

03.라즈베리파이 운영체제

강의 목표

1. 리눅스 운영체제의 특징을 이해한다.
2. 리눅스에서 사용자 관리를 이해한다.
3. 리눅스 파일 시스템의 구조를 이해하고 관련 명령을 활용할 수 있다.
4. 파일 속성을 이해하고 수정할 수 있다.
5. 리눅스에서 프로세스를 이해하고 관련 명령을 활용할 수 있다.
6. 리눅스에서 많이 사용하는 명령을 활용할 수 있다.



라즈베리파이 운영체제

리눅스 역사

4

□ 리눅스 탄생

- ▣ 리누스 토발즈(Linus Torvalds)에 의해 개발
 - 헬싱키대학교의 대학원생
 - 유닉스 운영체제를 모델로 하여 개발
 - 1991년 9월 17일, **리눅스 커널(Linux Kernel)** 공개

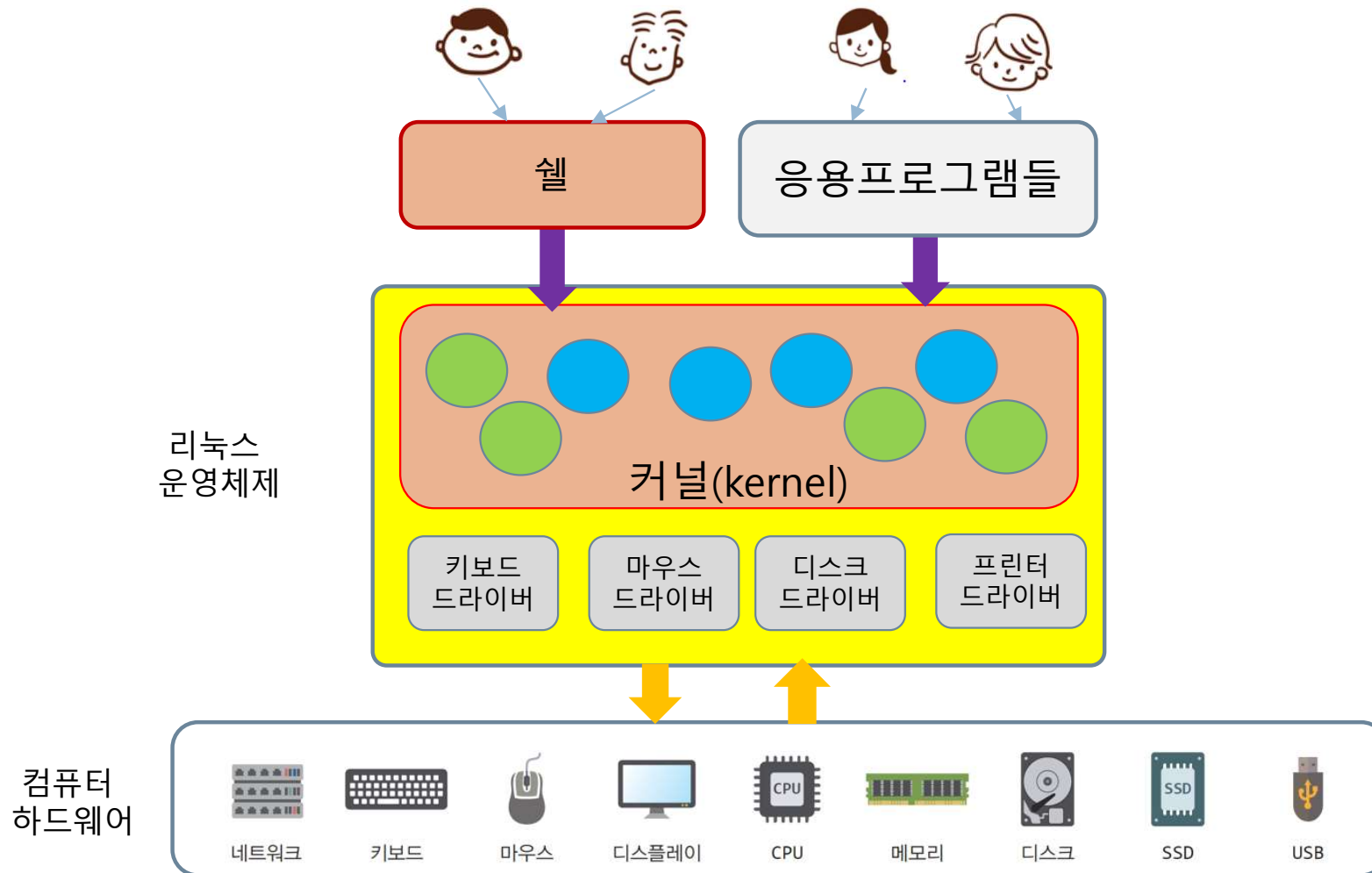


리눅스 운영체제 구성

5

□ 운영체제

- ▣ 사용자와 컴퓨터 하드웨어 사이에서 중계 역할을 하면서, 프로그램의 실행을 관리하고 제어하는 시스템 소프트웨어



리눅스 운영체제의 특징

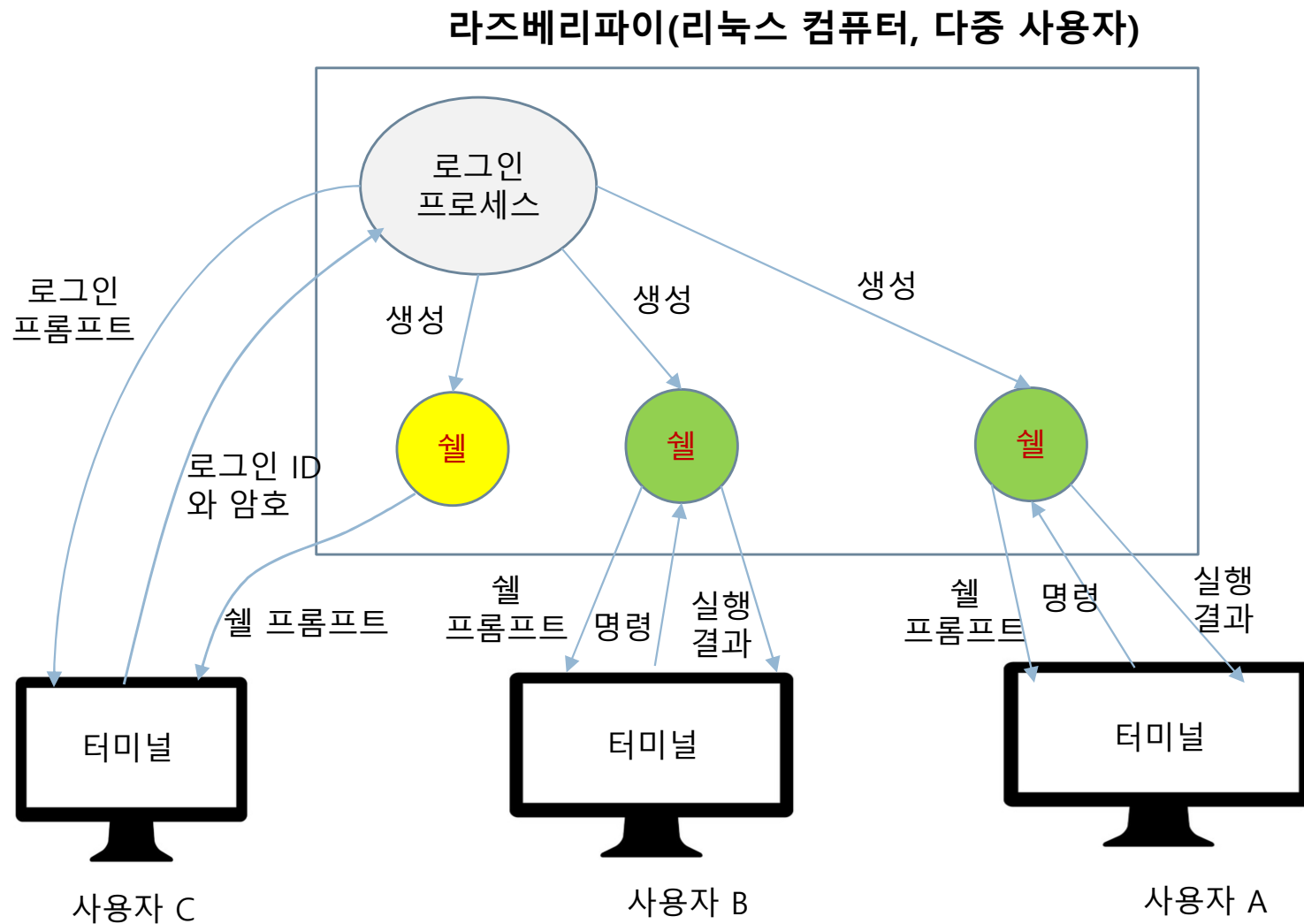
6

- 리눅스는 공개 소프트웨어
 - ▣ 누구나 수정, 개발, 배포 가능
- 유닉스 운영체제와 호환
 - ▣ 유닉스와 동일한 API와 명령 사용
- 높은 이식성
 - ▣ 리눅스 커널 및 응용 프로그램은 다양한 종류의 CPU를 가진 하드웨어 플랫폼에 쉽게 설치 가능
- 다중사용자/다중프로세스/멀티스레드 지원
 - ▣ 여러 사용자가 동시에 사용하는 다중 사용자 운영체제
 - ▣ 동시에 여러 응용프로그램(태스크 혹은 프로세스)이 동시에 실행
- 다양한 배포판
- 셸
 - ▣ 사용자로부터 명령을 입력받아 실행해주는 프로그램 제공
 - csh, tcsh, ksh, bash 등
 - 라즈베리파이에서는 bash 사용

로그인 과정과 셸

7

로그인 과정



셸 동작 사례

8

- 사용자가 입력한 ps 명령과 cd 명령을 실행하는 사례

```
pi@pi:~ $ ps
PID TTY TIME CMD
20888 pts/0 00:00:00 bash
24382 pts/0 00:00:00 ps
pi@pi:~ $ cd /etc
pi@pi:/etc $
```


셸과 프롬프트

9

□ 셸

- ▣ 사용자로부터 명령을 입력 받아 실행해주는 프로그램

□ 프롬프트

```
pi @ pi :/etc $
```

명령을 기다리는 프로그램이
셸임을 나타내는 기호

현재 디렉터리

호스트 이름

로그인 id

```
pi @ pi :~ $
```

명령을 기다리는 프로그램이
셸임을 나타내는 기호

홈 디렉터리에 있음을 나타냄



사용자 관리

사용자 유형

11

□ 루트 사용자

- ▣ 리눅스 설치 시 자동으로 생성되며, 계정 이름은 root
- ▣ 시스템에서 가장 높은 권한을 가짐
 - 어떤 디렉터리에도 진입 가능
 - 어떤 파일도 읽거나 쓰거나 삭제 가능
 - 어떤 디렉터리에도 파일 생성 가능
- ▣ 수퍼 사용자(super user)라고도 함

□ 일반 사용자

- ▣ 로그인 후 자신의 홈 디렉터리에 자유롭게 디렉터리나 파일을 만들 수 있음
 - 다른 디렉터리나 파일은 권한이 부여된 경우에만 액세스 허용
- ▣ 라즈베리파이에서 대표적인 일반사용자: pi

사용자 계정 정보

12

- 사용자 계정과 관련된 관리 파일 4개
 - ▣ /etc/passwd, /etc/shadow, /etc/group, /etc/gshadow 텍스트
- /etc/passwd 파일
 - ▣ 모든 사용자 계정 저장
 - ▣ 로그인 과정에서 시스템에 존재하는 사용자인지 판단할 때 사용
 - ▣ /etc/passwd 파일의 한 행
 - 하나의 사용자 계정에 대한 7가지 정보 포함
 - 사용자 이름
 - 암호 표시 문자
 - UID(User ID)
 - GID(Group ID)
 - 사용자 계정에 대한 설명
 - 홈 디렉터리
 - 로그인 셸

/etc/passwd 파일 내용 예

13

```
pi@pi:~ $ cat /etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
```

```
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

```
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

```
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

```
mosquitto:x:116:124::/var/lib/mosquitto:/usr/sbin/nologin
```

```
...
```

```
pi:x:1000:1000:,,,:/home/pi:/bin/bash
```

```
...
```

```
jmlee:x:1003:1003:Prof. Jae Moon Lee:/home/jmlee:/bin/bash
```

```
kitae:x:1004:1004:Prof. Kitae Hwang:/home/kitae:/bin/bash
```

```
...
```

```
pi@pi:~ $
```

/etc/passwd 파일 구성

14

- /etc/passwd 파일의 한 행 - 한 사용자에 대한 정보

kitae:x:1000:1000:Prof. Kitae Hwang:/home/kitae:/bin/bash



sudo 명령

15

- 루트 사용자 권한이 필요할 때 사용
 - ▣ 'superuser do'의 약자로 루트 사용자의 권한으로 명령 실행
 - ▣ 루트 사용자로부터 위임받은 사용자에게 한해, 루트 사용자와 동일한 권한으로 실행
- 라즈베리파이 운영체제를 설치할 때 만든 pi 사용자
 - ▣ sudo 사용 가능
 - ▣ sudo 명령을 실행할 권한을 위임받는 사용자

pi@pi:~ \$ halt

Failed to set wall message, ignoring: Interactive authentication required.

Failed to halt system via logind: Interactive authentication required.

Failed to open initctl fifo: 허가 거부

Failed to talk to init daemon.

pi@pi:~ \$ sudo halt

Connection to 192.168.0.11 closed by remote host.

Connection to 192.168.0.11 closed.

sudo halt 명령을 실행하면 라즈베리파이를 끄게 되므로 하지 말것

사용자 관리 명령

16

□ 사용자 관리 명령

명령	내용
useradd	새로운 사용자를 등록한다. 유사 명령 adduser도 있음
userdel	기존의 사용자를 삭제한다. 유사 명령 deluser도 있음
usermod	사용자의 정보를 변경함
who	호스트에 로그인한 사용자의 정보를 자세히 출력
users	호스트에 로그인한 사용자의 정보를 간략히 출력
whoami	현재 로그인한 사용자 이름을 출력
passwd	비밀번호를 변경

□ 사용자 계정 관리 주요 옵션

옵션	관련 명령	내용
-m	useradd, usermod	홈 디렉터리를 생성할 것을 지시
-d	useradd, usermod	생성할 홈 디렉터리 경로명 지정
-c	useradd	사용자 계정에 대한 설명을 문자열로 지시
-g	useradd, usermod	그룹을 지정
-r	userdel	홈 디렉터리와 메일 스푼을 삭제

사용자 계정 생성

17

(1) useradd 명령으로 사용자 계정 생성

```
pi@pi:~ $ sudo useradd kitae -m
```

```
pi@pi:~ $ sudo useradd kitae -c "Prof. Hwang" -m
```

- ▣ 생성후 /etc/passwd 내용

```
kitae:x:1001:1001:Prof. Hwang:/home/kitae:/bin/bash
```

(2) passwd 명령으로 사용자 암호 설정

```
pi@pi:~ $ sudo passwd kitae
```

```
새 암호:[암호입력]
```

입력된 암호는 보이지 않음

```
새 암호 재입력:[암호입력]
```

```
passwd: 암호를 성공적으로 업데이트했습니다
```

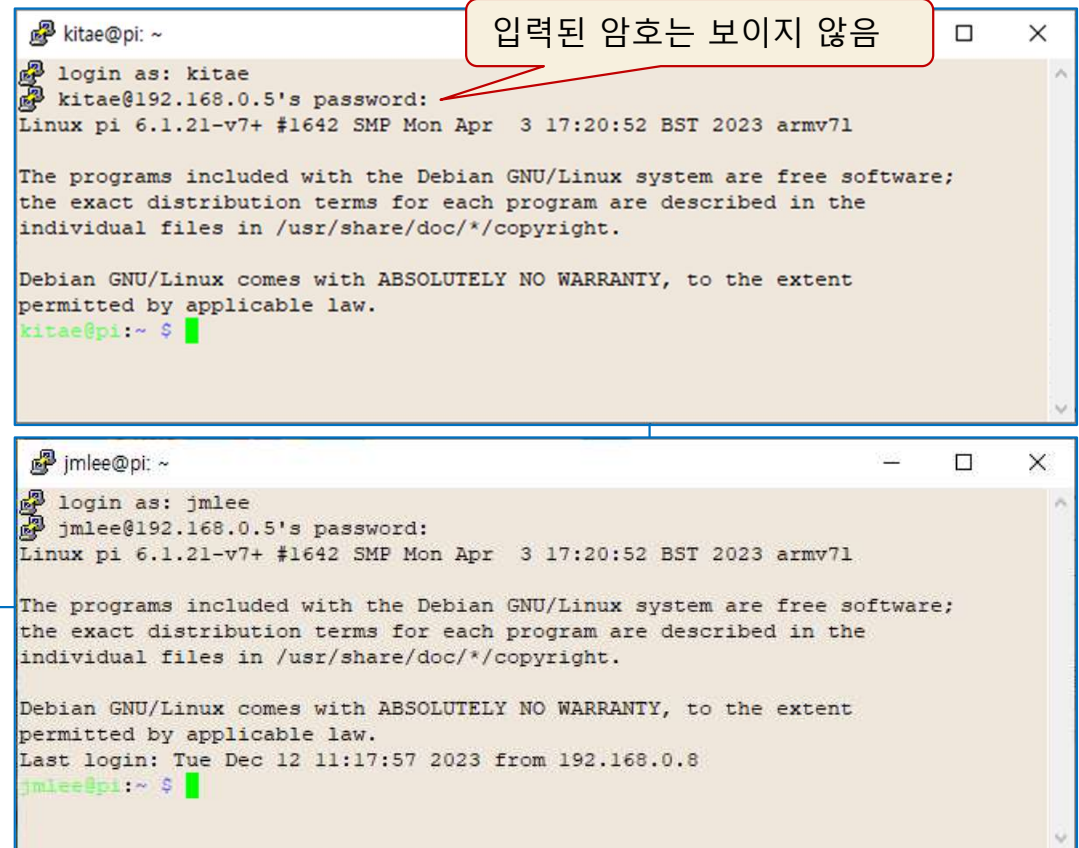
```
pi@pi:~ $
```

예제 3-1 사용자 kitae와 jmlee 추가 실습

18

useradd 명령을 활용하여 kitae와 jmlee 사용자 계정을 추가하라. 이때 사용자에게 대한 설명(-c옵션)으로 "student"를 넣어라. 또한 passwd 명령을 활용하여 암호를 설정하고, 터미널을 열어 kitae와 jmlee로 각각 로그인하여 보라.

```
pi@pi:~ $ sudo useradd kitae -c student -m
pi@pi:~ $ sudo useradd jmlee -c student -m
pi@pi:~ $ sudo passwd kitae
New password:[암호입력]
Retype new password:[암호입력]
passwd: password updated successfully
pi@pi:~ $ sudo passwd jmlee
New password:[암호입력]
Retype new password:[암호입력]
passwd: password updated successfully
pi@pi:~ $ ls /home
jmlee kitae pi
pi@pi:~ $
```



```
kitae@pi: ~
login as: kitae
kitae@192.168.0.5's password:
Linux pi 6.1.21-v7+ #1642 SMP Mon Apr 3 17:20:52 BST 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
kitae@pi:~ $
```

입력된 암호는 보이지 않음

```
jmlee@pi: ~
login as: jmlee
jmlee@192.168.0.5's password:
Linux pi 6.1.21-v7+ #1642 SMP Mon Apr 3 17:20:52 BST 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Dec 12 11:17:57 2023 from 192.168.0.8
jmlee@pi:~ $
```

사용자 계정 삭제와 로그인 사용자 보기

19

- userdel 명령으로 사용자 계정 삭제
 - ▣ 사용자 계정, 홈 디렉터리, 도착했지만 아직 읽지 않은 메일이 저장된 메일 스푼(mail spool) 등 모두 삭제

```
pi@pi:~ $ sudo userdel -r jmlee
```

- 현재 로그인한 사용자 보기, who와 users

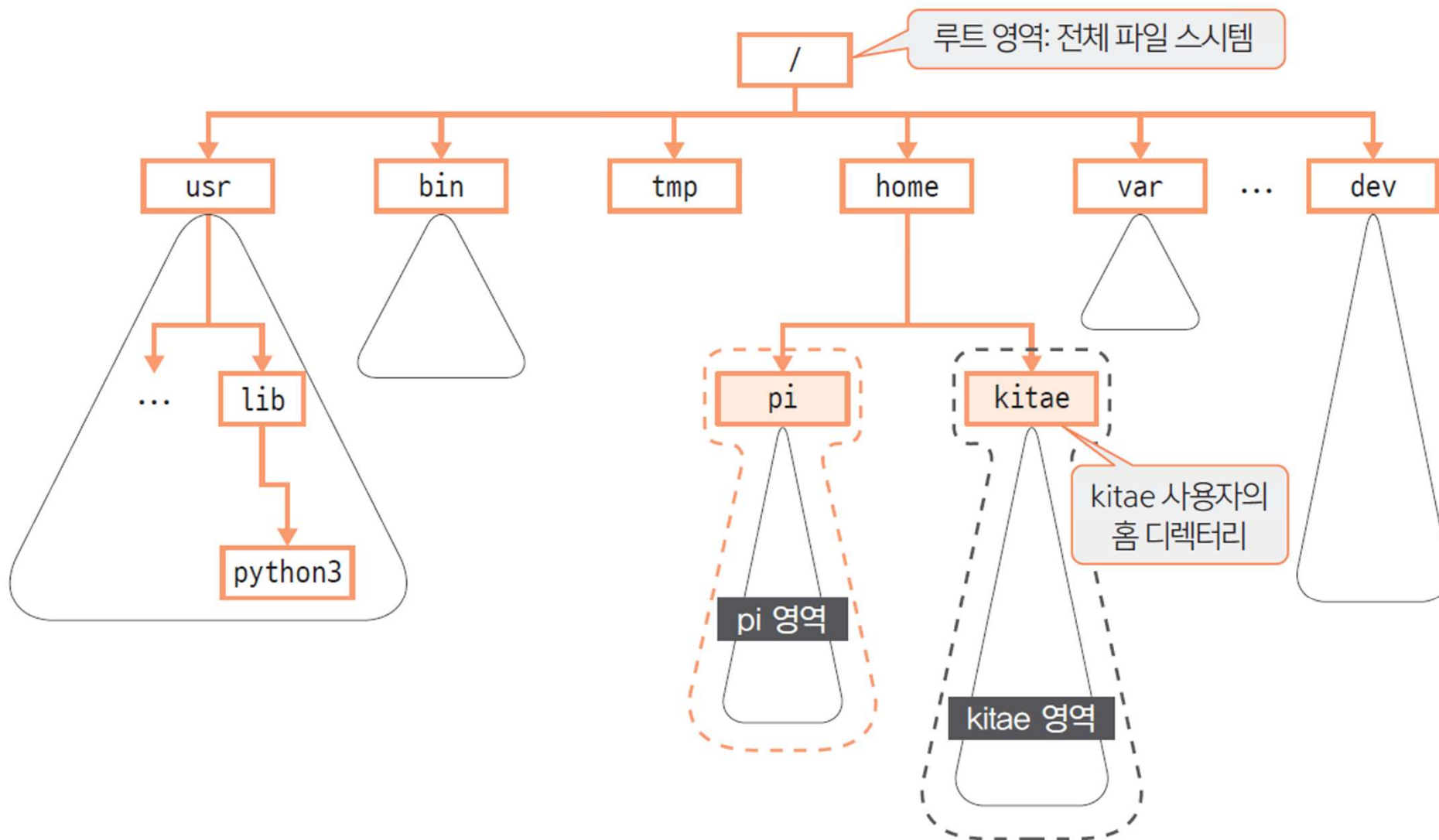
```
pi@pi:~ $ who
pi tty1          2023-04-07 15:02
pi tty7          2023-04-07 15:02 (:0)
pi pts/0         2023-04-08 10:43 (192.168.0.10)
kitae pts/2      2023-04-09 23:55 (192.168.0.12)
pi@pi:~ $ users
kitae pi pi pi
pi@pi:~ $
```



파일 관리

리눅스의 파일 시스템 구조

21



* 파일 시스템의 모든 파일은 sd 카드에 저장됨

파일 시스템 경로

22

- 절대경로
 - ▣ 루트 디렉터리에서 시작하여 특정 디렉터리나 파일의 경로 표현
- 상대경로
 - ▣ 현재 디렉터를 기준으로 특정 디렉터리나 파일의 경로 표현

절대경로 사용하기

23

- 절대경로 사용 사례
 - ▣ pi 디렉터리의 절대 경로
 - /home/pi
 - ▣ wpa_supplicant.conf 파일의 절대 경로
 - /etc/wpa_supplicant/wpa_supplicant.conf
 - ▣ 파이선이 설치된 디렉터리의 절대 경로
 - /usr/lib/python3

```
pi@pi:~ $ cd /usr/lib/python3
pi@pi:/usr/lib/python3 $
```

상대경로 사용하기

24

□ 특수 기호와 함께 상대 경로 사용

▣ (.)과 (..) 기호 활용

- . : 현재 디렉터리

```
pi@pi:~ $ cd ../../kitae
pi@pi:/home/kitae $
```

```
pi@pi:/home/kitae $ cd ..
pi@pi:/home $
```

▣ 상대 경로에 ~ 기호 활용

- ~ : 현재 사용자의 홈 디렉터리

```
pi@pi:/home/kitae $ cd ~
pi@pi:~ $ pwd
/home/pi
pi@pi:~ $
```


vi를 이용한 파일 편집 (1)

25

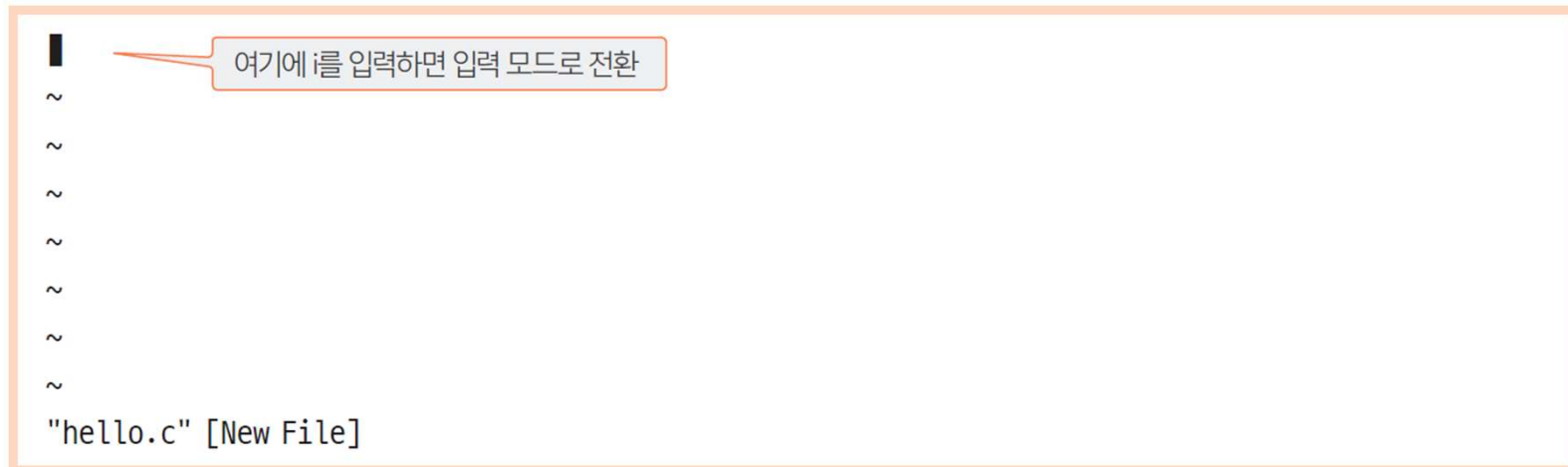
- vi - 유닉스 계열 운영체제의 대표 파일편집프로그램

[단계 1] vi 실행으로 hello.c 생성하기

```
pi@pi:~ $ vi hello.c
```

[단계 2] 명령모드 -> 입력모드 전환하기

- ▣ 아래와 같이 i 키를 입력



```
~
~
~
~
~
~
~
~
~
"hello.c" [New File]
```

여기에 i를 입력하면 입력 모드로 전환

vi를 이용한 파일 편집 (2)

26

[단계 3] 입력하기

- ▣ 코드 입력 - 현재 그대로 틀린 상태로 입력

```
#include <stdio.h>
@include <unistd.h>
int main() {
    Whlie(1{
        printf("Hello\n");
        sleep(1)
    }
    return 0;
}
~
~
```

오류난 곳 3개 수정 필요

@ → #

While(1{ → while(1) {

sleep(1) → sleep(1);

vi를 이용한 파일 편집 (3)

27

[단계 4] 'ESC' 키로 입력모드에서 명령모드로 전환

- ▣ 입력모드 상태에서 'ESC' 키 입력

[단계 5] vi 편집 종료

- ▣ 명령모드에서 ':' 키 입력, vi 창의 맨 밑에 나타남
 - 맨 밑에 나타나지 않으면 아직도 입력 모드이므로 '단계 4' 다시 실행 후 '단계 5' 실행

```
#include <stdio.h>
#include <unistd.h>
int main() {
    Whlie(1{
        printf("Hello\n");
        sleep(1)
    }
    return 0;
}
~
:wq!
```

설명을 위해 만든 오류 1

설명을 위해 만든 오류 2

설명을 위해 만든 오류 3

여기에 :wq!를 입력

vi를 이용한 파일 편집 (4)

28

[단계 6] hello.c 파일 확인

- ▣ ls 명령을 사용하여 hello.c가 생성된 것 확인

```
pi@pi:~ $ ls
hello.c
pi@pi:~ $
```

- ▣ cat 명령을 사용하여 hello.c의 내용 확인

```
pi@pi:~ $ cat hello.c
#include <stdio.h>
#include <unistd.h>
int main() {
    while(1{
        printf("Hello\n");
        sleep(1)
    }
    return 0;
}
pi@pi:~ $
```

vi를 이용한 파일 편집 (5)

29

[단계 7] vi로 hello.c 수정

- ▣ \$ vi hello.c

```
#include <stdio.h>
@include <unistd.h>
int main(){
    Whlie(1{
        printf("Hello\n");
        sleep(1)
    }
    return 0;
}
~
~
"hello.c" 9 lines, 109 bytes
```

[단계 8] 명령모드에서(현재 명령모드) 커서 이동

- 커서(화면 상의 검은 사각형)
- ▣ 상하좌우 화살키를 사용하여 커서 이동

vi를 이용한 파일 편집 (6)

30

[단계 9] 편집된 내용 수정 (1)

▣ 첫번째 오류 '@include'를 '#include'로 수정

- (1) 상하좌우 화살 키를 이용하여 커서를 '@'위로 이동
- (2) 'x' 키 입력
 - 'x' 키는 현재 커서 위치의 문자를 지우는 편집 명령
 - '@' 가 지워지고 왼쪽으로 'include' 가 이동되며, 커서는 현재 'i' 위에 있게 됨
- (3) 'i' 키를 입력하여 입력모드로 전환, '#' 키를 입력하면 커서가 있는 위치에 '#' 문자 삽입
- (4) 'ESC' 키를 입력하여 명령모드로 전환

vi를 이용한 파일 편집 (7)

31

[단계 9] 편집된 내용 수정(2)

▣ 두 번째 오류 'Whlie(1{'; 을 'while(1) {' 로 수정

- 이 라인에 여러 오류 존재, 이 라인을 지우고 새로 입력하는 방법 설명
- (1) 상하좌우 화살 키 활용, 커서를 'Whlie(1{';가 있는 라인의 아무 위치로 이동
- (2) 'dd' 키 입력(d를 2번 입력).
 - dd는 커서가 있는 라인을 지우는 명령
- (3) 커서를 'main' 있는 라인의 아무 위치로 이동, 'o' 키 입력
 - 'o' 키는 커서가 있는 라인의 아래에 빈 줄을 만들고, 빈 줄로 커서를 옮기고, 입력모드로 바꾸는 명령
- (4) 현재 입력모드이므로 [TAB] 키 입력, 'while(1) {'를 입력한 후, 'ESC' 키를 입력하여 명령모드로 전환

vi를 이용한 파일 편집 (8)

32

[단계 9] 편집된 내용 수정(3)

▣ 마지막 오류, 'sleep(1)' 끝에 ';'을 추가

- (1) 커서를 'sleep(1)'의 마지막 글자 ')' 위치로 이동
- (2) 'a' 키 입력.

- 'a' 키는 입력모드로 바꿨고, 커서 다음 위치부터 입력을 시작하는 명령. (참고로 'sleep(1)' 이 있는 라인의 아무 위치에서 [Shift]+A 키를 입력하면 현재 라인의 마지막부터 입력할 수 있다.)

- (3) ';' 키 입력, 'ESC' 키를 입력하여 명령모드로 전환

vi를 이용한 파일 편집 (9)

33

[단계 9] 최종 수정된 결과

```
#include <stdio.h>
#include <unistd.h>
int main() {
    while(1) {
        printf("Hello\n");
        sleep(1);
    }
    return 0;
}
~
~
```

vi를 이용한 파일 편집 (10)

34

[단계 10] 종료하기

- ▣ [단계 5]와 동일

[단계 11] 올바른 입력 확인하기

- ▣ hello.c를 컴파일하여 올바른 입력 확인
- ▣ 리눅스의 c 컴파일러는 gcc - 컴파일 결과 a.out 실행 파일 생성

```
pi@pi:~ $ gcc hello.c
pi@pi:~ $ ls
a.out hello.c
pi@pi:~ $
```

- ▣ a.out 실행

```
pi@pi:~ $ ./a.out
Hello
Hello
Hello
^C
pi@pi:~ $
```

디렉터리와 파일 관련 명령 (1)

35

□ 파일 관련 명령 요약

명령	내용
\$ pwd	현재 디렉터리의 절대 경로 출력
\$ ls path	파일 및 서브 디렉터리 목록 출력
\$ mkdir path/name	디렉터리 생성
\$ cd path	디렉터리 이동
\$ cp path1/name1 path2/name2	파일 복사
\$ mv path1/name1 path2/name2	파일 이동 혹은 파일 이름 변경
\$ tree path	path의 서브 디렉터리를 트리 형태로 출력
\$ rm path/name	파일 삭제
\$ cat path/name	파일 내용 출력
\$ more path/name	파일 내용을 페이지 단위로 출력
\$ vi path/name	파일을 생성하거나 수정

디렉터리와 파일 관련 명령 (2)

36

□ pwd

- ▣ pwd는 'print working directory'를 줄인 명령
- ▣ 현재 작업하고 있는 디렉터리 경로 출력

```
pi@pi:~ $ pwd  
/home/pi  
pi@pi:~ $
```

□ ls

- ▣ 디렉터리에 있는 파일과 서브디렉터리들을 출력하는 명령

```
pi@pi:~ $ ls  
a.out hello.c  
pi@pi:~ $
```

디렉터리와 파일 관련 명령 (3)

37

□ ls -l

- -l(영어 엘) 옵션은 자세히 설명하라는 지시
- ls -l을 사용하면 다양한 정보를 얻을 수 있음

```
pi@pi:~ $ ls -l
total 16
-rwxr-xr-x 1 pi pi 8108 12월 26 11:19 a.out
-rw-r--r-- 1 pi pi 113 12월 26 11:18 hello.c
pi@pi:~ $
```

□ ls -al

- -a 옵션은 'all'의 약자, 숨겨진 정보도 함께 출력

```
pi@pi:~ $ ls -al
total 32
drwxr-xr-x 4 pi pi 4096 12월 26 11:27 .
drwxr-xr-x 32 pi pi 4096 12월 26 11:18 ..
-rwxr-xr-x 1 pi pi 8108 12월 26 11:19 a.out
-rw-r--r-- 1 pi pi 113 12월 26 11:18 hello.c
pi@pi:~ $
```

디렉터리와 파일 관련 명령 (4)

38

□ mkdir

- ▣ 새로운 디렉터리를 생성하는 명령

```
pi@pi:~ $ mkdir ch03
pi@pi:~ $ ls
a.out ch03 hello.c
pi@pi:~ $ mkdir ch03/test
pi@pi:~ $ ls ch03
test
pi@pi:~ $
```

디렉터리와 파일 관련 명령 (5)

39

□ mv

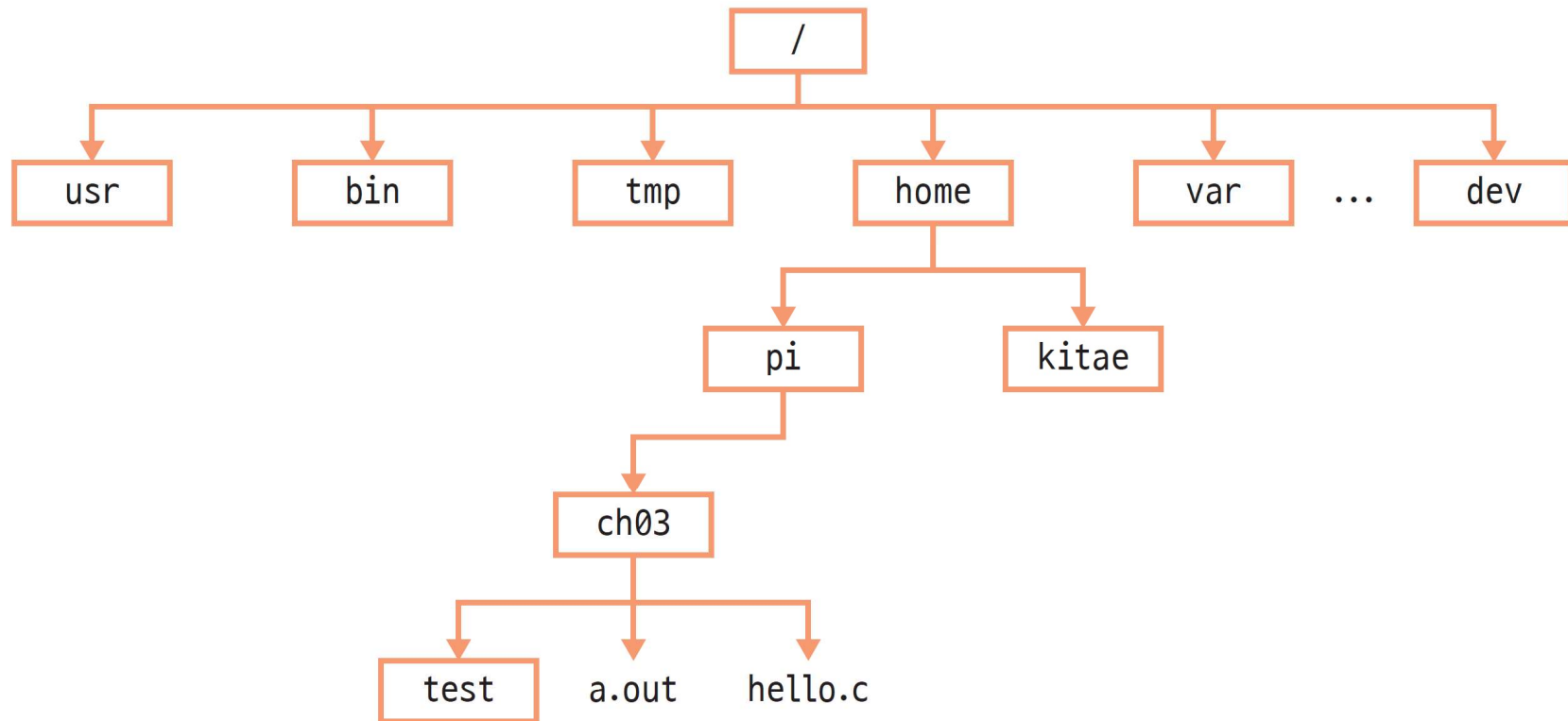
- ▣ 파일을 이동시키는 명령
 - 파일 이름도 바꿀 수 있음

```
pi@pi:~ $ mv a.out ch03/a.out
pi@pi:~ $ mv hello.c ch03/.
pi@pi:~ $ ls
ch03
pi@pi:~ $ ls ch03
a.out hello.c test
pi@pi:~ $
```

디렉터리와 파일 관련 명령 (6)

40

- 앞의 여러 명령 실행 결과로 구성된 디렉터리 구조



디렉터리와 파일 관련 명령 (7)

41

□ cd

- ▣ 현재 디렉터리를 이동시키는 명령

```
pi@pi:~ $ cd ch03
pi@pi:~/ch03 $ cd test
pi@pi:~/ch03/test $ ls
pi@pi:~/ch03/test $ cd ../
pi@pi:~/ch03 $
```

디렉터리와 파일 관련 명령 (8)

42

□ cp - 파일 복사 명령

- 사례 : 현재 디렉터리의 hello.c 파일을 복사하여 hello2.c 파일 생성

```
pi@pi:~/ch03 $ cp hello.c hello2.c
pi@pi:~/ch03 $ ls
a.out hello2.c hello.c test
pi@pi:~/ch03 $
```

- 사례 : 현재 디렉터리의 hello.c 파일을 복사하여 test 디렉터리에 hello2.c 생성

```
pi@pi:~/ch03 $ cp hello.c ./test/hello2.c
pi@pi:~/ch03 $ ls test
hello2.c
pi@pi:~/ch03 $
```

- 사례 : 현재 디렉터리의 hello.c 파일을 복사하여 test 디렉터리에 같은 이름으로 생성

```
pi@pi:~/ch03 $ cp hello.c ./test
pi@pi:~/ch03 $ ls test
hello2.c hello.c
pi@pi:~/ch03 $
```

디렉터리와 파일 관련 명령 (9)

43

□ rm

▣ 파일 삭제 명령

```
pi@pi:~/ch03 $ ls
a.out hello2.c hello.c test
pi@pi:~/ch03 $ rm hello2.c
pi@pi:~/ch03 $ ls
a.out hello.c test
pi@pi:~/ch03 $
```

▣ -r 옵션을 사용하면, 하위 디렉터리까지 지우기

```
pi@pi:~/ch03 $ ls
a.out hello.c test
pi@pi:~/ch03 $ rm -r test
pi@pi:~/ch03 $ ls
a.out hello.c
pi@pi:~/ch03 $
```

디렉터리와 파일 관련 명령 (10)

44

□ cat

- ▣ 텍스트 파일의 내용을 출력하는 명령
 - 텍스트 파일만 출력 가능. 바이너리 파일 a.out은 출력 안 됨

```
pi@pi:~/ch03 $ cat hello.c
#include <stdio.h>
#include <unistd.h>
int main() {
    while(1) {
        printf("Hello\n");
        sleep(1);
    }
    return 0;
}
pi@pi:~/ch03 $
```

□ more

- ▣ cat 명령과 동일하지만, 페이지 단위로 끊어서 출력

파일 관련 특수 기호 ~ 활용

45

□ ~ 기호

- ▣ '~'는 파일의 경로명에서 홈 디렉터리를 나타냄. '틸다'라고 읽음
 - '~/ch03/test'는 홈 디렉터리/ch03에 있는 test 디렉터를 뜻함

```
pi@pi:~/ch03 $ ls
a.out hello.c
pi@pi:~/ch03 $ mkdir test
pi@pi:~/ch03 $ cp a.out ~/ch03/test
pi@pi:~/ch03 $ ls test
a.out
pi@pi:~/ch03 $
```

파일 관련 특수 기호 * 활용

46

□ * 기호

▣ '*'는 문자열 와일드카드

- 명령에 '*'가 있으면, 그 곳에 어떠한 단어가 들어갈 수 있음을 뜻함

```
pi@pi:~/ch03 $ ls
a.out hello.c test
pi@pi:~/ch03 $ cp h*.c ~/ch03/test
pi@pi:~/ch03 $ ls test
a.out hello.c
pi@pi:~/ch03 $
```

파일 속성

47

- 파일 속성 관점에서 3가지 유형의 사용자
 - ▣ 나(user)
 - ▣ 나와 같은 그룹에 속한 사용자(group)
 - ▣ 나와 다른 그룹에 속한 사용자(other)
- 파일 속성 보기
 - ▣ 파일에 관한 다양한 정보, 'ls -l'이나 'ls -al' 명령으로 볼 수 있음

```
pi@pi:~/ch03 $ ls -l
```

```
total 20
```

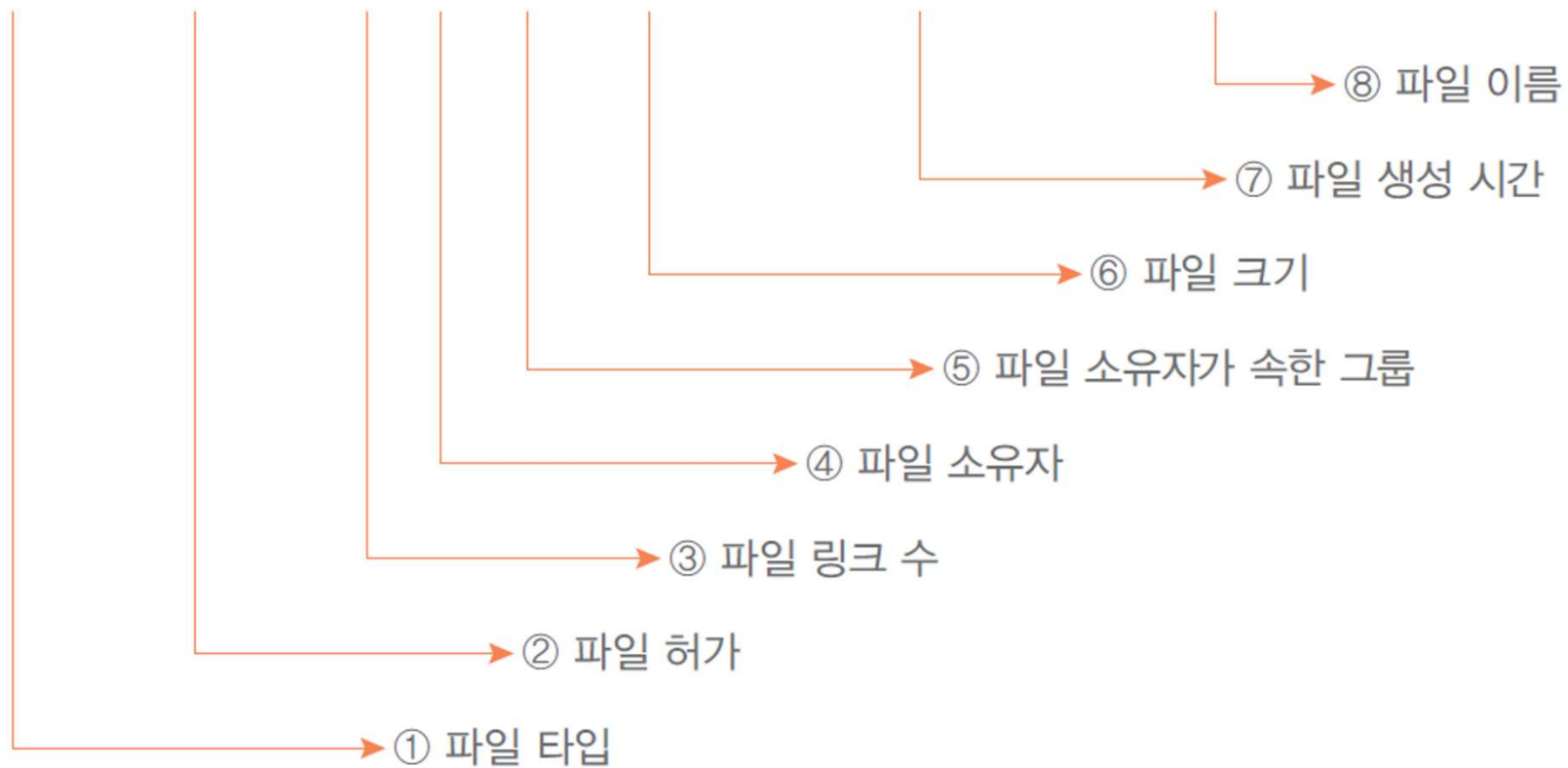
-rwxr-xr-x	1	pi	pi	8108	12월 26 12:36	a.out
-rw-r--r--	1	pi	pi	113	12월 26 12:27	hello.c
drwxr-xr-x	2	pi	pi	4096	12월 26 13:52	test

```
pi@pi:~/ch03 $
```

파일 속성 종류

48

- rwxr-xr-x 1 pi pi 8108 12월 26 12:36 a.out



파일 속성 설명

49

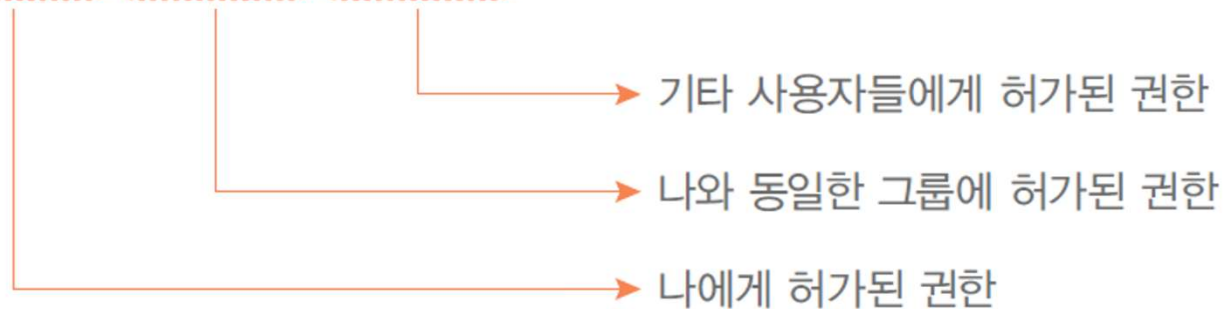
파일 속성 항목	내용
① 파일 타입	파일 속성의 첫 번째 문자로서, -는 일반 파일, d는 디렉터리, b는 블록 장치, c는 문자 장치, l은 링크
② 파일 허가	사용자 유형별 파일 접근(읽기: r, 쓰기: w, 실행: x) 허가
③ 파일 링크 수	하위 디렉터리에서 디렉터리 수
④ 파일 소유자	파일을 만든 사용자
⑤ 파일 그룹	파일 소유자가 속한 그룹
⑥ 파일 크기	파일의 경우 파일의 크기(바이트)
⑦ 파일 시간	파일 생성/수정한 마지막 시간
⑧ 파일 이름	파일 또는 디렉터리 이름

파일 허가

50

□ 사용자 유형별 파일 허가

- r w x r w x r w x



□ r

- 파일 읽기 허용

□ w

- 파일 쓰거나 수정 허용
- 파일 쓰기가 허용되면 파일 삭제도 가능

□ x

- 파일의 실행 허용(실행 가능한 파일에 대해서만 의미 있음)
- 디렉터리의 경우 디렉터리 안으로 이동 가능

파일 허가 사례 (1)

51

(1) hello.c 파일 허가 사례

```
-rw-r--r-- 1 pi pi 113 12월 26 12:27 hello.c
```

- 기타 사용자들은 읽을 권한만 허용
- pi와 동일한 그룹에 읽는 권한만 허용
- pi 사용자는 읽고, 쓸 수 있으나 실행은 불허

(2) a.out 파일의 실행 허가 사례

- ▣ a.out이 실행 파일
- ▣ a.out을 실행시키려면 x 옵션 설정 필요

```
-rwxr-xr-x 1 pi pi 8072 12월 26 12:36 a.out
```

파일 허가 사례 (2)

52

(3) test 디렉터리로 진입 사례

```
drwxr-xr-x 2 pi pi 4096 12월 26 13:52 test
```

- 파일 속성의 맨 앞의 문자가 'd'이므로, test는 디렉터리임
- 디렉터리의 경우 파일 허가 옵션 중 x 옵션이 있으면, 디렉터리로 진입하는 것을 허가한다는 의미
- test 디렉터리의 파일 허가 속성에는 모든 사용자 유형에 대해 x 옵션이 들어 있으므로 pi 사용자를 비롯한 모든 사용자가 cd 명령으로 test 디렉터리에 진입할 수 있음

파일 허가 변경 (1)

53

- chmod 명령을 이용한 파일 허가 변경
 - ▣ 루트 사용어나 파일의 소유자만 chmod 명령 사용 가능

chmod 파일허가 파일이름

(1) 8진수로 파일 허가 표현

```
rw-r--r-- -> 110 100 100 -> 644  
rwxrwx-- -> 111 110 000 -> 760  
rwxrwxrwx -> 111 111 111 -> 777
```

- ▣ hello.c 파일의 허가 속성을 664로 변경하는 사례

```
pi@pi:~/ch03 $ chmod 664 hello.c  
pi@pi:~/ch03 $ ls -l hello.c  
-rw-rw-r-- 1 pi pi 112 12월 26 12:27 hello.c  
pi@pi:~/ch03 $
```

파일 허가 변경 (2)

54

(2) +, -, u, g, o 옵션으로 파일 허가 변경

- ▣ u(user), g(group), o(other), a(all)와 +, -, r, w, x 문자 함께 이용
- ▣ +는 권한 추가
- ▣ -는 권한 삭제

```
pi@pi:~/ch03 $ chmod u+w hello.c // pi 사용자에게 쓰기 권한 부여
pi@pi:~/ch03 $ chmod g+w hello.c // pi와 동일 그룹 사용자들에게 쓰기 권한 부여
pi@pi:~/ch03 $ chmod a+r hello.c // 모든 사용자에게 읽기 권한 부여
pi@pi:~/ch03 $ chmod o+rw hello.c // 기타 사용자들에게 읽기와 쓰기 권한 부여
pi@pi:~/ch03 $ chmod o-x a.out // 기타 사용자들에게 실행 권한 제거
pi@pi:~/ch03 $ chmod go-x test // 다른 사용자가 test 디렉터리에 들어가는 권한 제거
```

예제 3-2 다중 사용자들 사이의 파일 허가 실습

55

예제 3-1에서 생성한 kitae와 pi 계정을 활용하여 각각 로그인하고, pi는 hello.c를 복사하여 helloc2.c를 만든 후 hello2.c는 kitae가 볼 수 없도록 파일 허가 속성을 변경하라.

터미널1에서 pi로 로그인

```
pi@pi:~ $ cd ~/ch03
pi@pi:~/ch03 $ ls -l
total 20
-rwxr-xr-x 1 pi pi 8072 12월 26 12:36 a.out
-rw-r--r-- 1 pi pi 99 1월 13 08:38 hello2.c
-rw-r--r-- 1 pi pi 99 12월 26 12:27 hello.c
drwxr-xr-x 2 pi pi 4096 12월 26 13:52 test
pi@pi:~/ch03 $ chmod 711 hello2.c
pi@pi:~/ch03 $ ls -l
total 20
-rwxr-xr-x 1 pi pi 8072 12월 26 12:36 a.out
-rwx--x--x 1 pi pi 99 1월 13 08:38 hello2.c
-rw-r--r-- 1 pi pi 99 12월 26 12:27 hello.c
drwxr-xr-x 2 pi pi 4096 12월 26 13:52 test
pi@pi:~/ch03 $
```

예제 3-2 (계속)

56

터미널2에서 kitae로 로그인

```
kitae@pi:~ $ cat /home/pi/ch03/hello.c
```

```
#include <stdio.h>
```

```
#include <unistd.h>
```

```
int main(){
```

```
    while(1){
```

```
        printf("Hello\n");
```

```
        sleep(1);
```

```
    }
```

```
    return 0;
```

```
}
```

```
kitae@pi:~ $ cat /home/pi/ch03/hello2.c
```

```
cat: /home/pi/ch03/hello.c: 허가 거부
```

```
kitae@pi:~ $
```

kitae 사용자는 hello2.c 파일을
읽을 허가 없기 때문



프로세스 관리

□ 프로세스

- ▣ 프로그램이 실행 중일 때 이를 프로세스(process)라고 부름
- ▣ 운영체제
 - 프로세스에게 유일한 프로세스 번호(PID)를 할당
 - PID를 포함하여, 프로세스의 스케줄링 우선순위, 프로세스가 사용한 CPU 시간, 프로세스의 상태 정보, 부모프로세스의 번호 등 운영체제 내부에 유지
- ▣ 부모 프로세스 : 자식 프로세스를 만든 프로세스
- ▣ 자식 프로세스 : 새로 생성된 프로세스

□ 프로세스 관리란?

- ▣ 프로세스 생성, 프로세스 실행, 프로세스 일시 중단, 프로세스 종료, 프로세스들 간의 정보 소통 등에 대한 관리
- ▣ 운영체제에 의해 관리됨

프로세스 상태와 실행

59

- 프로세스의 실행 중인 2가지 상태
 - ▣ 포그라운드 상태(FG) - 키보드 입력을 독점하는 상태
 - ▣ 백그라운드 상태(BG) - 키보드에 대한 사용 권한이 없는 상태
- 포그라운드 상태로 프로세스 실행시키기

```
pi@pi:~/ch03 $ ./a.out  
Hello  
Hello
```

- 포그라운드 상태의 프로세스 강제 종료, [Ctrl+C] 키

```
pi@pi:~/ch03 $ ./a.out  
Hello  
Hello  
^C  
pi@pi:~/ch03 $
```

[Ctrl+C] 입력

프로세스 관리 (1)

60

- 포그라운드 상태의 프로세스 일시 정지, [Ctrl+Z] 키

```
pi@pi:~/ch03 $ ./a.out
```

```
Hello
```

```
Hello
```

```
^Z
```

[Ctrl+Z] 입력

```
[1]+  멈춤 ./a.out
```

```
pi@pi:~/ch03 $
```

- ps 혹은 jobs 명령
 - ▣ 프로세스 목록 보기

```
pi@pi:~/ch03 $ ps
```

```
PID  TTY  TIME  CMD
```

```
4746 pts/0 00:00:02 bash
```

```
16937 pts/0 00:00:00 a.out
```

```
16938 pts/0 00:00:00 ps
```

```
pi@pi:~/ch03 $
```

```
pi@pi:~/ch03 $ jobs
```

```
[1]+  멈춤 ./a.out
```

```
pi@pi:~/ch03 $
```

프로세스 관리 (2)

61

□ fg 명령

- 일시 정지된 프로세스를 다시 포그라운드 상태로 실행

```
pi@pi:~/ch03 $ fg
./a.out
Hello
Hello
^C [Ctrl+C] 입력
pi@pi:~/ch03 $
```

□ bg 명령

- 일시 정지된 프로세스를 백그라운드 상태로 실행

```
pi@pi:~/ch03 $ ./a.out
Hello
Hello
^Z [Ctrl+Z] 입력
[1]+  멈춤 ./a.out
pi@pi:~/ch03 $ bg
[1]+ ./a.out &
Hello
pi@pi :~/ch03 $
```

프로세스 관리 (3)

62

□ 백그라운드 상태의 프로세스를 포그라운드 전환

```
pi@pi:~/ch03 $ bg
```

```
[1]+ ./a.out &
```

```
Hello
```

```
pi@pi:~/ch03 $ Hello
```

```
Hello
```

a.out이 백그라운드 상태에서 출력한 내용

```
Hello
```

```
fg
```

fg 입력

```
./a.out
```

a.out이 포그라운드 상태로 전환됨을 나타냄

```
Hello
```

a.out이 포그라운드 상태에서 출력한 내용

```
^C
```

```
pi@pi:~/ch03 $
```

프로세스 관리 (4)

63

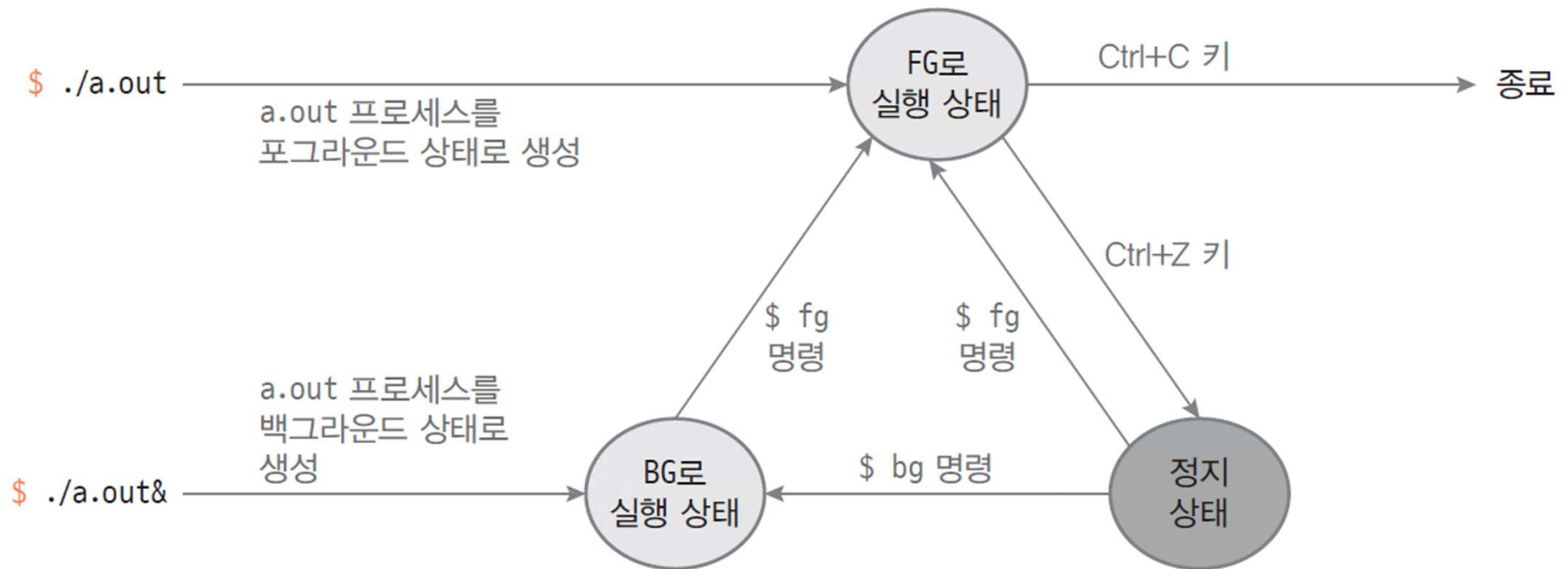
- 처음부터 백그라운드 상태로 실행시키기, & 기호

```
pi@pi:~/ch03 $ ./a.out&  
[1] 17022  
pi@pi:~/ch03 $ Hello  
Hello  
Hello  
Hello  
Hello
```

프로세스 관리 (5)

64

□ 프로세스 상태 변화



프로세스 상태 보기 (1)

65

□ ps

- ▣ ps는 가장 많이 사용되는 명령 중 하나이며 다양한 옵션이 있음
- ▣ 'ps'만 입력하면 현재 사용자가 현재 터미널에서 실행시킨 모든 프로세스 목록

명령	내용
ps	현 사용자가 현재 터미널에서 실행한 모든 프로세스 목록 출력
ps -ef	시스템 전체에 실행되는 모든 프로세스 목록 출력

```
pi@pi:~/ch03 $ ps
PID          TTY          TIME         CMD
4746         pts/0        00:00:02    bash
17089        pts/0        00:00:00    ps
pi@pi:~/ch03 $
```

프로세스 상태보기 (2)

66

□ ps -ef

- ▣ 현재 실행되는 모든 프로세스 목록 보기

```
pi@pi:~/ch03 $ ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	02:26	?	00:00:04	/sbin/init splash

.... 중간 생략함

root	2009	2	0	07:35	?	00:00:00	[kworker/0:2-events]
------	------	---	---	-------	---	----------	----------------------

root	2012	2	0	07:35	?	00:00:00	[kworker/1:0-mm_percpu_wq]
------	------	---	---	-------	---	----------	----------------------------

pi	2013	1349	0	07:36	pts/0	00:00:00	ps -ef
----	------	------	---	-------	-------	----------	--------

```
pi@pi:~/ch03 $
```

프로세스 강제 종료

67

□ kill

- ▣ kill 명령은 본래 프로세스 사이의 통신 방법 중 하나임
 - 한 프로세스가 다른 프로세스에게 알림을 보내는 것은 신호(signal)라고 하는데, kill은 신호를 보내는 명령
- ▣ kill 명령의 옵션중 -9는 프로세스에게 종료 신호를 보내는 것임
 - 결과적으로 프로세스를 종료시키게 됨

```
pi@pi:~/ch03 $ ./a.out
```

```
Hello
```

```
Hello
```

```
^Z
```

```
[1]+  멈춤 ./a.out
```

```
pi@pi:~/ch03 $ ps
```

```
PID TTY TIME CMD
```

```
1239 pts/0 00:00:00 bash
```

```
2007 pts/0 00:00:00 a.out
```

```
2026 pts/0 00:00:00 ps
```

```
pi@pi:~/ch03 $
```

```
pi@pi:~/ch03 $ kill -9 2026
```

```
-bash: kill: (2026) - 그런 프로세스가 없음
```

```
pi@pi:~/ch03 $
```


ps를 실행 후 자동
종료되었기 때문

```
pi@pi:~/ch03 $ kill -9 2007
```

```
[1]+  죽었음 ./a.out
```

```
pi@pi:~/ch03 $
```

2007번 프로세스 강
제 종료 시킴



유용한 명령

파일에서 텍스트 검색, grep

69

□ grep

- ▣ 파일들에 대해 '문자열' 검색하여 라인들 출력
- ▣ 사례
 - \$ grep in hello.c – hello.c 파일에서 "in" 문자열 검색

```
pi@pi:~/ch03 $ grep in hello.c
#include <stdio.h>
#include <unistd.h>
int main( ){
    printf("Hello\n");
pi@pi:~/ch03 $
```

- \$ grep -n main *.c – 현재 디렉터리에서 .c의 이름을 가진 모든 파일을 대상으로 "main" 문자열 검색

```
pi@pi:~/ch03 $ grep -n main *.c
hello2.c:4:int main( ){
hello.c:4:int main( ){
pi@pi:~/ch03 $
```

□ 파이프란?

- ▣ 한 프로그램의 출력 내용을 다른 프로그램의 입력으로 연결
- ▣ '|' 기호 사용
- ▣ 사용 사례

```
pi@pi:~/ch03 $ cat /etc/passwd | grep pi
pi:x:1000:1000:,,,:/home/pi:/bin/bash
pi@pi:~/ch03 $
```

```
pi@pi:~/ch03 $ ps -ef | grep bash
pi  795  598  0 02:26 tty1    00:00:00 -bash
pi 1349 1348  0 02:29 pts/0    00:00:00 -bash
pi 2044 1349  0 08:01 pts/0    00:00:00 grep --color=auto bash
pi@pi:~/ch03 $
```

history 명령

71

- 셸은 사용자가 입력한 명령들에 번호를 붙여 보관
- history 명령
 - ▣ 지금까지 셸에서 입력한 명령들을 시간 순으로, 번호를 붙여 출력
 - 히스토리 번호는 해당 명령을 소환할 때 활용됨

```
pi@pi:~/ch03 $ history
... 앞부분 생략
1326 grep in hello.c
1327 grep -n main *.c
1328 cat /etc/passwd | grep pi
1329 ps -ef | grep bash
1330 history
pi@pi:~/ch03 $
```

히스토리 관련 명령

72

□ ! 기호 활용

! 기호 활용 명령	내용
!히스토리번호	히스토리 번호의 명령 실행
!문자열	문자열로 시작하는 가장 최근에 실행한 명령 실행
!!	바로 이전에 실행한 명령 재실행

```
pi@pi:~/ch03 $ !1329
ps -ef | grep bash
pi 665 557 0 16:04 tty1 00:00:00 -bash
pi 1327 1326 0 16:27 pts/0 00:00:00 -bash
pi 1433 1327 0 16:52 pts/0 00:00:00 grep --color=auto bash
pi@pi:~/ch03 $ !p
ps -ef | grep bash
pi 665 557 0 16:04 tty1 00:00:00 -bash
pi 1327 1326 0 16:27 pts/0 00:00:00 -bash
pi 1436 1327 0 16:53 pts/0 00:00:00 grep --color=auto bash
pi@pi:~/ch03 $ !!
ps -ef | grep bash
pi 665 557 0 16:04 tty1 00:00:00 -bash
pi 1327 1326 0 16:27 pts/0 00:00:00 -bash
pi 1438 1327 0 16:53 pts/0 00:00:00 grep --color=auto bash
pi@pi:~/ch03 $
```


디렉터리 구조 보기

73

□ tree

- ▣ 디렉터리와 그 하위 모든 서브디렉터리에 만들어진 파일들을 트리 모양으로 출력

```
pi@pi:~/ch03 $ tree
```

```
.  
├── a.out  
├── hello.c  
├── hello2.c  
└── test  
    ├── a.out  
    ├── hello.c  
    └── hello2.c
```

```
1 directory, 6 files
```

```
pi@pi:~/ch03 $
```

소프트웨어 관리, apt-get

74

□ apt-get

- ▣ 리눅스에서 패키지를 설치, 제거 및 업데이트할 수 있는 명령

명령	설명
<code>sudo apt-get update</code>	설치 가능한 패키지 리스트를 업데이트
<code>sudo apt-get upgrade</code>	apt-get update로 가져온 각 패키지의 최신 버전에 맞게 업그레이드
<code>sudo apt-get install mosquitto</code>	mosquitto 소프트웨어를 설치
<code>sudo apt-get remove mosquitto</code>	설치된 mosquitto를 제거하되 설정 정보 등은 남겨둠
<code>sudo apt-get purge mosquitto</code>	설치된 mosquitto를 제거하되 설정 등 관련 정보 제거



환경 변수

□ 환경 변수란

- 컴퓨터의 이름, 로그인한 사용자의 이름, 실행 가능한 파일을 검색한 경로명, 사용자의 홈 디렉터리 등 다양한 정보를 저장하는 변수
- 현재 프로세스의 환경변수는 자식 프로세스에게 복사되어 전달
- 부모 프로세스가 자식 프로세스에게 정보를 전달하기 위해 사용

□ 환경 변수 저장

- "환경변수이름=값"의 형태로 저장

환경변수 보기 (1)

77

- bash 쉘에서 환경변수 보는 여러 방법
 - ▣ \$ printenv – 전체 환경변수 출력
 - ▣ \$ printenv HOME – 환경변수 HOME의 값 출력
 - ▣ \$ echo \$HOME – 환경변수 HOME의 값 출력
 - ▣ \$ env – 전체 환경변수 출력
- 전체 환경변수 출력

```
pi@pi:~/ch03 $ printenv
SHELL=/bin/bash
NO_AT_BRIDGE=1
PWD=/home/pi/ch03
LOGNAME=pi
...
TEXTDOMAIN=Linux-PAM
_=/usr/bin/printenv
pi@pi:~/ch03 $
```

개별 환경변수 값 보기

78

- ▣ 환경변수 이름으로 값을 볼 수 있음

```
pi@pi:~/ch03 $ printenv HOME  
/home/pi  
pi@pi:~/ch03 $
```

PATH 환경변수

79

□ PATH 환경변수

- ▣ 환경 변수 중에 가장 많이 사용, 중요한 환경 변수
- ▣ 실행가능한 프로그램들이 있는 디렉터리(path)들을 값으로 가짐
 - 각 디렉터리는 ':'으로 분리되어 있음

```
pi@pi:~/ch03 $ echo $PATH
/home/pi/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr
/local/games:/usr/games
pi@pi:~/ch03 $
```

- 셸은 사용자가 입력한 명령을 PATH 환경 변수에 지정된 디렉터리에
서만 찾아서 실행함

셸이 ls 명령을 실행하는 과정

80

□ 셸이 ls 명령을 실행하는 과정

```
pi@pi:~/ch03 $ ls
```

- 셸은 ls 프로그램을 실행시키기 위해 PATH 환경변수에 저장된 경로명들을 순서대로 검색
 - 처음에, '/home/pi/.local/bin' 디렉터리에서 찾으나 없음
 - 다음에, '/usr/local/sbin' 디렉터리에서 찾으나 없음
 - 다음에, '/usr/local/bin' 디렉터리에서 찾으나 없음
 - 다음에, '/usr/sbin' 디렉터리에서 찾으나 없음
 - '/usr/bin' 디렉터리에서 찾음
 - 셸은 ls 프로그램(ls 명령) 실행

a.out 프로그램을 찾는데 실패하는 경우

81

□ a.out 프로그램을 찾는데 실패하는 경우

```
pi@pi:~/ch03 $ ls  
a.out hello.c hello2.c test  
pi@pi:~/ch03 $
```

```
pi@pi:~/ch03 $ a.out  
-bash: a.out: 명령어를 찾을 수 없음  
pi@pi:~/ch03 $
```

- ch03 디렉터리에 a.out 명령이 있지만,
- PATH 환경 변수에 지정된 디렉터리들 내에 a.out이 없기 때문

a.out 프로그램을 찾는데 성공하는 경우

82

□ a.out 프로그램을 찾는데 성공하는 경우

▣ 방법1: 경로를 명시적으로 주기

- 다음과 같이 ./a.out을 사용하여 경로 명시

```
pi@pi:~/ch03 $ ./a.out
Hello
Hello
^C
pi@pi:~/ch03 $
```

▣ 방법2: PATH 환경변수에 a.out 경로 등록

```
pi@pi:~/ch03 $ export PATH=.:$PATH
pi@pi:~/ch03 $ echo $PATH
./home/pi/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games
pi@pi:~/ch03 $
```

```
pi@pi:~/ch03 $ a.out
Hello
Hello
^C
pi@pi:~/ch03 $
```

수정한 PATH 환경변수를 영구 유지 방법

83

- .profile 파일
 - ▣ 사용자가 로그인할 때 반드시 실행되는 스크립트 파일
 - ▣ 홈 디렉터리에 만들어두면 됨
 - ▣ 현재 리눅스에서 자동으로 생성
- .profile 파일에 PATH를 설정하는 명령 작성

```
pi@pi:~ $ cd ~
pi@pi:~ $ cat .profile
# 앞 부분 생략
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
PATH="$HOME/.local/bin:$PATH"
fi
export PATH=.:$PATH # PATH에 .를 맨 앞에 붙여 새로 만듦
pi@pi:~ $
```

- 다시 로그인을 하거나 source 실행 실행

```
pi@pi:~$ source .profile
```

- ▣ PATH 환경 변수에 현재 디렉터리(.)이 저장됨