

System Programming & OS 실습

5. Shell Script

최민국, 정지현, 안석현, 김선재

Dankook University

{mgchoi, wlgjsjames7224, seokhyun, rlatjswo0824}@dankook.ac.kr

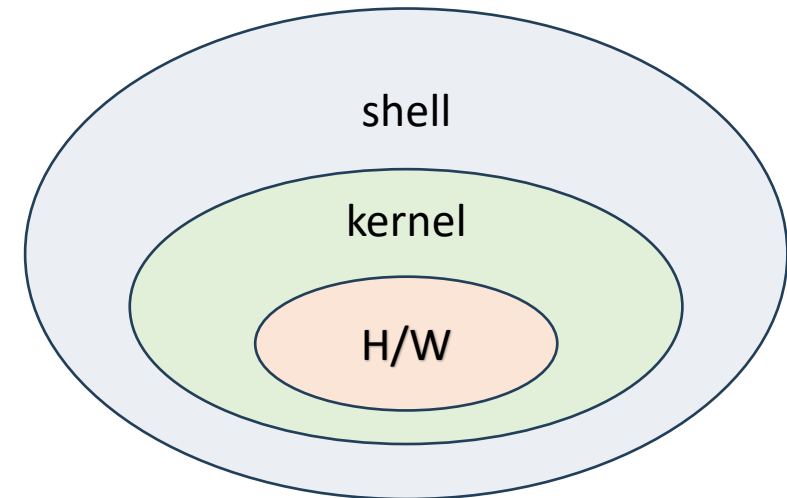
❖ Shell Script

- Shell Script 기초 문법
- 실습

❖ 리눅스 셸



User

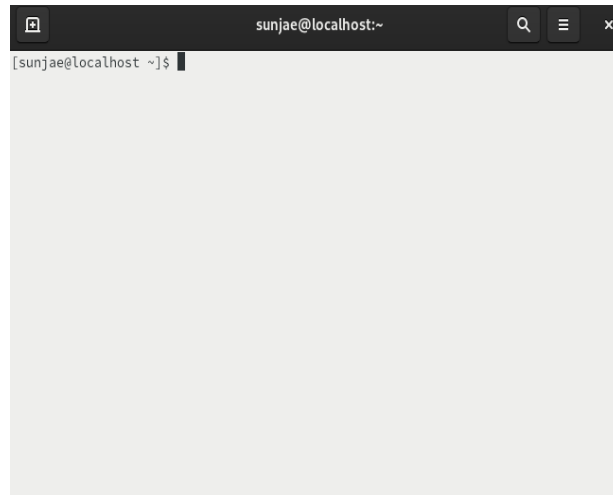


Linux system

❖ 터미널을 통한 셸의 사용



User



Terminal program

/dev/pts...



bash

Shell Script

❖ Shell Script

- 유닉스(Unix), 리눅스(Linux) 등의 운영체제에서 사용되는 스크립트 언어
- 일반적인 명령어들을 모아서 하나의 파일로 작성한 것
-> 명령어들을 순차적으로 실행시켜주는 인터프리터
- 반복적인 작업을 자동화하거나 여러 명령어를 순차적으로 실행하기 위해 사용



Shell Script

❖ Shell Script 기초 문법

- 제일 상단에 `#!/bin/bash` 입력
 - `#!` : 쉼뱅(shebang)으로 스크립트가 실행될 때 사용할 셸 지정
 - `/bin/bash` : bin 하위 폴더에 bash 셸로 실행 지정
- `#!/bin/bash` vs `#!/bin/sh`
 - bash : sh의 개선된 버전으로 기능이 많고 확장성이 뛰어남
 - sh : 가장 기본적인 셸로 최소한의 기능만 제공

bash	sh
<code>#!/bin/bash</code>	<code>#!/bin/sh</code>
더 많은 기능	최소한의 기능
작업 제어 지원	작업 제어 지원하지 않음
유효한 POSIX 셸이 아님	유효한 POSIX 셸
사용하기 편리	bash에 비해 사용하기 어려움
확장된 언어	오리지널 언어

Shell Script

❖ Shell Script 기초 문법

- echo : 문자열을 컴퓨터 터미널에 출력하는 명령어

```
[ec2-user@ip-172-31-8-194 ~]$ vim myshell.sh
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash

echo "hello, world"
```

- vim myshell.sh

```
#!/bin/bash
```

```
echo "hello, world"
```

- bash myshell.sh
- chmod +x myshell.sh

-> execute 권한 추가

- ./myshell.sh

```
[ec2-user@ip-172-31-8-194 ~]$ chmod +x myshell.sh
[ec2-user@ip-172-31-8-194 ~]$ ls
cscope.files  cscope.out  day5  myshell.sh  test2.c  test.c  test.c~
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
hello, world
```

Shell Script

❖ Shell Script 기초 문법

- 변수 사용
 - 변수 선언 : `language="hello world"`
 - 변수 사용 : `$language`

`#!/bin/bash`

`language="hello world"`

`echo "$language"`

```
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash
language="hello world"
echo "$language"
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
hello world
```


Shell Script

❖ Shell Script 기초 문법

- function 이용

```
#!/bin/bash
```

```
function Taba() {
```

```
    echo "hello, Taba"
```

```
    echo $1
```

```
}
```

```
Taba "hello, $1 print"
```

```
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash

function Taba() {
    echo "hello, Taba"
    echo $1
}

Taba "hello, $1 print"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
hello, Taba
hello, print
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh test
hello, Taba
hello, test print
```

Shell Script

❖ Shell Script 기초 문법

- Local 키워드를 이용한 전역변수 & 지역변수

```
#!/bin/bash
```

```
str="str_hello"
```

```
function Taba() {
```

```
    local str2="str2"
```

```
    echo $str1
```

```
    echo $str2
```

```
}
```

```
Taba
```

```
echo "global $str"
```

```
echo "local $str2"
```

```
[ec2-user@ip-172-31-8-194 ~]$ cat myshell.sh
#!/bin/bash

str="str_hello"
function Taba() {
    local str2="str2"
    echo $str1
    echo $str2
}
Taba
echo "global $str"
echo "local $str2"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh
str_hello
str2
global str_hello
local
```

Shell Script

❖ Shell Script 기초 문법

- 예약 변수 & 환경 변수
 - 시스템에서 미리 정해둔 변수들이 존재

`#!/bin/bash`

`echo "Home : $HOME"`(사용자 홈 디렉토리)

`echo "Path : $PATH"`(실행파일 찾을 디렉토리 경로)

`echo "pwd : $PWD"`(현재 작업중인 디렉토리 경로)

`Echo "user : $USER"`(사용자 이름)

`Echo "OS type : $OSTYPE"`(운영체제 종류)

```
[ec2-user@ip-172-31-8-194 ~]$ cat system_v.sh
#!/bin/bash

echo "Home : $HOME"
echo "Path : $PATH"
echo "pwd : $PWD"
echo "user : $USER"
echo "OS type : $OSTYPE"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./system v.sh
Home : /home/ec2-user
Path : /home/ec2-user/.local/bin:/home/ec2-us
bin:/usr/local/sbin:/usr/sbin
pwd : /home/ec2-user
user : ec2-user
OS type : linux-gnu
```

Shell Script

❖ Shell Script 기초 문법

- 매개변수
 - \$1, \$2, \$3 ... \${10}
- 셸 스크립트는 특수변수라고 불리는 매개변수가 존재
- 사용자가 전달하는 변수나 함수의 인자를 전달하는데 사용

```
[ec2-user@ip-172-31-8-194 ~]$ $0./test.sh $11 $22 3 4 ... $nn
```

Shell Script

❖ Shell Script 기초 문법

- 매개변수를 이용한 예제
 - \$1, \$2, \$3 ... \${10}

- ./myshell.sh \$1 \$2 \$3 \$4 입력 시

Hello world(\$1+\$2) 와 더하기 값 출력

```
./myshell.sh $1 $2 $3 $4
```

```
first var : hello  
second var : world  
hello world  
result=7
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:=문자열}
- 변수가 설정되어있지 않거나 NULL 이라면
문자열로 치환

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh  
#!/bin/bash
```

```
var1="Tabal"  
var2=""
```

```
echo "print var1 : $var1"  
echo "print var2 : $var2"
```

```
echo ""  
echo "print var1 : ${var1:=Tabal test}"  
echo "print var2 : ${var2:=hello test}"
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
```

```
print var1 : Tabal  
print var2 :
```

```
print var1 : Tabal test  
print var2 : hello test
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:=문자열}
- 기본값 설정을 통해 출력
 - Username : 입력 값이 없을 경우 “guest”를 출력
 - Greeting : 입력 값이 없을 경우 “welcome”을 출력

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh  
username:Guest  
greeting: Welcome
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./myshell.sh  
username:seokhyun  
greeting: Taba education
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:?에러 메시지}
- 변수가 설정되어있지 않거나 NULL이라면 에러 메시지 출력 후 종료
- 에러메시지를 설정하지 않을경우
-> "Parameter null or not set" 출력

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

str1="str1"
str2=""
error_msg="sorry"

echo ${str1:?$error_msg}
echo ${str2:?$error_msg}

[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
str1
./str v.sh: line 8: str2: sorry
```


Shell Script

❖ Shell Script 기초 문법

- 그 외에 변환과 설명

변환	설명
<code>\${변수-문자열}</code>	변수가 설정되지 않은 경우, 문자열을 변수로 치환
<code>\${변수:-문자열}</code>	변수가 설정되지 않았거나 NULL인 경우 문자열을 변수로 치환
<code>\${변수=문자열}</code>	변수가 설정되지 않은 경우, 변수에 문자열을 저장하고 치환
<code>\${변수:=문자열}</code>	변수가 설정되지 않았거나 NULL인 경우 문자열을 변수에 저장하고 치환
<code>\${변수:?에러 메세지}</code>	변수가 설정되지 않았거나 NULL인 경우, 에러메세지 출력하고 종료
<code>\${변수:시작위치}</code>	변수값이 문자열인 경우, 시작 위치부터 문자열 길이 끝까지 출력
<code>\${변수:시작위치:길이}</code>	변수값이 문자열인 경우, 시작 위치부터 길이까지 출력

Shell Script

❖ Shell Script 기초 문법

- 문자열 패턴 및 변경

패턴, 변경	설명
\${변수#패턴}	변수 앞 에서부터 처음 찾은 패턴과 일치하는 패턴 앞의 모든 문자열 제거
\${변수##패턴}	변수 앞 에서부터 마지막 찾은 패턴과 일치하는 패턴 앞의 모든 문자열 제거
\${변수%패턴}	변수 뒤 에서부터 처음 찾은 패턴과 일치하는 패턴 뒤의 모든 문자열 제거
\${변수%%패턴}	변수 뒤 에서부터 마지막 찾은 패턴과 일치하는 패턴 뒤의 모든 문자열 제거
\${#변수}	변수의 길이
\${변수/찾는문자열/바꿀문자열}	변수에서 찾는 문자열을 바꿀 문자열로 변경, 없으면 삭제
\${변수/#찾을문자열/바꿀문자열}	변수에서 문자열에서 찾을 문자열 시작과 맞으면 문자열 변경
\${변수/%찾을문자열/바꿀문자열}	변수에서 문자열에서 찾을 문자열 마지막과 맞으면 문자열 변경

Shell Script

❖ Shell Script 기초 문법

- \${변수-문자열}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash
name=${USER-"unknown user"}
echo "hello, $name!"
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
hello, ec2-user!
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:시작위치}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash
str="hello my name is seokhyun"
echo "${str:6}"
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
my name is seokhyun
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:시작위치:길이}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

str="hello 6my 12name is seokhyun"

echo "${str:6:7}"

[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
my name
```

Shell Script

❖ Shell Script 기초 문법

- \${변수:시작위치:길이}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

str="hello 6my 12name is seokhyun"

echo "${str:6:7}"

[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
my name
```

Shell Script

❖ Shell Script 기초 문법

- \${변수#패턴}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

filename="test.txt"

echo "${filename#*.}"
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
txt
```

Shell Script

❖ Shell Script 기초 문법

- \${변수%패턴}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh
#!/bin/bash

filename="test.txt"

echo "${filename%.txt}"

[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
test
```


Shell Script

❖ Shell Script 기초 문법

- 그 외의 패턴

- \${변수/찾는문자열/바꿀문자열}
- \${변수//찾는문자열/바꿀문자열}
- \${변수/찾는문자열}
- \${변수//찾는문자열}
- \${변수/#찾을문자열/바꿀문자열}
- \${변수/%찾을문자열/바꿀문자열}

```
[ec2-user@ip-172-31-8-194 ~]$ cat str_v.sh  
#!/bin/bash
```

```
test_str="hello ann seok hyun hello ann"
```

```
echo ${test_str/hello/hi}
```

```
echo ${test_str//hello/hi}
```

```
echo ${test_str/hello}
```

```
echo ${test_str//hello}
```

```
echo ${test_str/#he/what?}
```

```
echo ${test_str/%lo/what??}
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./str_v.sh
```

```
hi ann seok hyun hello ann
```

```
hi ann seok hyun hi ann
```

```
ann seok hyun hello ann
```

```
ann seok hyun ann
```

```
what?llo ann seok hyun hello ann
```

```
hello ann seok hyun hello ann
```

Shell Script

❖ Shell Script 기초 문법

- 조건문

if [첫 번째 조건식] then

수행문

Elif [두 번째 조건식] then

수행문

Else

수행문

fi

```
[ec2-user@ip-172-31-8-194 ~]$ cat if_case.sh
#!/bin/bash
v1=10
v2=10

if [ $v1 = $v2 ]
then
    echo True
else
    echo False
fi

[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh
True
```

Shell Script

❖ Shell Script 기초 문법

• 조건문

- 사용자가 입력한 2 개의 숫자를 비교하는 스크립트 작성

- 스크립트는 2개의 숫자를 매개변수로 받는다.

- 입력된 2개의 숫자를 조건에 따라 결과 출력

- $A = B$: 두 숫자는 같습니다.
- $A > B$: 첫 번째 숫자가 더 큼니다.
- $A < B$: 두 번째 숫자가 더 큼니다.

- `-eq` : 2 개의 값이 같으면

- `-gt` : 값1이 값 2보다 크면 참

- `-lt` : 값 1이 값2보다 작으면 참

```
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh 1 3
v1 < v2
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh 1 1
v1 = v2
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh 3 1
v1 > v2
```

Shell Script

❖ Shell Script 기초 문법

- 조건문

case \$변수 in

조건값 1)

수행문 ;;

조건값 2)

수행문 ;;

*) //조건 1, 조건 2외

수행문

esac

```
[ec2-user@ip-172-31-8-194 ~]$ cat if_case.sh
```

```
#!/bin/bash
```

```
case $1 in
    hi)
        echo hi
        ;;
    hello)
        echo hello
        ;;
    *)
esac
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh hi
```

```
hi
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./if_case.sh hello
```

```
hello
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
#!/bin/bash

for num in 1 2 3 4 5
do
    echo $num;
done
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
1
2
3
4
5
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
for file in $HOME/*
```

```
do
```

```
    echo $file;
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
/home/ec2-user/cscope.files
```

```
/home/ec2-user/cscope.out
```

```
/home/ec2-user/day5
```

```
/home/ec2-user/if_case.sh
```

```
/home/ec2-user/iter.sh
```

```
/home/ec2-user/myshell.sh
```

```
/home/ec2-user/str_v.sh
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

- {시작..끝..증가값}

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
for num in {1..10..2}
```

```
do
```

```
    echo $num;
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
1
```

```
3
```

```
5
```

```
7
```

```
9
```

Shell Script

❖ Shell Script 기초 문법

- 반복문

for 변수 in [범위]

do

반복할 수행문

done

- 배열, 리스트 사용 둘 다 동일

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
arr=("a" "b" "c" "d" "e")
```

```
for str in ${arr[@]}
```

```
do
```

```
    echo $str;
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
a
```

```
b
```

```
c
```

```
d
```

```
e
```


Shell Script

❖ Shell Script 기초 문법

- 반복문

While [\$변수1 연산자 \$변수2]

do

수행문

done

```
[ec2-user@ip-172-31-8-194 ~]$ cat iter.sh
```

```
#!/bin/bash
```

```
n=1
```

```
while [ $n -lt 5 ]
```

```
do
```

```
    echo $n
```

```
    n=$((n+1))
```

```
done
```

```
[ec2-user@ip-172-31-8-194 ~]$ ./iter.sh
```

```
1
```

```
2
```

```
3
```

```
4
```

Shell Script 실습

❖ 파일 100개 생성(A001, A002, A003...)

>

- 파일 위치: /home/[user name]/
- 파일 명: TABA_7_file_[num].txt
- 파일내용: 이 파일은 TABA_7_file_[num].txt 입니다.

[num]: 1,2,3...

❖ 파일 읽기

cat

- TABA_7_file_[번호].txt 파일 읽기

❖ 파일 복사

cp

- 파일 명: TABA_7_file_copy.txt

❖ 파일 삭제

rm

- 이전에 생성한 모든 파일 삭제

Source command

❖ What is the Source Command?

- 셸 스크립트 파일에 정의된 변수, 함수, 환경 설정 등을 현재 셸에 적용

▪ Syntax of Source Command

- Source [filename]
- . [filename]

▪ Example of Source Command

- Execution of Shell Scripts:
- Modifying Environment Variables:
- Loading Configuration Files:
- Defining Shell Functions:

Source command

❖ Execution of Shell Scripts

- 스크립트 파일 실행권한 없어도 됨

```
-rw-r--r--. 1 ec2-user ec2-user 70 Dec 27 01:10 source_ex1.sh

[ec2-user@ip-172-31-7-196 shellscript]$ cat source_ex1.sh
#!/bin/bash

MY_VARIABLE="Hello, from myscript.sh"

echo $MY_VARIABLE
[ec2-user@ip-172-31-7-196 shellscript]$ source source_ex1.sh
Hello, from myscript.sh
```

❖ Modifying Environment Variables

```
[ec2-user@ip-172-31-7-196 shellscript]$ cat source_ex2.sh
#!/bin/bash

export MY_ENV_VARIABLE="This is my environment variable"
[ec2-user@ip-172-31-7-196 shellscript]$ source source_ex2.sh
[ec2-user@ip-172-31-7-196 shellscript]$ echo $MY_ENV_VARIABLE
This is my environment variable
```

Source command

❖ Loading Configuration Files

- ~/.bashrc 파일에 alias myalias='ls -la' 추가
- 'myalias'라는 이름의 별칭을 생성하며,
이 별칭을 사용할때마다 ls -la 명령이 실행

```
[ec2-user@ip-172-31-7-196 shellscrip]$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
then
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"
fi
export PATH

# Uncomment the following line if you don't like system
e:
# export SYSTEMD_PAGER=

# User specific aliases and functions
if [ -d ~/.bashrc.d ]; then
    for rc in ~/.bashrc.d/*; do
        if [ -f "$rc" ]; then
            . "$rc"
        fi
    done
fi

unset rc
alias myalias='ls -la'
```

```
[ec2-user@ip-172-31-7-196 shellscrip]$ source ~/.bashrc
[ec2-user@ip-172-31-7-196 shellscrip]$ myalias
total 100
drwxr-xr-x. 3 ec2-user ec2-user 4096 Dec 27 01:23 .
drwx----- 5 ec2-user ec2-user 4096 Dec 27 01:26 ..
-rwxr-xr-x. 1 ec2-user ec2-user 131 Dec 27 00:42 exercise.sh
drwxr-xr-x. 2 ec2-user ec2-user   6 Dec 27 00:32 file_100
-rwxr-xr-x. 1 ec2-user ec2-user 238 Dec 27 00:13 file_100.sh
-rwxr-xr-x. 1 ec2-user ec2-user 220 Dec 27 00:28 file_copy.sh
-rwxr-xr-x. 1 ec2-user ec2-user 131 Dec 27 00:24 file_read.sh
-rwxr-xr-x. 1 ec2-user ec2-user 214 Dec 27 00:32 file_remove.sh
```

❖ Defining Shell Functions

```
[ec2-user@ip-172-31-7-196 shellscrip]$ vi source_ex4.sh
[ec2-user@ip-172-31-7-196 shellscrip]$ cat source_ex4.sh
#!/bin/bash

my_function() {
    echo "This is my custom function"
}

[ec2-user@ip-172-31-7-196 shellscrip]$ source source_ex4.sh
[ec2-user@ip-172-31-7-196 shellscrip]$ my_function
This is my custom function
```

❖ Input

- Usage: ./mysl *[argument]*
- [argument]는 디렉토리

*선택

❖ Output

- 디렉토리 내의 파일과 디렉토리 목록을 보여줌
- 1. ./mysl
 - 현재 디렉토리 내의 파일과 디렉토리 목록을 보여줌
- 2. ./mysl [argument]
 - argument에 해당하는 디렉토리 내의 파일과 디렉토리 목록을 보여줌

opendir(), readdir(), closedir()

❖ opendir()

- parameters(const char *name)
 - *const char name: The path of the directory to open
- return value
 - On a successful read, a pointer to the directory stream.
 - If an error occurs, opendir() return NULL.

❖ readdir()

- parameters(DIR *drip)
 - Dir drip: DIR pointer returned by opendir
- return value
 - On a successful read, return 0.
 - If an error occurs, readdir() return -1.

<https://man7.org/linux/man-pages/index.html>

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <errno.h>
6 #include <dirent.h>
7 #include <sys/types.h>
8
9 int main(int argc, char *argv[]) {
10 // 변수 선언
11     DIR *dir = NULL;
12     struct dirent* dentry = NULL;
13     char *dir_name = ".";
14
15     // 예외 처리
16     if (argc == 1) { // args 없는 경우 현재 디렉토리 "." 내용을 보여줌.
17         dir = opendir(dir_name);
18     } else if (argc == 2) {
19         dir_name = argv[1]; // 디렉토리 이름을 argv에서 가져옴
20         dir = [1] fill out here using directory syscall
21     } else {
22         printf("argc %d : We only accept 1 or 2 args for now\n", argc);
23         exit(-1);
24     }
25     while ((dentry = [2] using dir syscall) != NULL) {
26         printf("%s\n", dentry->d_name); // 디렉토리 항목 이름을 출력
27     }
28     [3] using dir syscall
29 }

```

Using System call

opendir

readdir

closedir


```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <errno.h>
6 #include <dirent.h>
7 #include <sys/types.h>
8
9 int main(int argc, char *argv[]) {
10 // 변수 선언
11     DIR *dir = NULL;
12     struct dirent* dentry = NULL;
13     char *dir_name = ".";
14
15     // 예외 처리
16     if (argc == 1) { // args 없는 경우 현재 디렉토리 "." 내용을 보여줌.
17         dir = opendir(dir_name);
18     } else if (argc == 2) {
19         dir_name = argv[1]; // 디렉토리 이름을 argv에서 가져옴
20         dir = opendir(dir_name); // 주어진 디렉토리를 연다.
21     } else {
22         printf("argc %d : We only accept 1 or 2 args for now\n", argc);
23         exit(-1);
24     }
25     while ((dentry = readdir(dir)) != NULL) {
26         printf("%s \n", dentry->d_name); // 디렉토리 항목 이름을 출력
27     }
28     closedir(dir);
29 }
```

Using System call

opendir

readdir

closedir

❖ myls 실행화면

- 인자(argv) 0개와 1개 비교

```
[ec2-user@ip-172-31-15-105 taba9]$ ./mysls
```

```
.  
..  
mycp  
mycp.c  
origin  
mycp_adv.c  
mycp_adv  
dest  
mysls.c  
mysls
```

```
[ec2-user@ip-172-31-15-105 taba9]$ ./mysls ..
```

```
.  
..  
.bash_logout  
.bash_profile  
.ssh  
.bash_history  
.vimrc  
.bashrc  
day2  
taba7  
taba9  
.viminfo
```