
메일, 로그 스풀링,
웹서비스

홈 디렉터리와 현재 작업 디렉터리

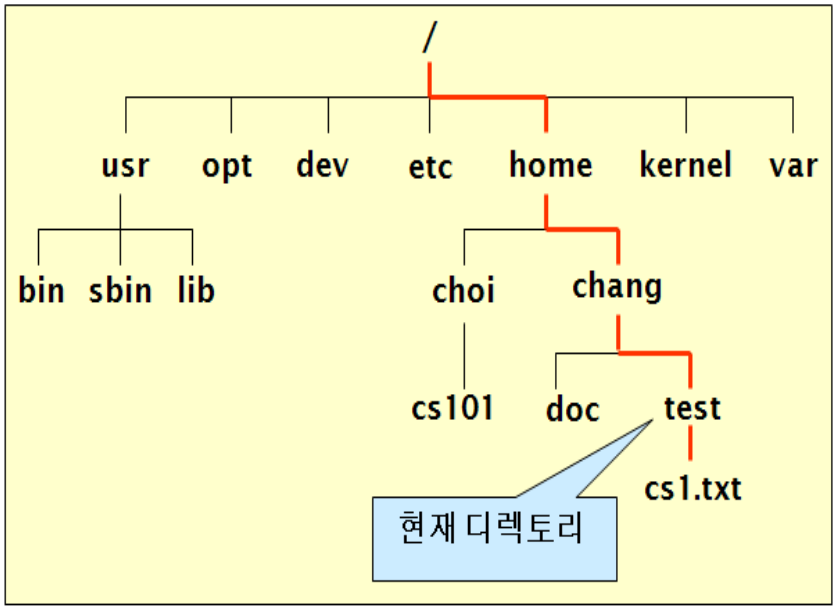
- 리눅스는 다중 사용자 시스템이기 때문에 각 사용자마다 별도의 홈 디렉터리(home directory)가 있음.
- 홈 디렉터리
  - 사용자 계정을 만들 때 관리자에 의해 정해짐.
  - 사용자가 로그인하면 홈 디렉터리에서 작업을 시작하게 됨.
  - 홈 디렉터리의 이름은 일반적으로 사용자명과 같음.
  - 환경 변수 $HOME에 사용자의 홈 디렉터리 이름이 저장됨.
    - $ echo $HOME
- 현재 작업 디렉터리(current working directory)
  - 현재 작업 중인 디렉터리

디렉터리 관련 명령어

- pwd (print working directory)
  - 현재 작업 디렉터리를 출력함.
  - $ pwd
- mkdir (make directory)
  - 새 디렉터리를 만듦.
  - $ mkdir *만들 디렉터리명*
  - $ mkdir test // test라는 서브 디렉터리를 만듦.
- cd (change directory)
  - 현재 작업 디렉터리를 이동함.
  - $ cd [*이동할 디렉터리명*]
  - $ cd test // test 디렉터리로 이동.
  - $ pwd
  - $ cd // 홈 디렉터리로 이동.
  - $ pwd

경로명(pathname)

- 경로명이란 파일이나 디렉터리에 대한 이름을 말함.
- 경로명의 종류
  - 절대 경로명
    - 경로명을 루트 디렉터리로부터 시작하여 모두 적은 것.
  - 상대 경로명
    - 경로명을 현재 작업 디렉터리로부터 시작하여 적은 것.



cs1.txt의 절대 경로명
/home/chang/test/cs1.txt

cs1.txt의 상대 경로명
cs1.txt

~ : 홈 디렉터리
. : 현재 디렉터리
.. : 부모 디렉터리

디렉터리 기호

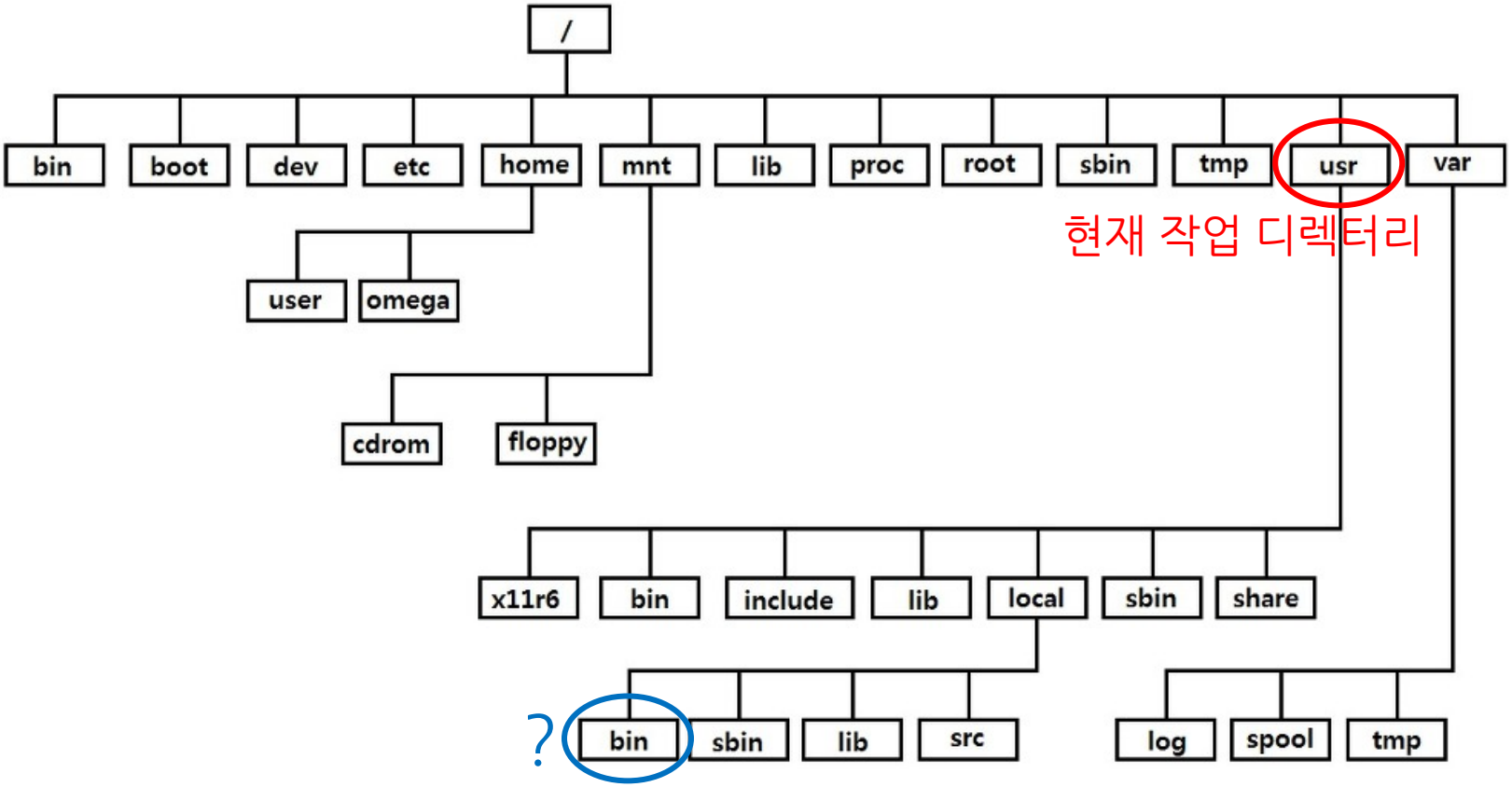
- 다음 기호를 이용하여 홈 디렉터리, 현재 디렉터리, 부모 디렉터리를 지정 할 수 있음.

~ : 홈 디렉터리 . : 현재 디렉터리 .. : 부모 디렉터리

```
cd ..        // 부모 디렉터리로 이동함.
cd ~         // 홈 디렉터리로 이동함.
```

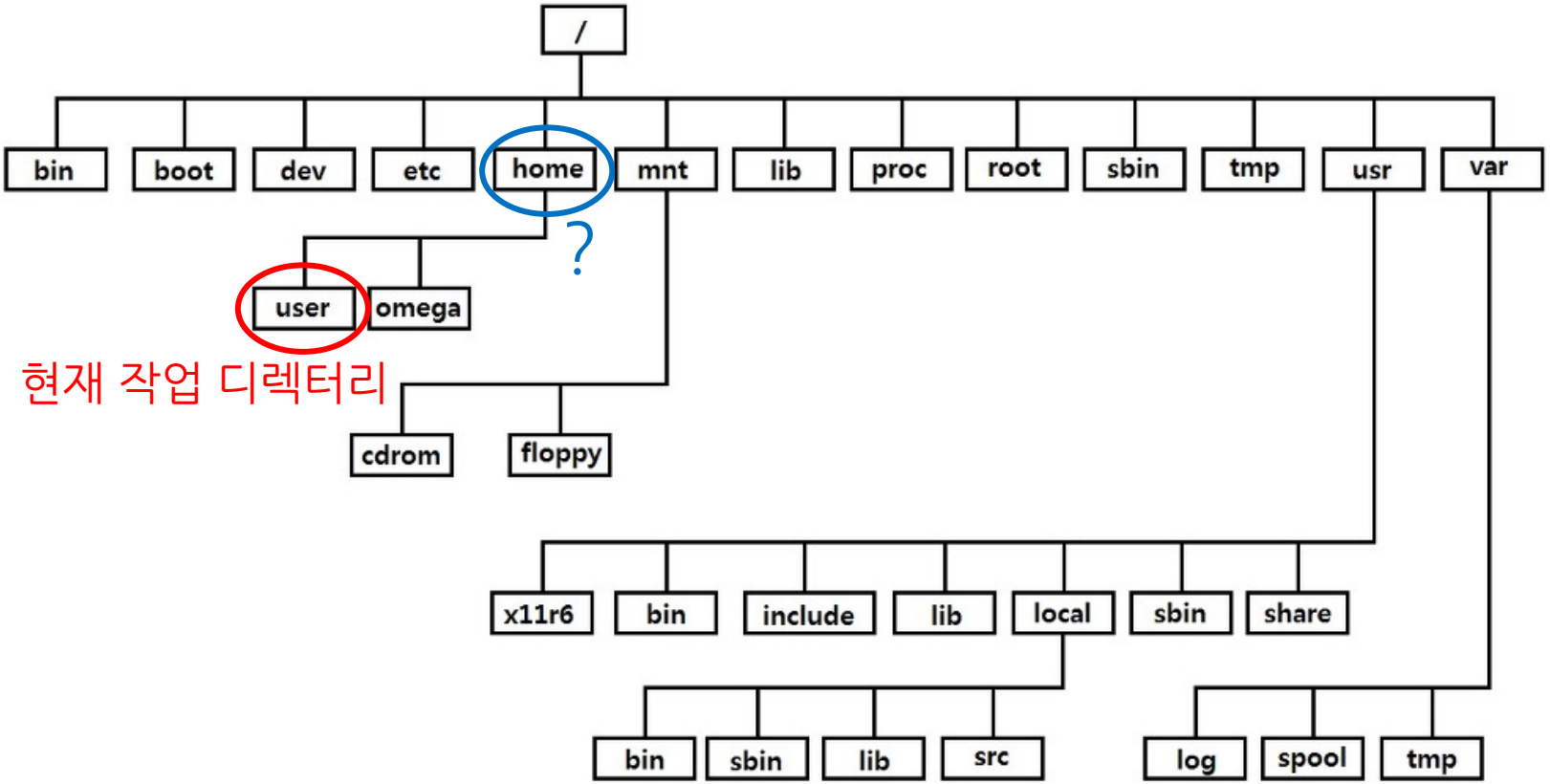
경로명 연습 (1)

- bin 디렉터리
  - 절대 경로:
  - 상대 경로:



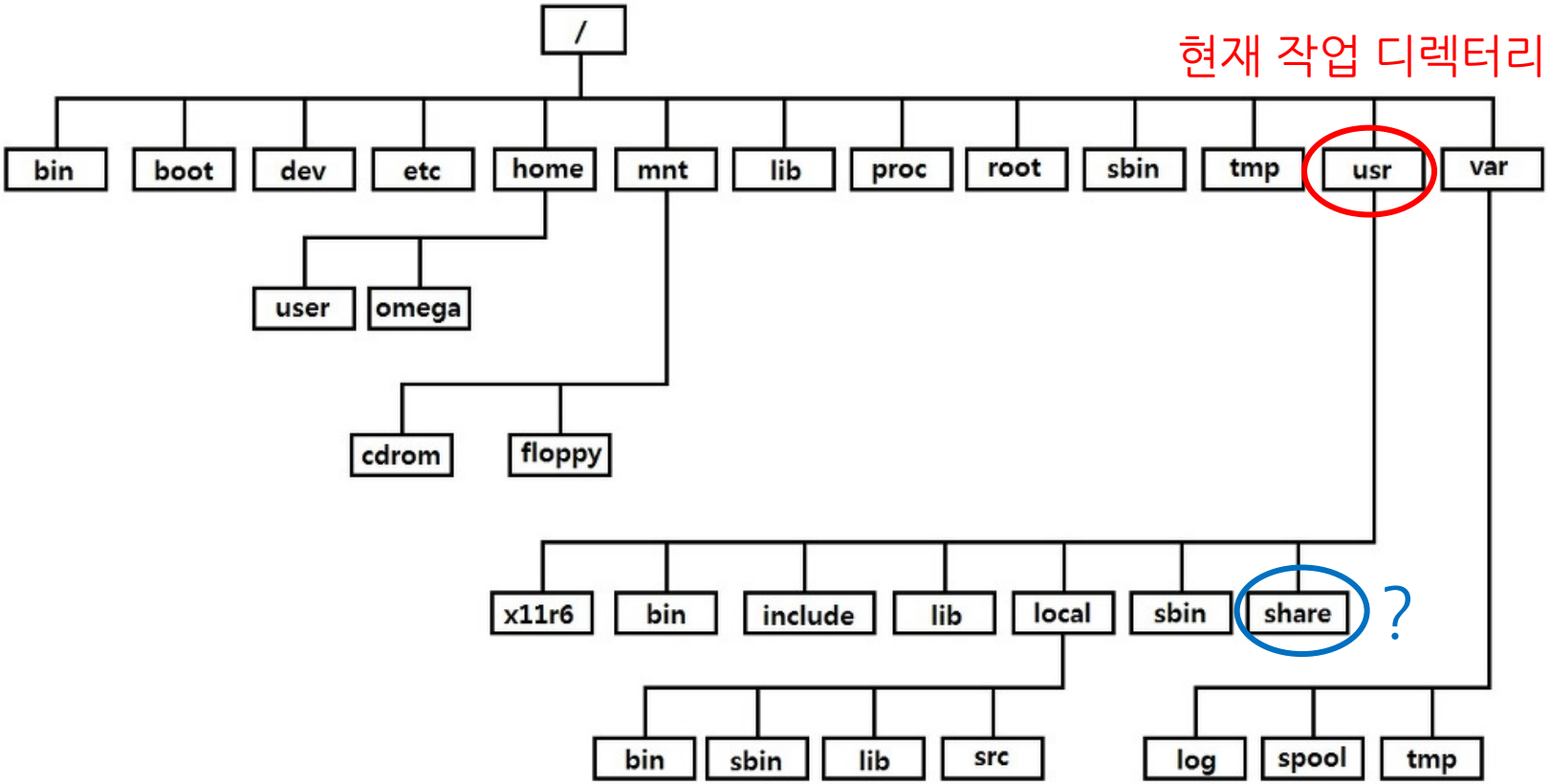
경로명 연습 (2)

- home 디렉터리
  - 절대 경로:
  - 상대 경로:



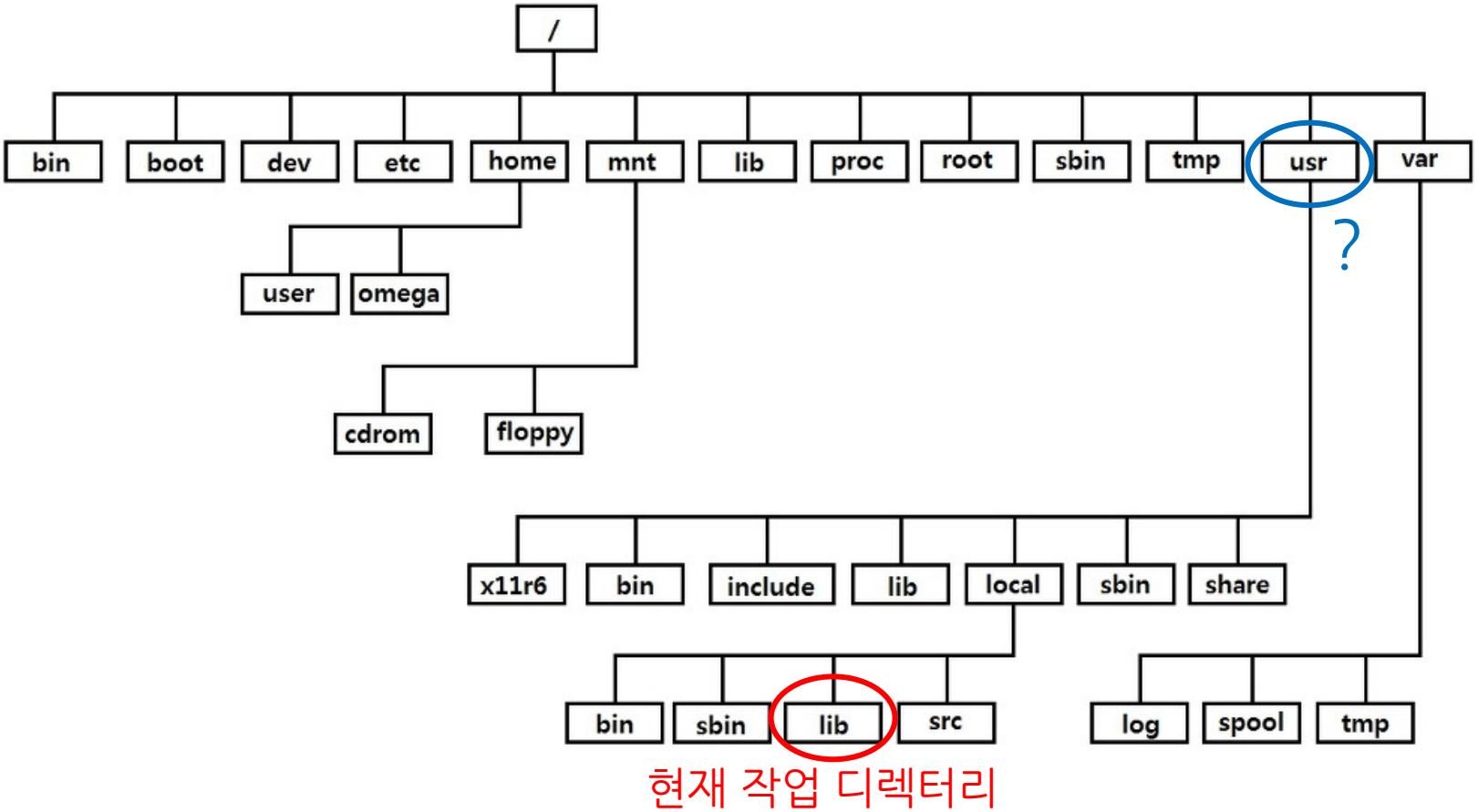
경로명 연습 (3)

- share 디렉터리
  - 절대 경로:
  - 상대 경로:



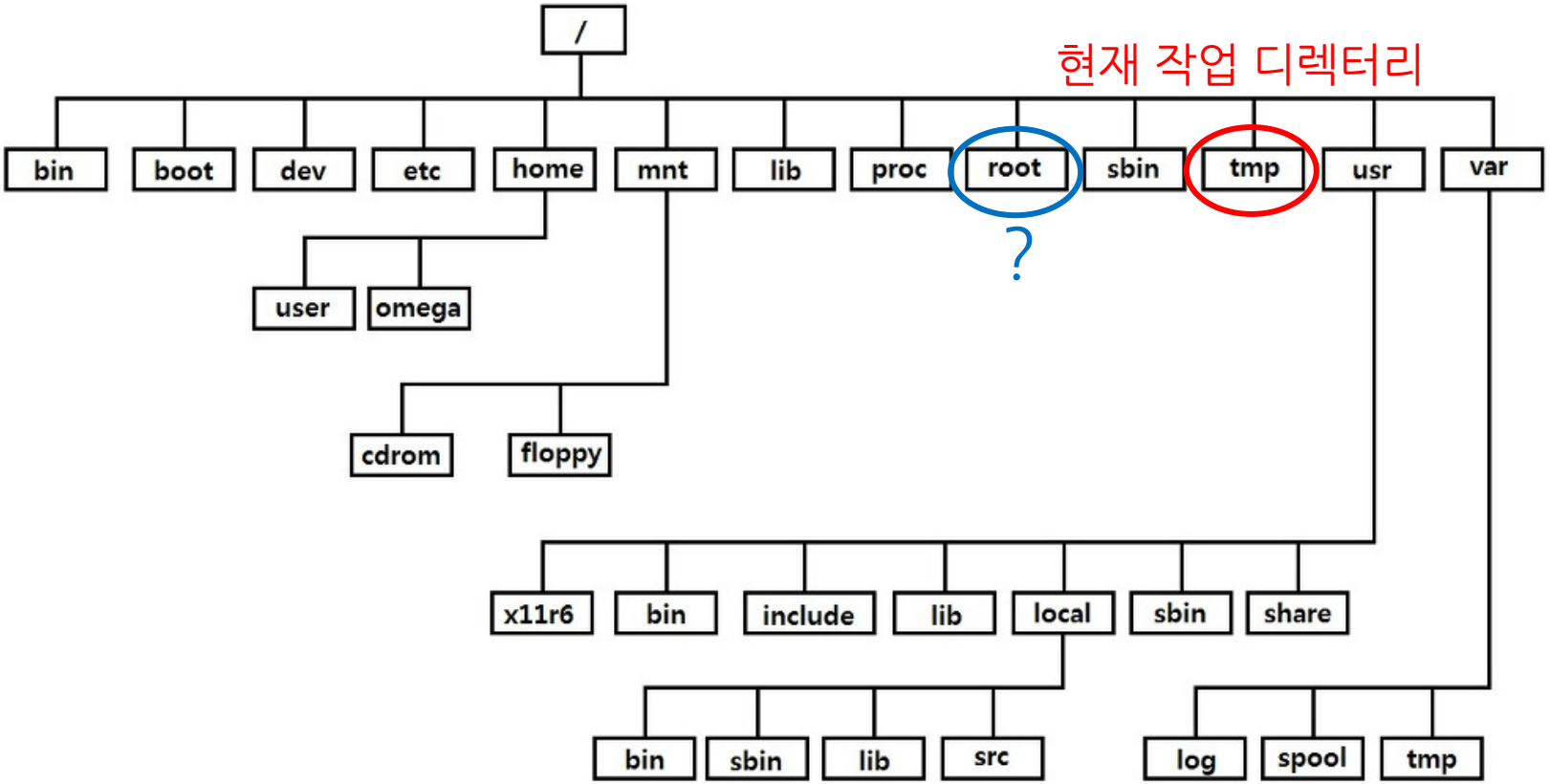
경로명 연습 (4)

- usr 디렉터리
  - 절대 경로:
  - 상대 경로:



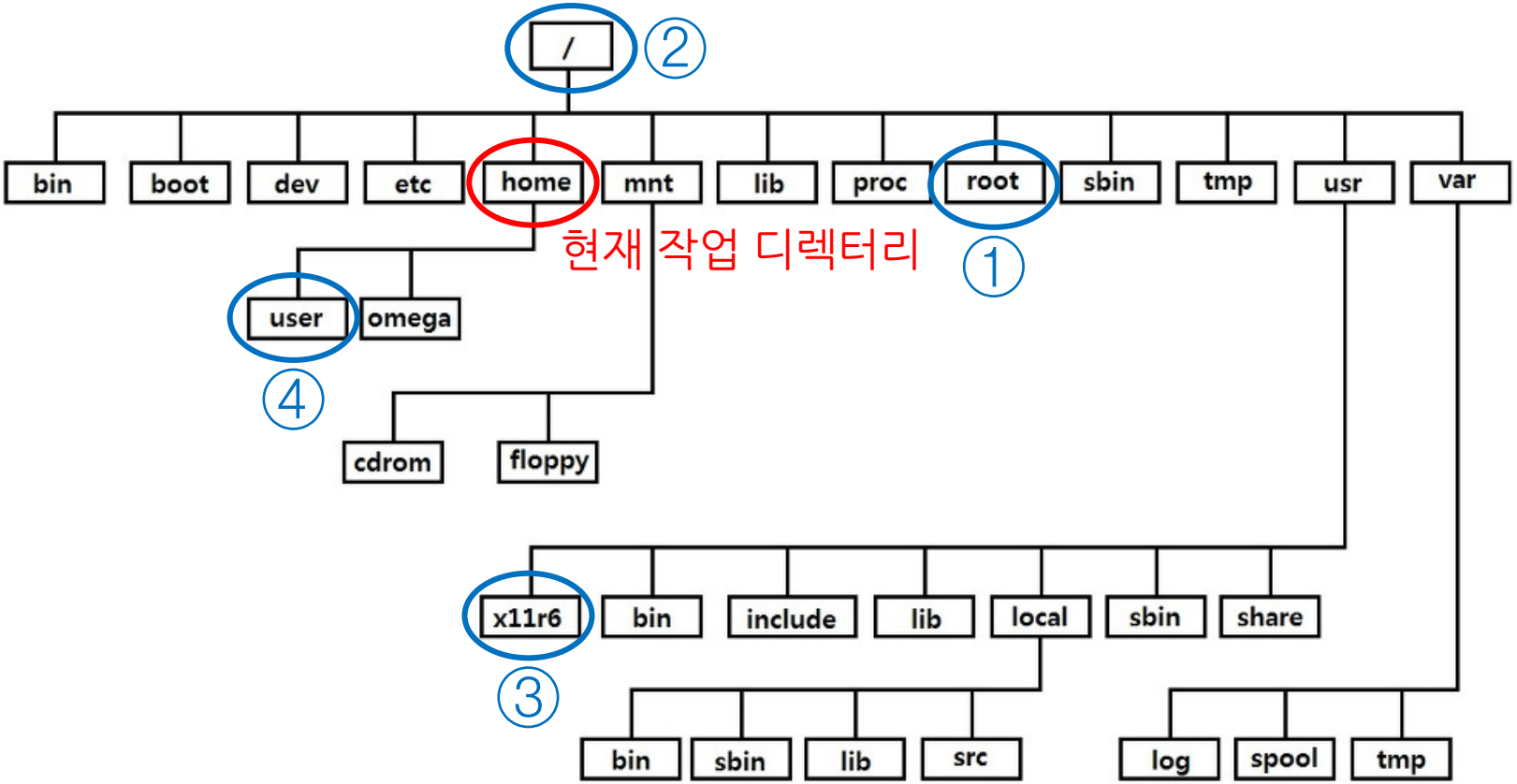
경로명 연습 (5)

- root 디렉터리
  - 절대 경로:
  - 상대 경로:



경로명 연습 (6)

- ①~④에 대한 절대 경로와 상대 경로를 각각 답하라.



디렉터리 내용 리스트

- ls (list)
  - 디렉터리의 내용을 리스트함.
- $ ls // 파일명만 출력함.
  cs1.txt
- $ ls -s // -s(size), 파일의 블록 크기를 출력함.
  총 6
  6 cs1.txt
- $ ls -a // -a(all), 숨김 파일까지 출력함.
  . .. cs1.txt

. 현재 디렉터리
.. 부모 디렉터리

- $ ls -l // -l(long), 파일의 상세 정보를 출력함.
  -rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
- $ ls -asl // 여러 옵션을 조합할 수 있음.
  총 10
  2 drwxr-xr-x 2 chang faculty 512 4월 16일 13:37 .
  2 drwxr-xr-x 3 chang faculty 512 4월 16일 13:37 ..
  6 -rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt

ls 명령어의 명령줄 인수

- 명령어의 우측에 적는 값을 명령줄 인수라고 함.
- ls [옵션] [*디렉터리 또는 파일들*]
- ls 명령어는 명령줄 인수로 받은 디렉터리들의 내용을 리스트함.
  - $ ls /tmp
  - $ ls ..
- 명령줄 인수가 없으면 현재 디렉터리의 내용을 리스트함.
  - $ ls
- 명령줄 인수가 파일들이면 그 파일들만 리스트함
  - $ ls cs1.txt
  - $ ls cs1.txt cs2.txt

디렉터리 관련 명령어

(p.71)

명령어	의미
ls	파일 및 디렉터리 리스트
ls -a	모든 파일과 디렉터리 리스트
ls -asl	모든 파일 자세히 리스트
mkdir	디렉터리 만들기
cd 디렉터리	디렉터리로 이동
cd	홈 디렉터리로 이동
cd ~	홈 디렉터리로 이동
cd ..	부모 디렉터리로 이동
pwd	현재 작업 디렉터리 프린트

실습

- p.100 실습문제
  - 2.1을 해봅시다.

파일 내용 출력

- 파일 내용 출력과 관련된 명령어들
  - cat, more, head, tail, wc 등
- 사용법
  - 명령어 [*파일들*]
  - cat cs1.txt // cs1.txt 파일 내용을 출력함.
  - cat // 단독으로 사용하면 키보드로 입력한 내용을 그대로 출력함. (more만 단독으로 사용 불가)

- 실습을 위한 예제 파일 만들기

```
$ cat > cs1.txt
Hello, World!
Stay hungry, stay foolish.
^D
```

^D : 정상 종료
^C : 강제 종료
^Z : 프로그램 정지후, 후면으로 보냄.

cat 명령어

- 파일 내용을 그대로 화면에 출력함.

```
$ cat cs1.txt

$ cat                    // 명령줄 인수로 파일을 지정하지 않으면
...                         키보드로 입력한 내용을 그대로 출력함.
^D
```

more / head / tail 명령어

- more 명령어
  - 하나 이상의 파일 이름을 받을 수 있으며 각 파일의 내용을 페이지 단위로 출력함.
  - 스페이스 바로 다음 페이지 이동, q로 종료함.
- head 명령어
  - 파일의 앞부분(10줄)을 출력한다.
  - head -5 cs1.txt // 보여줄 앞부분의 줄 수(5)를 지정할 수 도 있음.
- tail 명령어
  - 파일의 뒷부분(10줄)을 출력함.
  - tail -5 cs1.txt // 보여줄 뒷부분의 줄 수(5)를 지정할 수 도 있음.

wc 명령어

- wc 명령어 (word count)
  - 파일에 저장된 줄, 단어, 문자의 개수를 세서 출력함.

```
$ wc cs1.txt
38 318 2088 cs1.txt          // 38줄, 318단어, 2088 문자
```

  - 옵션을 사용하면 줄 수(-l), 단어 수(-w), 문자 수(-c)를 선택해서 출력할 수 있음.

```
$ wc -l cs1.txt
38 cs1.txt                   // 줄 수만 출력됨.
$ wc -w cs1.txt
318 cs1.txt                  // 단어 수만 출력됨.
$ wc -c cs1.txt
2088 cs1.txt                 // 문자 수만 출력됨.
```

more / head / tail / wc 연습

- 실습을 위한 예제 파일 만들기

  $ man ls > cju.txt // cju.txt 파일이 만들어짐.

- 실습

```
$ ls
$ cat cju.txt
$ more cju.txt                    // 스페이스 바로 이동, q로 종료
$ head cju.txt
$ head -3 cju.txt
$ tail cju.txt
$ tail -5 cju.txt
$ wc cju.txt
$ wc -l cju.txt
$ wc -w cju.txt
$ wc -c cju.txt
```

파일 및 디렉터리 조작 명령어

- cp 명령어 (copy)
- $ cp 파일1 파일2
  - 파일1을 복사하여 파일2를 현재 디렉터리 내에 새로 만듦.

```
$ cp cs1.txt cs2.txt
$ ls -l cs1.txt cs2.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:45 cs2.txt
```

- $ cp 파일 디렉터리
  - 파일의 복사본을 디렉터리 내에 만듦.

```
$ cp cs1.txt /tmp
$ ls /tmp/cs1.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 14:31 cs1.txt
```

mv 명령어

- mv (move)
- $ mv 파일1 파일2
  - 파일1을 파일2로 이동시킴.

```
$ mv cs2.txt cs3.txt          // cs2.txt 파일은 더 이상 존재하지 않게 됨.
$ ls -l
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 13:56 cs3.txt
```

- $ mv 파일 디렉터리
  - 파일을 디렉터리 내로 이동

```
$ mv cs3.txt /tmp
$ ls /tmp/cs3.txt
-rw-r--r-- 1 chang faculty 2088 4월 16일 14:56 cs3.txt
```

파일 및 디렉터리 삭제 명령어

- rm 명령어 (remove)
  - 명령줄 인수로 받은 파일들을 삭제함.

```
$ rm 파일1 [파일2]
$ rm cs1.txt                    // cs1.txt를 삭제함.
$ rm cs1.txt cju.txt            // cs1.txt와 cju.txt를 삭제함.
```

- rmdir 명령어 (remove directory)
  - 명령줄 인수로 받은 디렉터리들을 삭제함.
  - 단, 디렉터리 내에 아무 것도 없어야 삭제 가능함.

```
$ rmdir test
$ rmdir test1 test2
```

- 디렉터리 내의 모든 것을 한번에 지우려면

```
$ rm -r 디렉터리                // rmdir이 아닌 rm 명령어를 사용함.
```

링크

- 링크는 기존 파일에 대한 또 하나의 새로운 이름이라고 할 수 있음.
- 하드 링크
  - $ echo hi > hello.txt // hello.txt 파일 생성
  - $ ls -l
  - $ ln hello.txt hi.txt // 하드 링크 생성
  - $ ls -l
- 심볼릭 링크
  - $ ln -s hello.txt hey.txt // 심볼릭 링크 생성
  - $ ls -l

파일 관련 명령어

(p.83)

명령어	의미
cat 파일*	파일 디스플레이
more 파일+	한 번에 한 페이지씩 디스플레이
head 파일*	파일의 앞부분 디스플레이
tail 파일*	파일의 뒷부분 디스플레이
wc 파일*	줄/단어/문자 수 세기
cp 파일1 파일2	파일1을 파일2로 복사
mv 파일1 파일2	파일1을 파일2로 이름 변경
rm 파일+	파일 삭제
rmdir 디렉터리+	디렉터리 삭제

ln 파일1 파일2 링크 만들기

실습

- p.100 실습문제
  - 2.2를 해봅시다.
- p.101 실습문제
  - 2.3을 해봅시다.
  - 2.4를 해봅시다.

파일 속성

- 파일은 이름, 타입, 크기, 소유자, 사용권한, 수정 시간 등의 파일 속성을 갖음.

  $ ls -sl cs1.txt

  6 -rw-r--r-- 1 chang faculty 2088 4월 16일 13:37 cs1.txt

파일 속성	의미
블록 수	파일이 저장된 블록의 수 6
파일 타입	일반 파일(-), 디렉터리(d), 링크(l), 파이프(p), 소켓(s), 디바이스(b 혹은 c) 등의 파일 종류를 나타낸다. -
사용권한	소유자, 그룹, 기타 사용자의 파일에 대한 읽기/쓰기/실행 권한 rw-r--r--
소유자 및 그룹	파일의 소유자 및 소유자가 속한 그룹 chang faculty
크기	파일의 크기(파일이 가지고 있는 데이터의 크기) 2088
수정 시간	파일을 최후로 생성 혹은 수정한 시간 4월 16일 13:37

파일의 사용권한(permission)

- 시스템 관리자는 보안 관리를 위해 동일한 성격의 사용자들을 하나의 그룹으로 묶어서 관리함.
  - 사용자는 하나 이상의 그룹에 속함.
- 파일의 사용권한에는 읽기(r), 쓰기(w), 실행(x) 권한이 있음.

권한	파일	디렉터리
r	파일에 대한 읽기 권한	디렉터리 내에 있는 파일명을 읽을 수 있는 권한
w	파일에 대한 쓰기 권한	디렉터리 내에 파일을 생성하거나 삭제할 수 있는 권한
x	파일에 대한 실행 권한	디렉터리 내로 탐색을 위해 이동할 수 있는 권한

- 파일의 보안을 위해 파일의 소유자(owner)/그룹(group)/기타(others)로 구분하여 사용권한을 관리함.
- 파일의 사용권한이 rw-r--r-- 이면,
  - 소유자(rw-), 그룹(r--), 기타(r--)

chmod 명령어 (change mode)

- 파일 혹은 디렉터리의 사용권한을 변경하는 명령어

  $ chmod 사용권한 파일

  $ chmod [-R] 사용권한 디렉터리

  - -R 옵션은 디렉터리 내의 모든 파일, 하위 디렉터리에 대해서도 사용권한 변경이 적용됨.

• **8진수를 이용하는 방법**
사용권한 rw- rw- r--
2진수: 110 110 100
8진수: 6 6 4
$ chmod 664 cs1.txt
 	
$ chmod 400 cs1.txt
결과는?
	• **기호를 이용하는 방법**
	[u|g|o|a]$^{+}$[+|-|=][r|w|x]$^{+}$
	u(user), g(group), o(other), a(all)
	연산자: +(추가), -(제거), =(지정)
	권한: r(읽기), w(쓰기), x(실행)
	$ chmod g+w cs1.txt
	$ chmod o-r cs1.txt
	$ chmod go=rw cs1.txt

실습

- p.102 실습문제
  - 2.5를 해봅시다.

퀴즈

- cju.avi 파일의 사용권한이 다음과 같다. 사용권한의 의미를 설명하시오.

  rw-------

- cju.avi 파일의 사용권한을 rw-r--r--로 바꾸고자 한다. 8진수를 이용하는 방법의 올바른 명령어 사용방법을 답하시오.

  $

- 다음은 기호를 이용한 사용권한 변경 방법이다. cju.avi 파일의 사용권한은 어떻게 변경되는가?

  $ chmod a=r cju.avi

chown/chgrp 명령어

- chown 명령어 (change owner)
  - 파일이나 디렉터리의 소유자를 변경함.
  - 시스템 관리자만 사용 가능함.

  $ chown 사용자 파일
  $ chown [-R] 사용자 디렉터리
  $ chown cju2 cs1.txt // cs1.txt의 사용자를 cju2로 변경함.
- chgrp 명령어 (change group)
  - 파일이나 디렉터리의 그룹을 변경함.
  - 파일 소유자는 자신이 속한 그룹으로만 그룹을 변경할 수 있음.
  - 시스템 관리자는 자유롭게 그룹을 변경할 수 있음.

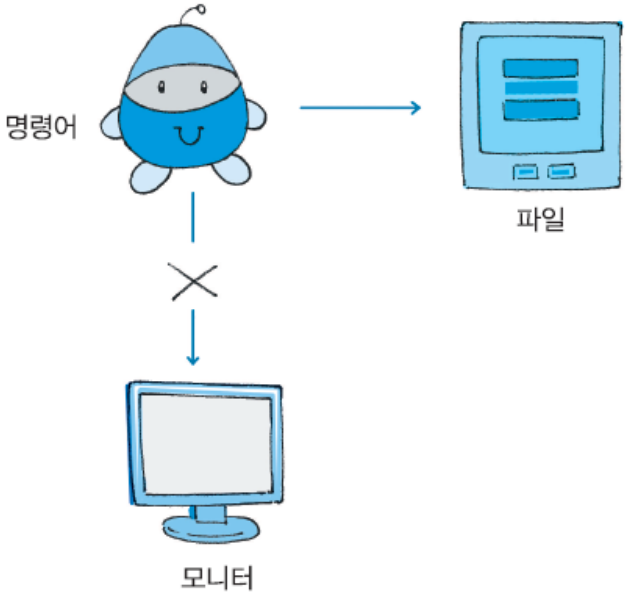
  $ chgrp 그룹 파일
  $ chgrp [-R] 그룹 디렉터리
  $ chgrp student cs1.txt // cs1.txt의 그룹을 student로 변경함.

출력 재지정

- 출력 재지정(output redirection)을 이용하면 명령어의 출력을 모니터에 출력하는 대신에 파일에 저장함.
  - 출력 재지정 기호 (>)
  - 사용법

    $ 명령어 > 파일

    $ who > names.txt



출력 재지정 예제

- cat 명령어와 출력 재지정을 이용한 간단한 파일 생성
  - cat 명령어는 명령줄 인수가 없으면 키보드로부터 입력 받은 내용을 모니터에 출력함.
  - 따라서 출력 재지정을 이용하면 모니터가 아닌 파일에 출력할 수 있음.

$ cat > list1.txt // 출력 재지정 Hi ! This is the first list. ^D	$ cat > list2.txt // 출력 재지정 Hello ! This is the second list. ^D

```
$ cat list1.txt                          // 파일 출력
$ cat list2.txt                          // 파일 출력
$ cat list1.txt list2.txt                // 두 파일 함께 출력
$ cat list1.txt list2.txt > list3.txt    // 출력 재지정
$ cat list3.txt                          // 파일 출력
```

출력 추가

- 출력 추가(output append)를 이용하면 명령어의 출력을 모니터 대신에 기존 파일의 뒤에 추가함.
  - 출력 추가 기호 (>>)
  - 사용법

    $ 명령어 >> 파일
  - 예

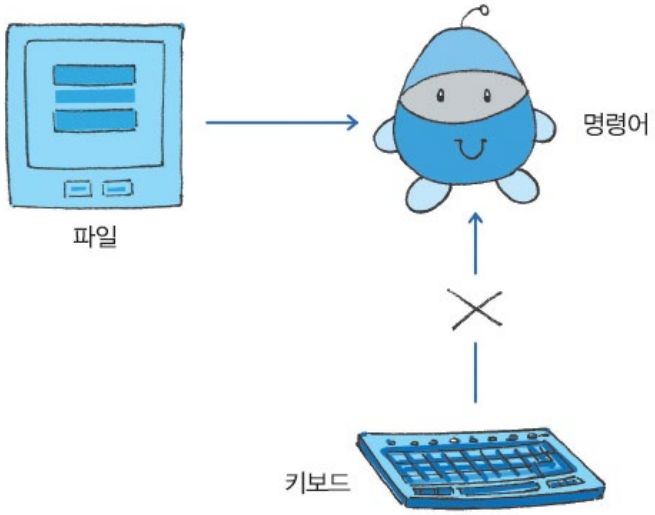
```
$ cat >> list1.txt              // 출력 추가
Bye !
This is the end of the first list.
^D
```

```
$ cat list1.txt                 // 출력 추가
Hi !                            } 기존 파일 내용
This is the first list.
Bye !                           } 추가된 내용
This is the end of the first list.
```

입력 재지정

- 입력 재지정(input redirection)을 이용하면 명령어의 입력을 키보드에서 받는 대신에 파일에서 받음.
  - 입력 재지정 기호 (<)
  - 사용법

```
$ 명령어 < 파일
$ wc < list1.txt
4 17 71
```



문서 내 입력

- 문서 내 입력(here document)이란 명령어를 실행할 때 문서(보통 스크립트) 내에서 입력을 받을 수 있는 기능을 말함.
  - 문서 내 입력 기호 (<<)
  - 사용법

```
$ 명령어 << 단어          // << 기호 뒤의 단어가
...                       // 다시 나타날 때까지의 내용을 입력 받음.
단어
```

  - 예

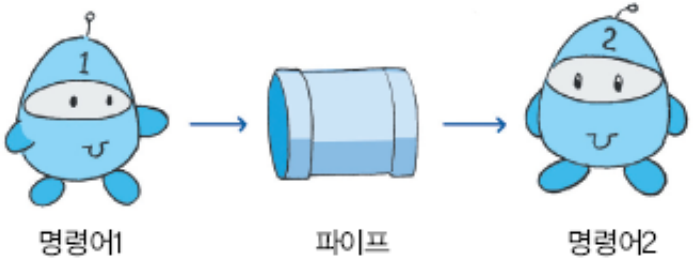
```
$ wc << end
> hello !
> word count
> end              // end가 나왔으므로 입력 받기를 종료함.
2 4 19             // end가 나오기 전까지의 2줄에 대한 결과
```

파이프

- 파이프의 필요성
  - 로그인 된 사용자들을 정렬해서 보여주기
    - 입력, 출력 재지정을 이용하면 가능하긴 하지만 상당히 번거로움.

```
$ who > names.txt          // 출력 재지정
$ sort < names.txt         // 입력 재지정
```

- 파이프(pipe)
  - 파이프를 이용하면 한 명령어의 출력을 다른 명령어의 입력으로 바로 전달할 수 있음.
  - 파이프 기호 (|)
  - 사용법
    - $ 명령어1 | 명령어2
  - 예
    - $ who | sort // 로그인 된 사용자를 정렬해서 보여줌.
    - $ who | wc -l // 로그인 된 사용자 수를 출력함.



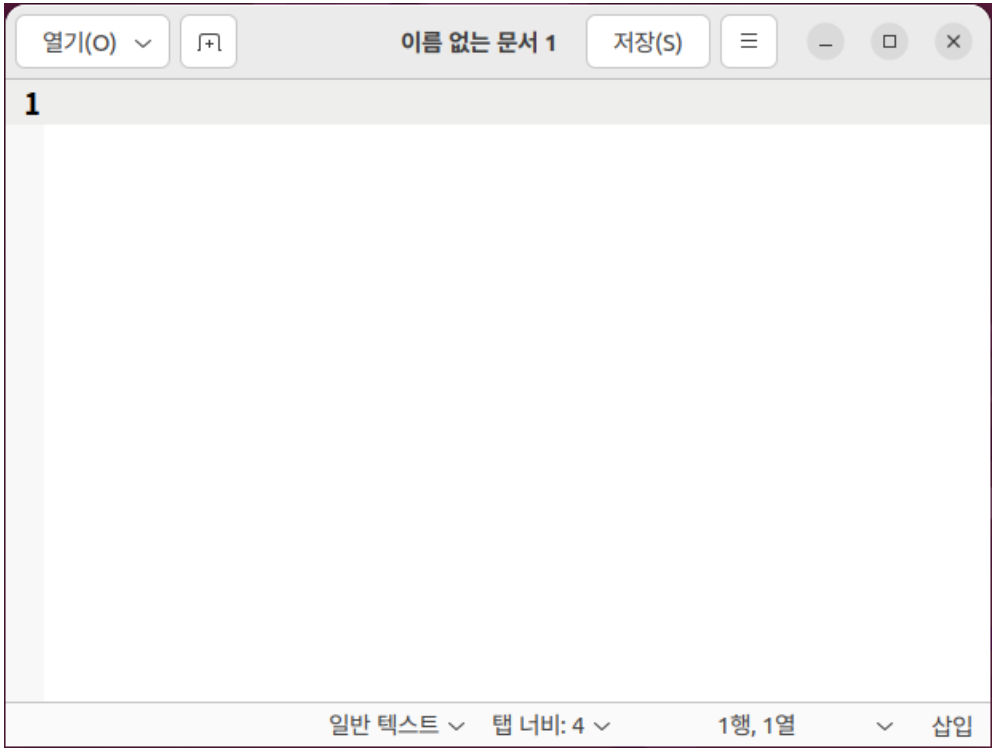
입출력 재지정 관련 명령어

(p.95)

명령어	의미
명령어 > 파일	표준 출력을 파일로 출력 재지정
명령어 >> 파일	표준 출력을 파일에 출력 추가
명령어 < 파일	표준 입력을 파일로 입력 재지정
명령어1 | 명령어 2	명령어1의 표준출력이 파이프를 통해 명령어2의 표준입력이 됨.
cat 파일1 파일2 > 파일3	파일1과 파일2를 연결하여 파일 3을 새로 만듦.

텍스트 편집기

- CUI 기반 텍스트 편집기
  - vi // Vi(Vim) Editor
  - nano // Nano Editor
- GUI 기반 텍스트 편집기
  - gedit // Gedit



질문

Q&A