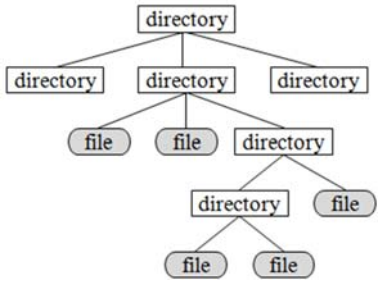
3. 파일시스템 사용

3.1 계층적 파일 시스템

- 파일(file)
  - 디스크에 저장되는 자료들의 모음
  - **파일이름**을 사용하여 자료들을 간편하게 다룸
- 계층적 파일 시스템
  - 디렉토리(directory) – 포함하고 있는 파일 또는 디렉토리 이름과 관련 정보 보관 (cf) 폴더(folder)



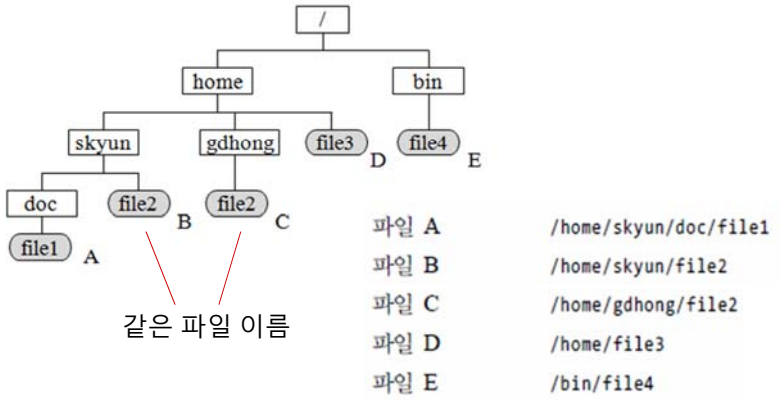
파일 유형과 파일 이름

- 파일 유형
  - 일반 파일(regular file)
  - 디렉토리 파일
  - 특수 파일 – 입출력 장치 정보 보관, 장치파일(device file)
- 파일 이름
  - / 를 제외한 모든 출력가능 문자 사용 가능
  - shell에서 특수한 용도로 사용하는 문자들을 사용하지 않을 것을 권장

    `< > ( ) [ ] { } * ? " ' - $ ^`
  - 다음 문자들 중에서 선택 사용 권장 (단, .과 ..은 사용할 수 없음)
    - 알파벳 문자(A-Z, a-z)
    - 숫자(0-9)
    - 밑줄문자(_), 마침표(.)
  - 대소문자를 구분함
  - **확장자**: 마지막 마침표 뒤의 문자열. 파일의 종류를 구분하는 용도
    - UNIX/Linux는 파일종류에 따라서 특정 확장자를 요구하지 않음
    - 특정 확장자는 대개 응용 프로그램이 요구하는 것임

경로 이름

- 루트(root) 디렉토리
  - 계층적 파일 시스템의 꼭대기 디렉토리
- 절대 경로 이름 : / (root)에 상대적인 경로, /으로 디렉토리 구분



작업 디렉토리/홈 디렉토리와 상대 경로 이름

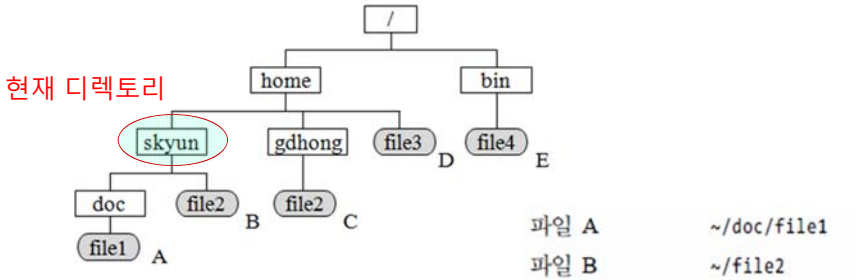
- 작업 디렉토리(working directory)
  - 프로세스(수행 중인 프로그램)가 현재 위치한 디렉토리
    - 작업 디렉토리의 파일은 파일이름만 사용하여 접근가능
  - 현재 작업 디렉토리, 현재 디렉토리라고도 함
- 상대 경로 이름
  - 작업 디렉토리에 상대적인 위치로 파일 이름을 표기
- 홈 디렉토리
  - 특정 사용자에게 부여된 디렉토리. 이 디렉토리 아래로 계층적으로
  - 현재 사용자나 특정 사용자의 홈디렉토리에 상대적 위치로 표기가능

.	(현재) 작업 디렉토리
..	부모 디렉토리
~	현재 사용자의 home 디렉토리
~gdhong	사용자 gdhong의 home 디렉토리

작업 디렉토리/홈 디렉토리와 상대 경로 이름

- 작업 디렉토리(working directory)
  - 프로세스(수행 중인 프로그램)가 현재 위치한 디렉토리
    - 작업 디렉토리의 파일은 파일이름만 사용하여 접근가능
  - 현재 작업 디렉토리, 현재 디렉토리라고도 함
- 상대 경로 이름
  - 작업 디렉토리에 상대적인 위치로 파일 이름을 표기
- 홈 디렉토리
  - 특정 사용자에게 부여된 디렉토리.
  - 로그인 시에 기본 작업 디렉토리로 사용됨
  - 현재 사용자나 특정 사용자의 홈디렉토리에 상대적 위치로 파일 이름 표기가능

.	(현재) 작업 디렉토리
..	부모 디렉토리
~	현재 사용자의 home 디렉토리
~gdhong	사용자 gdhong의 home 디렉토리

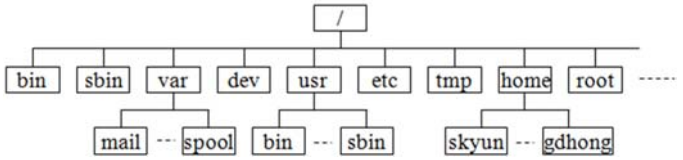


파일 A ~/doc/file1
파일 B ~/file2
파일 C ~gdhong/file2

파일 A doc/file1 또는 ./doc/file1
파일 B file2 또는 ./file2
파일 C ../gdhong/file2
파일 D ../file3
파일 E ../../bin/file4

전체 디렉토리 구조

- 리눅스 파일 시스템 계층 구조
  - 파일 시스템 계층구조 표준(FHS) 기반



  - 각 디렉토리의 용도
    - 교과서 34쪽 참조

3.2 파일과 디렉토리 관련 유틸리티

- 표준 입출력(standard input/output)
  - 표준입력(stdin): keyboard
  - 표준출력(stdout): 화면
  - 표준에러출력(stderr): 화면
- cat을 사용한 파일 생성

  $ cat ; 표준입력(stdin)을 화면(stdout)에 출력

  $ cat > lyrics ; 표준입력(stdin)을 화면 대신에 파일 heart로 출력을 저장 → redirection(5장)
  - [Enter]를 입력해야 입력이 프로그램에 전달됨
  - ^D : keyboard 입력 끝
- ls - 디렉토리 목록 보기 (list)

  $ ls … 현재 디렉토리 목록

  $ ls /usr … /usr 디렉토리 목록

  $ ls xyz … 파일 존재여부 확인에 사용 가능

ls – 디렉토리 목록 보기

- ls 명령어 옵션

  -l ; long format

  -a ; all (hidden file포함)

  (cf) **hidden file** : 파일 이름이 . 으로 시작하는 파일

  -F ; file 유형

  -s ; KB 단위 크기

  -t : 수정 시간 순서(time)

  -R ; recursive (subdirectory 내용 포함)

  -d ; 디렉토리에 대해서 디렉토리 이름만 출력

  -m ; 파일 목록을 컴마로 구분

  -R : recursive 출력
  - 옵션의 혼합 사용 가능

```
$ ls -la
합계 40
drwx------    4 gdhong student 4096 2015-07-26 13:27 .
drwxr-xr-x. 163 root   root    4096 2015-07-26 13:14 ..
```


파일 내용 보기

- cat – 파일 내용 보기 (concatenate)

  $ cat lyrics

  $ cat lyrics /etc/resolv.conf ; 여러 파일 내용 출력(연결)

  $ cat –n lyrics ; 줄 번호 출력
- more, less – 화면 단위로 파일 내용 보기
  - 긴 텍스트 파일의 내용 보기에 적합

    $ more /etc/protocols

    $ more +20 /etc/protocols
  - more 내에서의 명령 키 – 교과서 38쪽 참조

    space – 다음 페이지 enter – 다음 줄

    b – 이전 페이지 h - 도움말

    q, ^C – 종료 (마지막 도달시 자동 종료)

    …

파일 내용 일부 보기

- head – 파일 앞부분 보기

  $ head file ; 처음 10줄 표시

  $ head -2 file ; 처음 2줄 표시
- tail – 파일의 뒷부분 보기

  $ tail file ; 마지막 10줄 표시

  $ tail -20 file ; 마지막 20줄 표시

  $ tail -f file ; 다른 프로그램에 의해서 추가되는 내용을 계속하여 출력 (로그 파일을 계속적으로 읽는 데 주로 사용)

디렉토리 관련 명령어

- mkdir - 디렉토리 생성 (make directory)

```
$ mkdir newdir
$ mkdir -p dir1/dir2        ; 존재하지 않는 부모디렉토리도 생성
```

- cd - 디렉토리 이동 (change directory)

```
$ cd dir                    ; 자식 디렉토리
$ cd /usr/bin               ; 절대경로 사용
$ cd ../doc                 ; 상대경로 사용
```

- rmdir - 디렉토리 삭제 (remove directory)

```
$ rmdir olddir              ; 빈 directory이어야 함
                            (cf) rm -r
```


파일 관련 명령어

- cp - 파일 복사 (copy)

```
$ cp file1 file2            ; file1을 file2로 복사
$ cp file1 dir              ; file1을 디렉토리 dir로 복사
$ cp -i file1 file2         ; 확인 기능
$ cp -r dir1 dir2           ; 디렉토리의 recursive 복사
                              (subdirecty모두 복사)
```

- mv - 파일 이름 변경 (move)

```
$ mv file1 file2            ; 파일이름 변경
$ mv dir1 dir2              ; 디렉토리 이름 변경
$ mv file1 dir              ; 파일을 dir 디렉토리로 이동
$ mv -i file1 file2         ; 확인 기능
```

- rm - 파일 삭제 (remove)

```
$ rm file1 file2 ...
$ rm -i file                ; 확인 기능
$ rm -r dir                 ; 디렉토리의 recursive 삭제
                              (비어있지 않아도 됨)
```


단어 수 세기

- wc – 파일의 단어/행/문자 수 출력

```
$ wc file                   ; line수, word수, char수 출력
$ wc -l file                ; line수 출력
$ wc -wc file               ; word, char수 출력
```


3.3 파일 속성 및 관련 명령어

- 파일 속성

```
$ ls -l
합계 8
-rw-r--r-- 1 gdhong student  140 2015-07-26 13:27 lyrics
drwxr-xr-x 3 gdhong student 4096 2015-07-26 15:19 music
     ①    ②  ③     ④      ⑤         ⑥         ⑦
```

1) 파일 유형 및 허가권(read/write/execute)
2) 하드링크 수
3) 파일 소유자
4) 파일 그룹
5) 파일 크기(B)
6) 파일 수정시간
7) 파일 이름

기본 파일 속성

- 파일 수정 시간
  - 단순한 정보로도 의미가 있지만
  - make (10장), find (6장)와 같은 여러 유틸리티에서 사용함
- 파일 소유자/그룹
  - 파일 소유자 = 파일을 생성한 프로세스의 소유자
  - 파일 그룹 = 파일을 생성한 프로세스의 그룹
- 파일 유형 – 필드1의 첫째 문자

  drwxr-xr-x

  - : regular d : directory l : symbolic link
  b : block I/O c : char I/O p : pipe
  s : socket
  (교과서 47쪽 참조)

파일 허가권

- 파일 접근 권한(허가권)
  - 사용자들을 **소유자**(u), **그룹**(g), **나머지 사용자**(o)의 3 단계로 구분
  - 사용자단계마다 파일에 대한 **읽기(r), 쓰기(w), 실행(x)** 권한 별도 부여

user	group	other	; 3단계
rwx	rwx	rwx	; 의미
rw-	r--	r--	; -는 권한(허가권)이 없음을 의미

- 파일 유형에 따른 허가권 의미

허가권	일반 파일	디렉토리	특수 파일
읽기(r)	파일의 내용을 읽을 수 있음	디렉토리를 읽을 수 있음 (ls 수행 가능)	read()를 호출하여 장치에서 입력을 할 수 있음
쓰기(w)	파일의 내용을 변경할 수 있음	디렉토리 내용을 변경할 수 있음 (파일 생성 및 삭제 가능)	write()를 호출하여 장치로 출력을 할 수 있음
실행(x)	파일을 실행할 수 있음 (실행 파일)	디렉토리로 이동하여 디렉토리에 있는 파일에 접근할 수 있음 (cd 가능)	의미 없음

하드링크 수

- 하드 링크
  - 디스크에 저장된 파일에 대한 포인터
  - 같은 파일이 여러 경로 이름으로 하드 링크될 수 있다.
  - 즉, 같은 파일에 여러 경로 이름이 부여될 수 있다.
- 디렉토리의 하드링크 수 = 2 + (자식 디렉토리 수)
  - 디렉토리는 최소 2개의 하드링크(자신과 부모와 링크됨)를 가짐
  - 자식 디렉토리가 있으면 자식에서도 하드링크를 가짐

```
drwxr-xr-x 3 gdhong student 4096 2015-07-26 15:19 music
```


소유자, 그룹

- chown – 파일 소유자 변경
  - 관리자만 변경 가능

```
$ chown kim lyrics lyrics2  ; 파일 소유자를 kim으로 변경
$ chown -R kim music        ; 디렉토리 이하 모든 파일 소유자 재귀적 변경
```

- groups – 그룹 목록 보기

```
$ groups            ; 사용자가 속한 그룹 출력
```

- chgrp – 파일 그룹 변경 (둘 이상의 그룹에 속한 사용자에게 가능)

```
$ chgrp cte lyrics2
$ chgrp -R cte music
```

- newgrp – 셸의 그룹 변경 (둘 이상의 그룹에 속한 사용자)

```
$ .. 현재 셸이 기본 그룹에 속함
$ newgrp cte
$ .. 현재 셸이 cte 그룹에 속함 ; 새로운 셸이 생성됨
$ exit              ; 기본 그룹에 속한 원래 셸로 되돌아 감
```

자세한 파일 유형

- file - 자세한 파일 유형을 출력
  $ file *filename1 filename2 …*
  - ASCII text
  - UTF-8 Unicode text
  - directory
  - ELF 64-bit LSB executable, x86-64 …
  - symbolic link to …
  - …

chmod – 파일 허가권 변경

- 형식: chmod [-R] mode file ...
  - mode
    기호 형식: [u, g, o, a] [+ = -] [r, w, x, s]
    (ex) g+w, u-rw, a+x, +x (= u+x)
    8진수 형식: rwxr-xr-x → 111 101 101 = 0755

$ chmod g+w file ; group에 write허가 (파일허가권 수정)

$ chmod 754 file ; rwxr-xr– (파일허가권 지정)
$ chmod a=rwx,g=rx,o=r file ;

프로세스의 소유자/그룹과 파일 허가권

- 프로세스의 소유자 및 그룹
  로그인 **사용자**가 실행파일을 실행시켜서 **프로세스**를 생성할 때
  - 프로세스의 소유자 = 로그인 사용자
  - 프로세스의 그룹 = 로그인 사용자의 현재 그룹
- 프로세스가 생성한 파일의 소유자 및 그룹
  - 파일을 생성한 프로세스의 소유자와 그룹이 부여됨
- 파일 허가권 규칙
  - if (프로세스의 소유자 = 파일 소유자) → 소유자 허가권
  - else if (프로세스의 그룹 = 파일 그룹) → 그룹 허가권
  - else → 나머지 사용자 허가권

setuid, setgid 허가권

- 공유 파일 사용 문제
  - (프로세스 사용자/그룹) 을 (로그인 사용자/그룹) 으로 부여하는 방법은 여러 사용자가 쓰기를 해야 하는 공유 파일 사용에 부적합
    (ex) passwd 명령어 → 공유 패스워드 파일 사용
  - 해결책 : set user id, set group id 라는 특수한 파일 허가권 사용
- setuid (set user id)과 setgid (set group id) 허가권
  - 다음 허가권이 부여된 실행 파일이 실행될 때에 생성된 프로세스의 소유자와 그룹은 다음과 같이 지정됨
    - setuid 권한 → 프로세스 소유자 = 실행파일 소유자
    - setgid 권한 → 프로세스 그룹 = 실행파일 그룹
  - ls –l의 파일 허가권 출력에 x대신 s로 표시 (실행허가권이 없으면 S로)
    (ex) rw**s** r-x r-x ; setuid
    rw**s** r-**s** r-x ; setuid, setgid
  - 이 허가권을 가진 실행파일을 대개 상승된 권한을 가지고 프로그램을 실행함 → 보안에 문제가 있을 경우에는 시스템 전체 보안에 문제가 될 수 있으므로 보안에 유의하여 프로그램을 작성해야 함.

sticky 허가권

- sticky 허가권
  - 디렉토리에 부여되는 일반 사용자를 위한 특별한 허가권
  - 디렉토리에 일반 사용자에게 쓰기 허가권이 부여되지만,
    파일 삭제, 파일 이름 변경은 파일 소유자, 디렉토리 소유자와 관리자만 가능함
    (예) /tmp 디렉토리 (임시 파일 저장용)
  - ls –l의 파일 허가권 출력에 x대신 t로 표시 (실행허가권이 없으면 T로)
    drwxrwxrw**t** … /tmp

setuid, setgid, sticky 허가권 지정 - chmod

- setuid 허가권 지정
  $ chmod u+s exefile
- setgid 허가권 지정
  $ chmod g+s exefile
- sticky 허가권 지정
  $ chmod o+t directory
- 8진수를 사용한 허가권 지정
  - 기존 9비트 앞에 3비트 추가하여 setuid, setgid, sticky 허가권을 표
    - 4000: set-user-id
    - 2000: set-group-id
    - 1000: sticky

  $ chmod 4755 file ; rws**r**-s**r**-x
  $ chmod 6755 file ; rws**r**ws**r**-x
  $ chmod 1777 dir ; rwxrwxrw**t**

umask – 기본 파일 허가권 지정

- umask – 프로세스가 생성하는 파일 허가권 미리 지정
  $ umask … 현재의 umask 값 출력
  0022 ; 000 000 010 010 … 1에 대응되는 허가권 제한
  (그룹과 일반사용자 쓰기 허가권 제한)
  $ umask 002 … umask 값 지정
  ; 000 000 010 … 일반사용자 쓰기허가권 제한
- umask와 chmod
  - 파일 허가권 부여 대상을 지정하지 않으면 umask값으로 대상이 지정됨

  $ chmod +w file
    - umask=022 이면 소유자에게만 쓰기 허가권 부여
    - umask=002 이면 소유자, 그룹에게 쓰기 허가권 부여

접근 제어 목록(ACL)

- 파일의 3단계 접근 제어
  - 전통적 유닉스/리눅스 허가권은 3단계 파일 접근 제어를 사용
- 접근 제어 목록(Access Control List: ACL)
  - 특정 사용자에 대한 미세한 파일 접근 제어를 위해 ACL 사용 가능
  - ACL을 사용하기 위해서는 ACL 기능이 추가로 설치되어 있어야 하며, 일부 파일 시스템에서만 이 기능 사용 가능

3.4 파일 링크

- ln 명령어 – 링크 생성
  - 기존 파일에 새로운 이름을 추가로 부여하는 것
  - 다른 파일이름을 추가로 사용하거나, 다른 디렉토리에서 사용하도록 할 때에 유용
- 하드링크
  - 기존 파일에 대한 포인터를 직접 생성

    $ ln file1 file2

    $ ln file1 sub/file2

    $ ln file1 sub
  - 같은 파티션에 있는 파일에 대해서만 생성할 수 있음
- 심볼릭 링크(소프트 링크)
  - 연결할 파일에 대한 경로 이름을 저장 (윈도우의 바로가기와 유사)

    $ ln –s file1 file3

    $ ls -l

    lrwxrwxrwx. 1 ... file3 -> file1

링크 파일 삭제

- 링크 파일 삭제
  - file2 – file1의 hard link 파일, file3 – file1의 symbolic link 파일

    $ rm file1 … 원본 파일 삭제

    $ cat file2 … 파일 접근 가능 (직접 연결되므로)

    $ cat file3 … 파일 접근 불가능 (연결 파일이 삭제되었으므로)