HALLYM UNIVERSITY

Operating Systems LAB 3

Wonpyo Kim

skykwp@gmail.com

Outline

- Substance
  - Basic Linux Command
  - Make

Linux Command

- **man (an interface to the on-line reference manuals)**
  - man 은 명령어에 대한 다양한 옵션과 기본적인 소개를 위해 온라인 메뉴얼 페이지를 보여준다
  - 이 명령어로 다른 명령어의 기능과 사용법을 이해할 수 있다

```
kwp@A1409-OSLAB-01:~$ man poweroff
```

  - 위와 같이 **man [명령어]** 를 입력하면 아래와 같은 메뉴얼 페이지를 볼 수 있다

```
HALT(8)                                                    halt                                                    HALT(8)

NAME
       halt, poweroff, reboot - Halt, power-off or reboot the machine

SYNOPSIS
       halt [OPTIONS...]

       poweroff [OPTIONS...]

       reboot [OPTIONS...]

DESCRIPTION
       halt, poweroff, reboot may be used to halt, power-off or reboot the machine.

OPTIONS
       The following options are understood:

       --help
           Print a short help text and exit.

       --halt
           Halt the machine, regardless of which one of the three commands is invoked.

       -p, --poweroff
           Power-off the machine, regardless of which one of the three commands is invoked.
```

Linux Command

- ls (list directory contents)
  - ls 는 현재 디렉토리의 내용을 보여준다
  - 옵션을 사용하면, 자세한 내용을 출력할 수도 있다

```
[kwp@A1409-OSLAB-01:~$ ls -l
total 12
drwxr-xr-x 10 kwp kwp 4096  3월   4 14:57 Desktop
drwxrwxr-x  2 kwp kwp 4096  3월   8 10:58 script
drwxrwxr-x  6 kwp kwp 4096  3월  14 13:11 toy
```

-

옵션	설명
-l	파일크기, 날짜/시간, 허가권, 소유권, 그룹 등의 자세한 디렉토리의 내용을 출력준다
-a	숨겨진 파일을 포함해서 디렉터리 내의 모든 파일을 보여준다. 리눅스에서 마침표로 시작하는 파일은 숨김파일이며, 이 옵션에서는 숨김파일까지 출력준다
-1	현재 위치한 디렉터리의 범위에서 파일과 디렉터리의 이름만 출력준다
-R	R은 Recursive의 약자로 현재 디렉토리에 포함되는 하위 디렉토리의 정보를 재귀순환하며 모두 출력한다

Linux Command

- **pwd (print working directory)**
  - pwd 는 현재 작업중인 디렉토리의 경로를 출력한다

```
kwp@A1409-OSLAB-01:~$ pwd
/home/kwp
```

- **cd**
  - 작업할 디렉토리를 변경할 때 사용한다.

```
kwp@A1409-OSLAB-01:~$ ls
Desktop  script  toy
kwp@A1409-OSLAB-01:~$ cd script
kwp@A1409-OSLAB-01:~/script$ pwd
/home/kwp/script
```

Linux Command

- **mkdir (make directories)**
  - 만들고자하는 디렉토리가 존재하지 않는다면, 디렉토리를 생성한다

```
[kwp@A1409-OSLAB-01:~$ mkdir basic
[kwp@A1409-OSLAB-01:~$ ls
 basic  Desktop  script  toy
```

- **rmdir (remove empty directories)**
  - mkdir의 반대로 비어있는 디렉토리를 제거할 때 사용한다

```
[kwp@A1409-OSLAB-01:~$ ls
 basic  Desktop  script  toy
[kwp@A1409-OSLAB-01:~$ rmdir basic
[kwp@A1409-OSLAB-01:~$ ls
 Desktop  script  toy
```

Linux Command

- **find (search for files in a directory hierarchy)**
  - 특정 파일이나 디렉토리를 계층구조 검색을 통해 찾을 때 사용한다.

```
kwp@A1409-OSLAB-01:~$ find basic
basic
```

  - -atime, -size 등의 옵션을 이용할 수 있다.

- **locate (find files by name)**
  - find와 흡사한 명령으로 파일을 찾을 때 사용한다.

Linux Command

- **mv (move or rename files)**
  - 파일이나 디렉토리를 이동하거나 이름을 변경할 수 있다

```
[kwp@A1409-OSLAB-01:~$ mv basic basic_moved
[kwp@A1409-OSLAB-01:~$ ls
 basic_moved  Desktop  script  toy
[kwp@A1409-OSLAB-01:~$ mv basic_moved/ toy
[kwp@A1409-OSLAB-01:~$ ls ./toy
 basic_moved  lab1  lab2  lab3  test
```

- **cp (copy files and directories)**
  - 파일이나 디렉토리의 복사본을 생성한다.

```
[kwp@A1409-OSLAB-01:~$ cp -r basic_moved basic
[kwp@A1409-OSLAB-01:~$ ls
 basic  basic_moved  Desktop  script  toy
```

Linux Command

- **more (file perusal filter for crt viewing)**
  - 긴 내용 (콘솔 출력을 넘어가는)을 한 번에 한 화면씩 내용을 출력한다
  - more <filename> 으로 사용할 수 있다.

- **head, tail**
  - 파일의 맨 처음 부분이나 끝 부분을 볼 수 있다
  - head or tail -n <filename> 처음이나 마지막의 n줄 만큼 출력한다

Linux Command

- **rm (remove files and directories)**
  - 파일이나 디렉토리를 삭제한다

    - -r : 디렉토리와 하위 파일까지 함께 삭제
    - -i : 사용자에게 정말로 삭제할 것인지 물어본다

```
kwp@A1409-OSLAB-01:~$ rm -ri basic
rm: remove directory 'basic'? y
kwp@A1409-OSLAB-01:~$ ls
basic_moved  Desktop  more.txt  script  toy
```

Linux Command

- **ln (make links between files)**
  - 원본 파일을 가리키는 링크파일을 생성한다 (바로가기 개념)

  - 링크의 종류
    - 하드 링크
      - 원래의 파일과 동일한 i-node를 갖는다
      - i-node는 파일의 속성과 디스크 상의 위치에 대한 정보를 갖고있는 커널 구조체

- 심볼릭 링크
  - 링크 파일 이름이 원본파일을 가리킨다
  - 다른 파티션 뿐만 아니라 다른 네트워크 장치 상에 있는 파일도 링크가 가능하다

```
kwp@A1409-OSLAB-01:~/toy$ ln -s ../basic basic_link
kwp@A1409-OSLAB-01:~/toy$ ls -al
total 24
drwxrwxr-x  6 kwp kwp 4096  3월  14 14:23 .
drwx------ 14 kwp kwp 4096  3월  14 14:22 ..
lrwxrwxrwx  1 kwp kwp    8  3월  14 14:23 basic_link -> ../basic
drwxrwxr-x  2 kwp kwp 4096  3월   8 10:55 lab1
drwxrwxr-x  3 kwp kwp 4096  3월   9 19:34 lab2
drwxrwxr-x  2 kwp kwp 4096  3월  14 13:10 lab3
drwxrwxr-x  2 kwp kwp 4096  3월  14 13:11 test
```

Linux Command

- **kill (send a signal to process)**
  - 어플리케이션 (프로세스)을 강제로 종료하고 싶을 때 사용할 수 있다
  - 보통 이 명령어는 ps 명령어 다음에 사용한다. 즉, ps 명령어로 죽이고 싶은 프로세스의 ID를 확인해야 한다

```
kwp@A1409-OSLAB-01:~$ ps
  PID TTY          TIME CMD
18967 pts/1    00:00:00 bash
19880 pts/1    00:00:02 infinite_loop
19882 pts/1    00:00:00 ps
kwp@A1409-OSLAB-01:~$ kill -9 19880
kwp@A1409-OSLAB-01:~$ ps
  PID TTY          TIME CMD
18967 pts/1    00:00:00 bash
19883 pts/1    00:00:00 ps
[1]+  Killed                  ./infinite_loop  (wd: ~/basic)
(wd now: ~)
```

Linux Command

- **sudo (execute a command as another user)**
  - 리눅스의 장점 중 하나는 루트 사용자만이 어플리케이션의 설치나 삭제 및 시스템의 중대한 변경을 할 수 있다
  - 일반 사용자가 어플리케이션을 설치해야 한다면, sudo 명령으로 일시적으로 루트 권한을 부여할 수 있다
  - 하지만, 루트의 비밀번호를 알고 있어야 한다

```
kwp@A1409-OSLAB-01:~$ useradd dummy
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
kwp@A1409-OSLAB-01:~$ sudo useradd dummy
```

Linux Command

- chown (change file owner and group)
  - 파일의 소유권을 다른 사용자로 변경할 수 있다
  - 루트 사용자만이 이 명령을 사용할 수 있다

```
[kwp@A1409-OSLAB-01:~$ chown s_guest basic
chown: changing ownership of 'basic': Operation not permitted
```

- -R 옵션을 부여하면, 지정된 디렉토리의 하위 파일까지 모두 재귀순환으로 변경한다

```
[kwp@A1409-OSLAB-01:~$ chown -R s_guest basic
chown: changing ownership of 'basic/infinite_loop.c': Operation not permitted
chown: changing ownership of 'basic/infinite_loop': Operation not permitted
chown: changing ownership of 'basic': Operation not permitted
```

Linux Command

- **chmod (change file mode bits)**
  - 파일이나 디렉토리의 허가권을 변경한다
  - 허가권은 다음과 같이 4개의 부분으로 나뉜다
    - **파일타입**, **소유자**, **그룹**, **기타 사용자**

```
[kwp@A1409-OSLAB-01:~$ ls -al
 total 140
drwx------ 14 kwp  kwp   4096  3월  14 14:27 .
drwx--x--x 44 root root  4096  3월  14 14:30 ..
```

- 기호로 표시한 예

문자	허가권	값(8진수)
---	허가권 없음	0
r--	읽기만 가능	4
rw-	읽기/쓰기 가능	6
rwx	읽기/쓰기/실행 가능	7
r-x	읽기/실행 가능	5
--x	실행만 가능	1

문자	허가권	값(8진수)
r	읽기(READ)	4
w	쓰기(WRITE)	2
x	실행(EXECUTE)	1

Linux Command

- **chmod (change file mode bits)**

허가권	숫자 값	설명
-rw-------	600	소유자에게만 읽기/쓰기 권한이 있다. (대부분의 파일)
-rw-r--r--	644	소유자에게 읽기/쓰기 권한이 있고, 그룹과 기타 사용자는 읽기 권한만 있다.
-rw-rw-rw-	666	모든 사용자들에게 읽기/쓰기 권한을 부여한다. (보안 취약)
-rwx------	700	소유자가 읽기/쓰기/실행 권한이 있다. 소유자가 실행하려는 프로그램 파일에 사용한다. (C, C++ 파일 등)
-rwxr-xr-x	755	700과 소유자는 권한이 동일하며, 다른 사용자는 읽기/실행 권한이 있다.
-rwxrwxrwx	777	모든 사람이 읽기/쓰기/실행 권한이 있다. (보안 취약)
-rwx--x--x	711	소유자가 읽기/쓰기/실행 권한이 있다. 다른 사용자는 실행권한만 있다.
drwx------	700	mkdir로 디렉토리를 생성할 때 기본 권한이다. 오직 소유자만 진입할 수 있다.
drwxr-xr-x	755	이 디렉토리는 소유자에 의해서만 내용이 변경될 수 있으나, 다른 사용자가 접근해서 내용을 읽거나 실행할 수 있다.
drwx--x--x	711	모든 사용자가 이 디렉토리에 진입할 수 있지만, 기타 사용자는 실행만 가능하다. 즉, ls 명령이 수행될 수 없다.

Linux Command

- chmod (change file mode bits)
  - 다음과 같이 수행할 수 있다

```
[kwp@A1409-OSLAB-01:~$ ls -l
total 28
drwxrwxr-x  2 kwp kwp 4096  3월  14 14:27 basic
drwxrwxr-x  2 kwp kwp 4096  3월  14 14:01 basic_moved
drwxr-xr-x 10 kwp kwp 4096  3월   4 14:57 Desktop
-rw-rw-r--  1 kwp kwp 5171  3월  14 14:14 file
drwxrwxr-x  2 kwp kwp 4096  3월   8 10:58 script
drwxrwxr-x  6 kwp kwp 4096  3월  14 14:23 toy
[kwp@A1409-OSLAB-01:~$ chmod 000 file
[kwp@A1409-OSLAB-01:~$ ls -l
total 28
drwxrwxr-x  2 kwp kwp 4096  3월  14 14:27 basic
drwxrwxr-x  2 kwp kwp 4096  3월  14 14:01 basic_moved
drwxr-xr-x 10 kwp kwp 4096  3월   4 14:57 Desktop
----------  1 kwp kwp 5171  3월  14 14:14 file
drwxrwxr-x  2 kwp kwp 4096  3월   8 10:58 script
drwxrwxr-x  6 kwp kwp 4096  3월  14 14:23 toy
```

Outline

- Substance
  - Basic Linux Command
  - make

make

- make 유틸리티
  - make는 프로그램 그룹을 유지하는데에 필요한 유틸리티이다
  - make의 주된 목적은 프로그램 그룹 중에서 어떤 부분이 새롭게 컴파일되어야 하는지를 자동으로 판단해서 필요한 커맨드를 실행하여 재컴파일 하는 것이다
  - 다음과 같은 경우에 make를 사용하면 편리하다
    - 입력 파일이 바뀌면(수정) 자동적으로 결과파일이 바뀌기를 원할 때, 지능적으로 일을 수행하길 바랄때
    - LaTeX 파일처럼 자동적으로 프로그램이 수행되기를 바랄때
- 우분투에서 사용하는 GNU make는 보통 GNUmakefile, Makefile, makefile 중에 한 개의 파일만 있으면 그 파일을 읽게 된다. 일반적으로는 Makefile을 추천한다. makefile은 다른 소스파일에 묻혀 잘 보이지 않을 수 있다
- Makefile은 make가 이해하도록 쉘 스크립트 언어처럼 되어있다.

Makefile

- Makefile 의 기본 구조
  - Makefile 은 아래와 같이 목표(target), 의존관계(dependency), 명령(command)의 세 가지로 이루어진 규칙이 계속적으로 나열되어 있다. make가 지능적으로 파일을 갱신하는 것도 이 규칙을 따르기 때문이다.

```
target ... :  dependency ...
              command
              ...
              ...
```

  - 여기서 target 은 command 가 수행되어진 결과 파일을 지정하는 것이다. 목적파일(Object file)이나 실행파일이 된다.

  - command 부분에 정의된 명령들은 dependency 부분에 정의된 파일의 내용이 바뀌었거나, target 에 해당하는 파일이 없을 때 차례대로 실행된다. 코딩은 일반적으로 쉘에서 쓸 수 있는 모든 명령어들을 사용할 수 있다.

  - 명령 부분의 시작은 꼭 [TAB] 으로 시작해야 한다. 그냥 빈칸(Space) 등을 사용하면 에러가 발생한다.

Makefile

- Makefile 의 기본 구조
  - 리눅스 커널의 Makefile 의 일부분 예시

```
222 archscripts: scripts_basic
223         $(Q)$(MAKE) $(build)=arch/x86/tools relocs
224
225 ###
226 # Syscall table generation
227
228 archheaders:
229         $(Q)$(MAKE) $(build)=arch/x86/entry/syscalls all
230
231 ###
232 # Kernel objects
233
234 head-y := arch/x86/kernel/head_$(BITS).o
235 head-y += arch/x86/kernel/head$(BITS).o
236 head-y += arch/x86/kernel/ebda.o
237 head-y += arch/x86/kernel/platform-quirks.o
```

Makefile

- Makefile 예제
  - Lab 2 에서 다음과 같이 파일 6개를 생성했었다. 그것을 모두 컴파일 하기 위해선 다음과 같이 수행했다.

```
$ gcc -c main.c
$ gcc -c get_data.c
$ gcc -c smallest.c
$ gcc -c select_sort.c
$ gcc -c print_data.c
$ gcc -c exchange.c
$ gcc main.o get_data.o smallest.o select_sort.o print_data.o exchange.o -o lab02_kwp
```

- 하지만, 이런 작업은 고된 노동이 아닐 수 없다. 만약 파일이 20개 쯤 된다면 ?

- 이를 Makefile을 생성하고 make를 사용함으로써 해결할 수 있다.

Makefile

- Makefile 생성
  - 이전에 작성한 6개의 c 파일이 있는 디렉토리에서 vi Makefile 을 실행하여 편집기로 들어간다.
  - 다음과 같이 내용을 채워준다. 명령 라인의 시작은 [TAB] 이라는 것을 유념한다.

```
1
2 lab02_kwp : main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
3     gcc -o lab02_kwp main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
4
5 main.o : main.c
6     gcc -c main.c
7 exchange.o : exchange.c
8     gcc -c exchange.c
9 get_data.o : get_data.c
10    gcc -c get_data.c
11 smallest.o : smallest.c
12    gcc -c smallest.c
13 select_sort.o : select_sort.c
14    gcc -c select_sort.c
15 print_data.o : print_data.c
16    gcc -c print_data.c
17
```

Makefile

- Makefile 테스트
  - 저장을 하고 쉘에서 make 를 입력한다. 다음과 같은 출력을 보이면 성공이다.

```
kwp@A1409-OSLAB-01:~/toy/lab3$ make
gcc -c main.c
gcc -c exchange.c
gcc -c get_data.c
gcc -c smallest.c
gcc -c select_sort.c
gcc -c print_data.c
gcc -o lab02_kwp main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
```

Makefile

- Makefile 매크로
  - 앞선 Makefile 내용에서는 6개의 오브젝트 파일을 2번 입력해야 하는 번거로움이 존재했다.

```
2 lab02_kwp : main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
3     gcc -o lab02_kwp main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
```

  - 간단한 매크로를 추가하여 이를 해결할 수 있다. Makefile 의 처음 부분을 다음과 같이 수정한다.

```
2 OBJECTS = main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
3
4 lab02_kwp : $(OBJECTS)
5     gcc -o lab02_kwp $(OBJECTS)
6
7 main.o : main.c
8     gcc -c main.c
9 exchange.o : exchange.c
```

- 매크로를 사용하면 더욱 편리한 Makefile 을 만들 수 있다. 매크로를 선언하고 명령이나 의존 구역에서의 사용은 $(macro_name) 으로 사용할 수 있다.

Makefile

- Makefile 매크로
  - make 를 다시 수행하면 다음과 같이 나타난다.

```
[kwp@A1409-OSLAB-01:~/toy/lab3$ make
make: 'lab02_kwp' is up to date.
```

  - 이유는 Makefile 은 수정되었지만, 파일 내용은 그대로이기 때문이다. 새롭게 오브젝트 파일을 생성하고 연결하기 위해서는 오브젝트 파일을 제거해야한다.

- 이 때는 레이블을 추가하면 된다. 이번에는 목적파일 (object)을 삭제하는 clean 명령을 추가해본다.

```
17 print_data.o : print_data.c
18     gcc -c print_data.c
19
20 clean :
21     rm $(OBJECTS)
```

- Makefile 하단 부분에 clean 명령을 추가해준다.

Makefile

- Makefile 레이블
  - 방금 추가한 clean 레이블을 다음과 같이 실행할 수 있다.

```
[kwp@A1409-OSLAB-01:~/toy/lab3$ make clean
 rm main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
[kwp@A1409-OSLAB-01:~/toy/lab3$ make
 gcc -c main.c
 gcc -c exchange.c
 gcc -c get_data.c
 gcc -c smallest.c
 gcc -c select_sort.c
 gcc -c print_data.c
 gcc -o lab02_kwp main.o exchange.o get_data.o smallest.o select_sort.o print_data.o
```

- 오브젝트 파일이 삭제되어 make가 다시 수행되는 것을 볼 수 있다.

- clean은 오브젝트 파일이 삭제되는 것이 아니다. clean 에 등록된 command 를 수행할 뿐이다.