실습 2주차

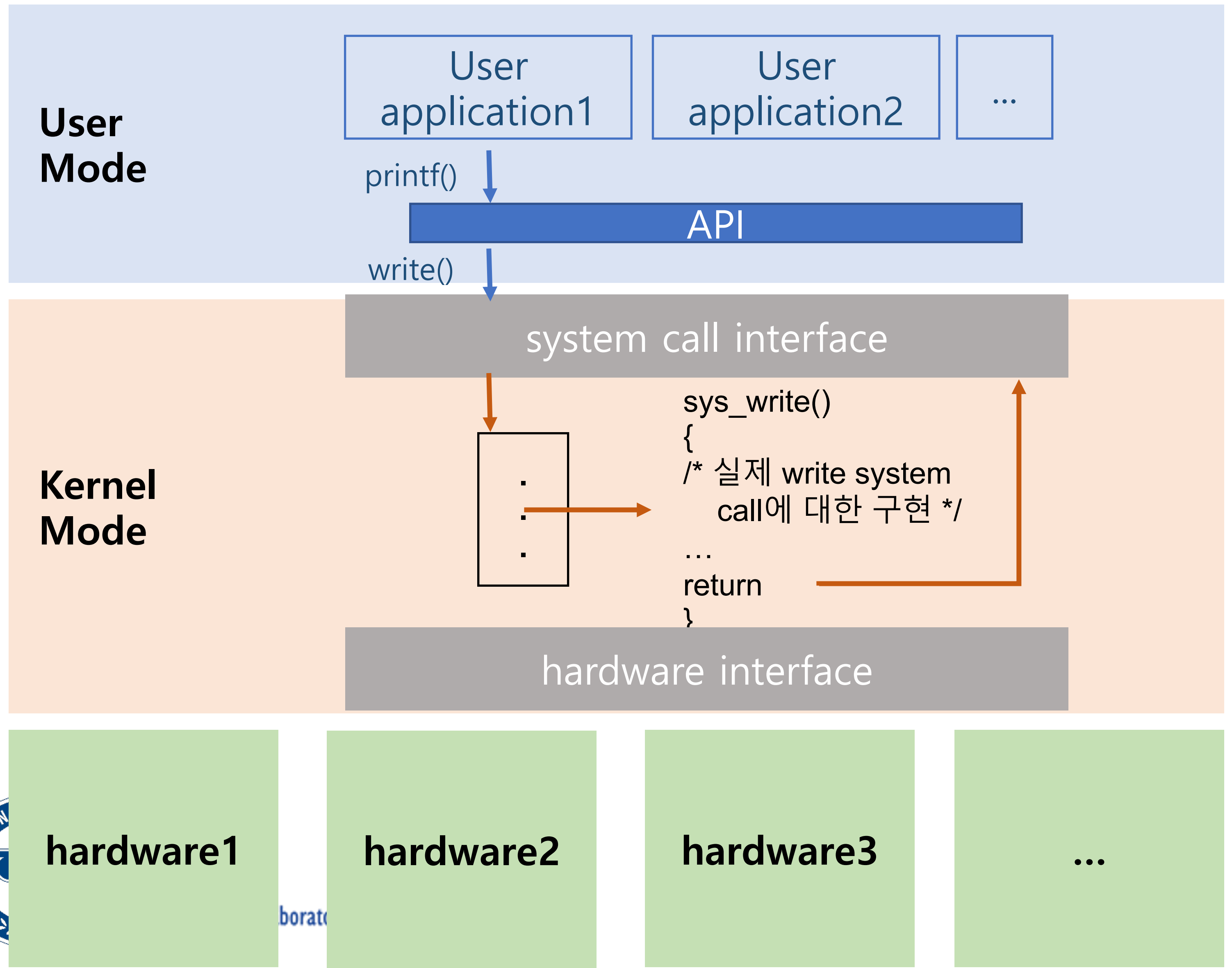
CPS LAB

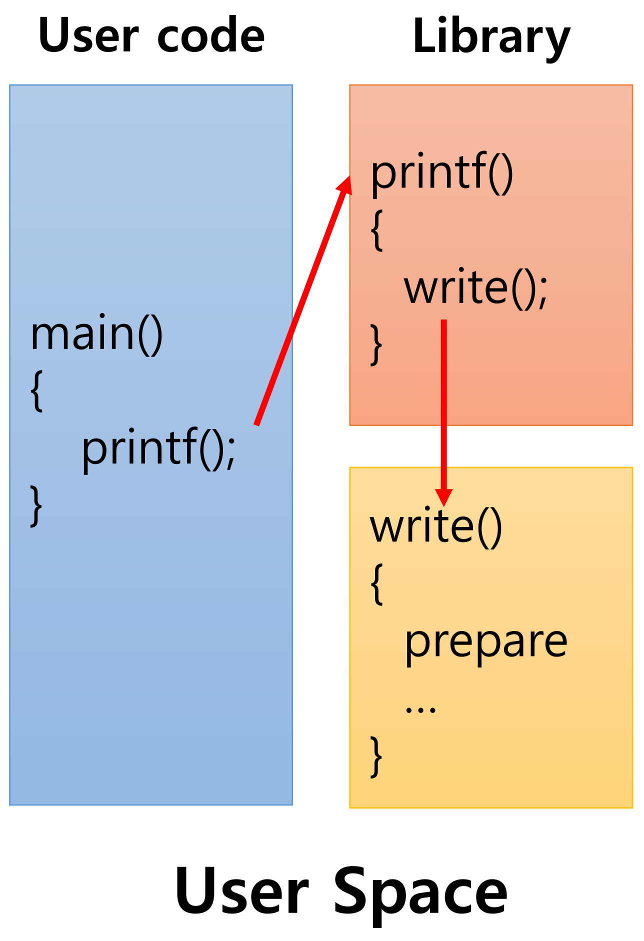
시스템 프로그래밍의 이해

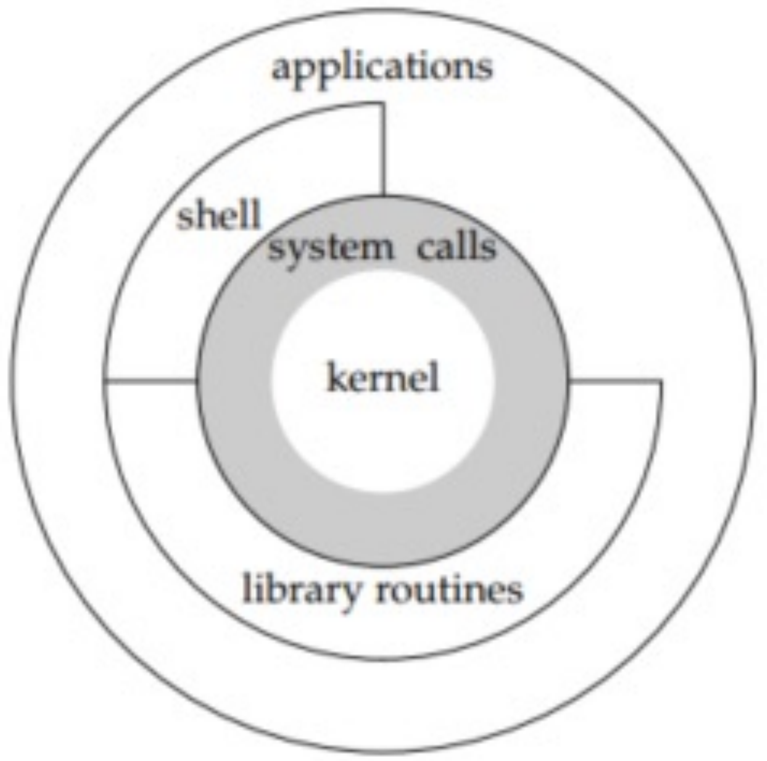
목차

System Call

- User process는 일반적으로 **kernel 영역에 직접적으로 접근할 수 없다.**
  → kernel의 자료구조 및 hardware 에 대한 접근 불가
- 커널 접근 방식
  - 1) **커널 모듈 구현 (추천)** 2) **커널 소스 직접 수정하고, 컴파일한다.** (상당한 위험이 있음)





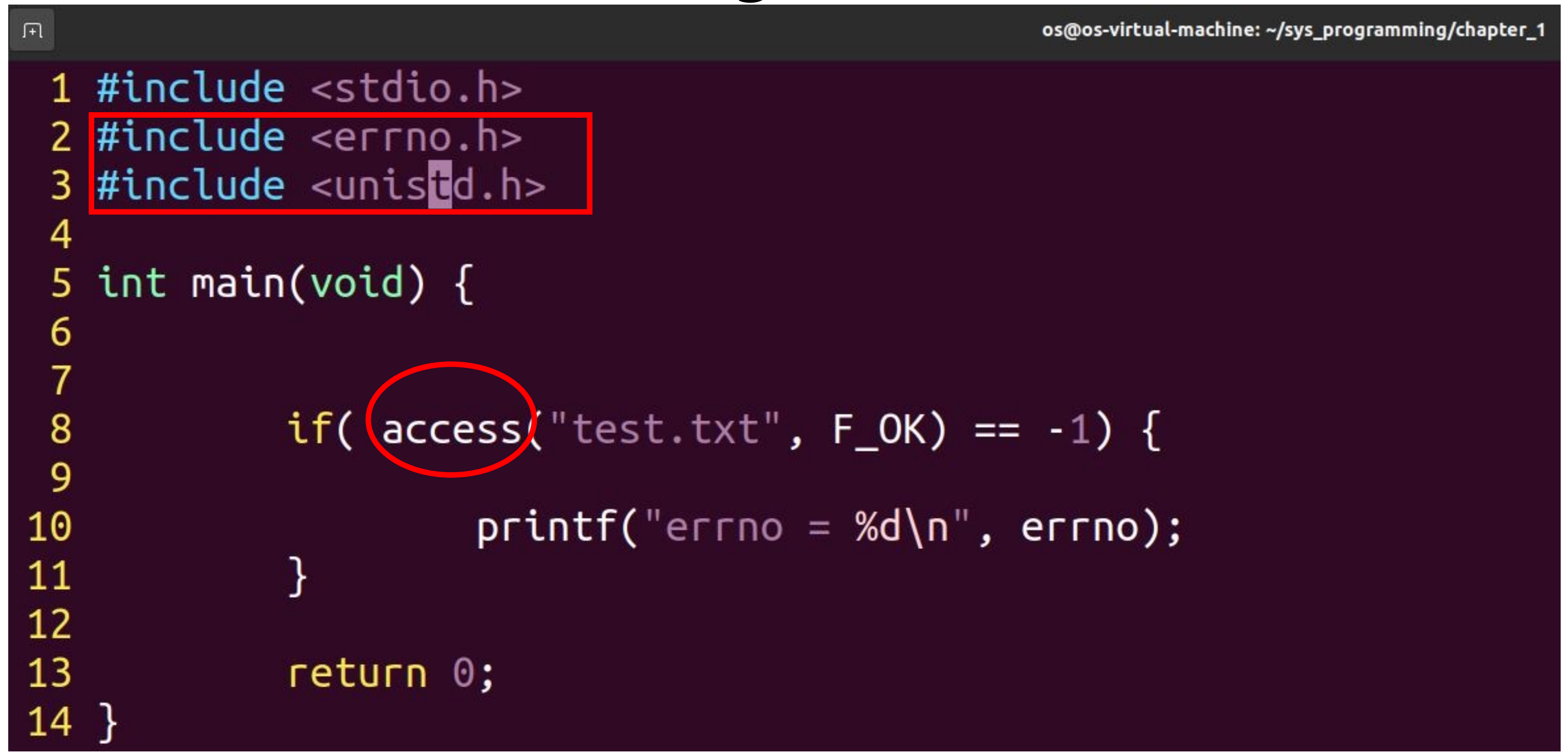


예제 풀이

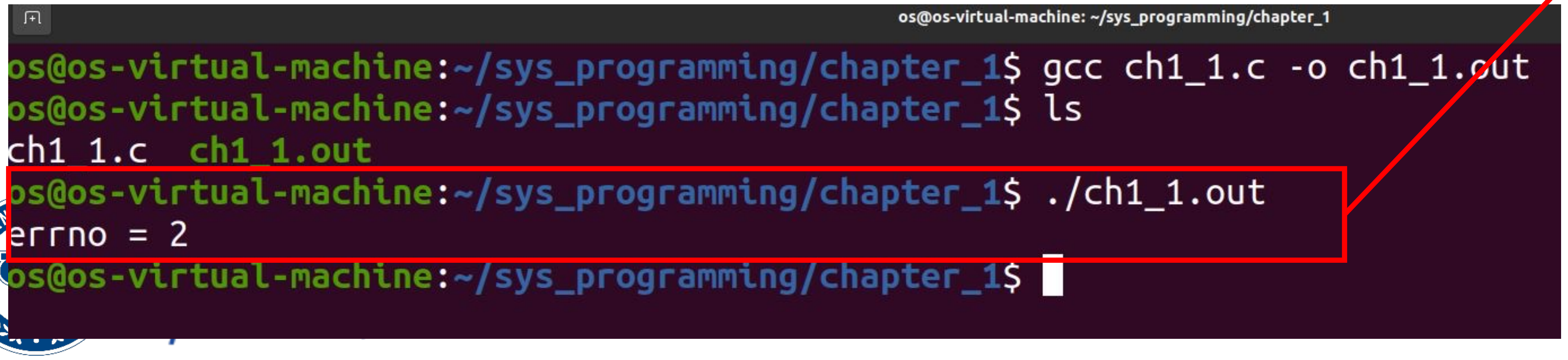
예제 1-1

시스템 호출의 오류 처리 (**ch1_1.c**)

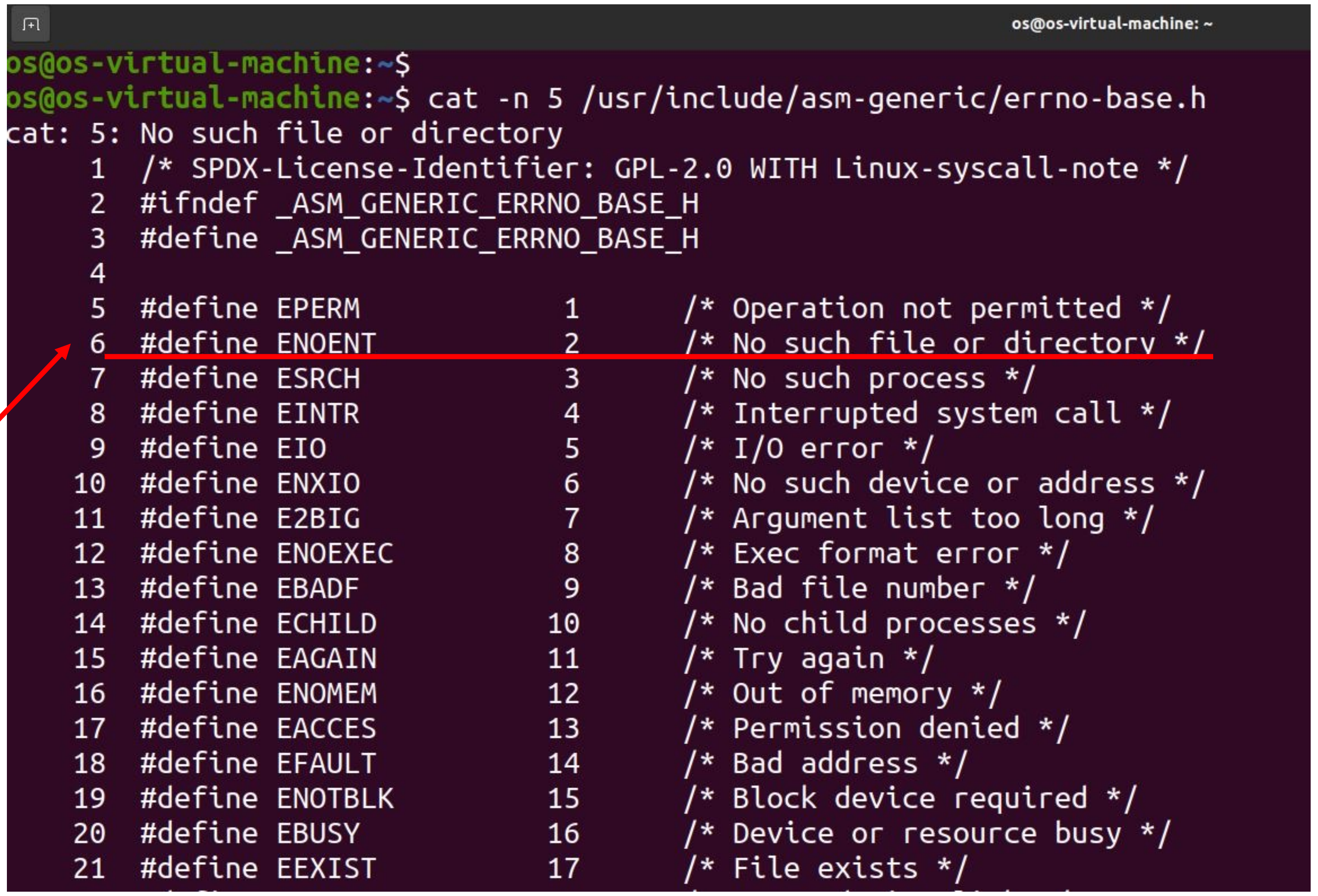
- 최근 시스템 호출 오류는 errno에 저장함
- **/usr/include/asm-generic/errno-base.h**에서 각종 에러를 설명함



```c
#include <stdio.h>
#include <errno.h>
#include <unistd.h>

int main(void) {


        if( access("test.txt", F_OK) == -1) {

                printf("errno = %d\n", errno);
        }

        return 0;
}
```



```
os@os-virtual-machine:~/sys_programming/chapter_1$ gcc ch1_1.c -o ch1_1.out
os@os-virtual-machine:~/sys_programming/chapter_1$ ls
ch1_1.c   ch1_1.out
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_1.out
errno = 2
os@os-virtual-machine:~/sys_programming/chapter_1$ 
```

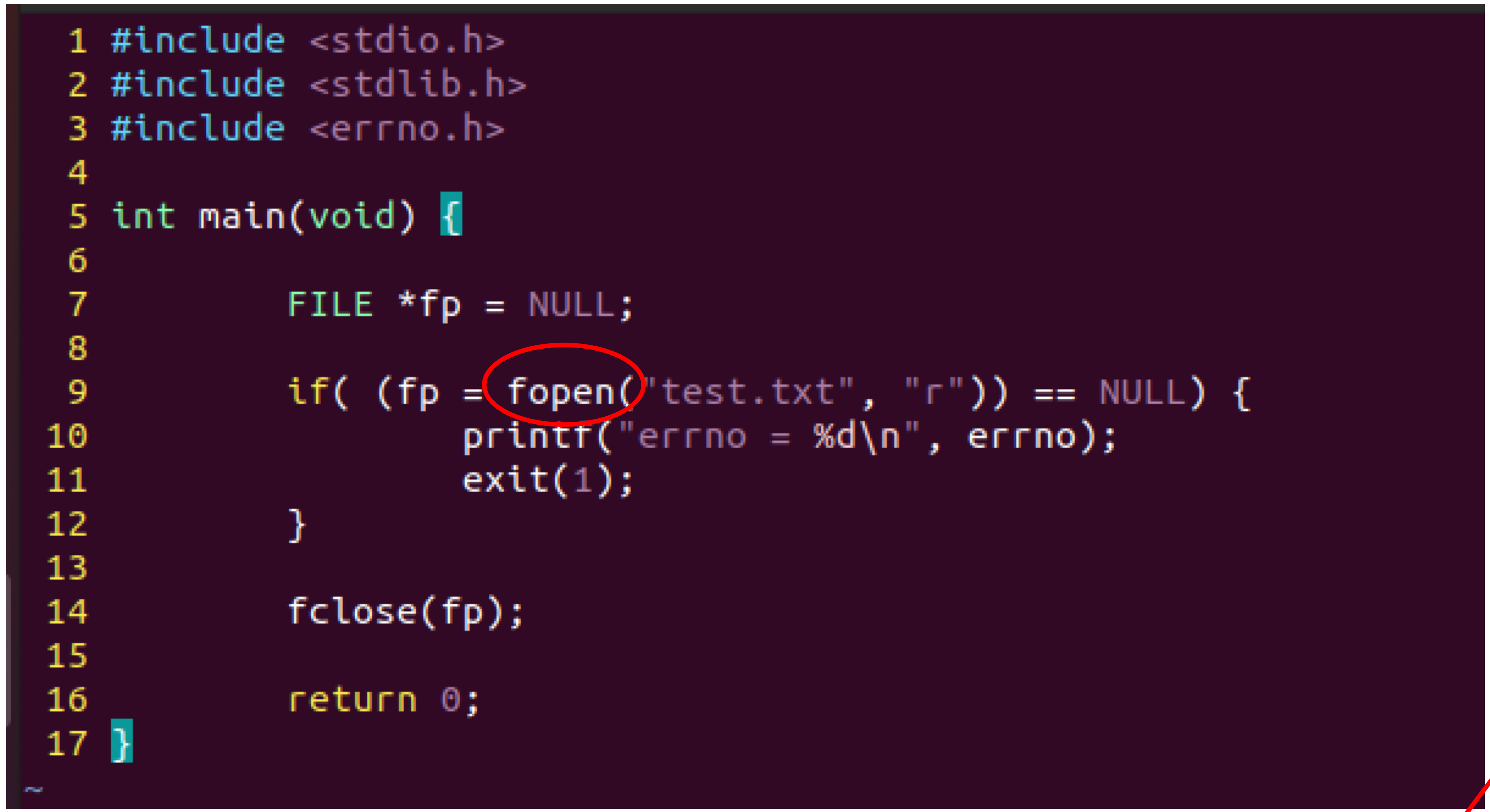


```
os@os-virtual-machine:~$
os@os-virtual-machine:~$ cat -n 5 /usr/include/asm-generic/errno-base.h
cat: 5: No such file or directory
     1  /* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
     2  #ifndef _ASM_GENERIC_ERRNO_BASE_H
     3  #define _ASM_GENERIC_ERRNO_BASE_H
     4
     5  #define EPERM            1      /* Operation not permitted */
     6  #define ENOENT           2      /* No such file or directory */
     7  #define ESRCH            3      /* No such process */
     8  #define EINTR            4      /* Interrupted system call */
     9  #define EIO              5      /* I/O error */
    10  #define ENXIO            6      /* No such device or address */
    11  #define E2BIG            7      /* Argument list too long */
    12  #define ENOEXEC          8      /* Exec format error */
    13  #define EBADF            9      /* Bad file number */
    14  #define ECHILD          10      /* No child processes */
    15  #define EAGAIN          11      /* Try again */
    16  #define ENOMEM          12      /* Out of memory */
    17  #define EACCES          13      /* Permission denied */
    18  #define EFAULT          14      /* Bad address */
    19  #define ENOTBLK         15      /* Block device required */
    20  #define EBUSY           16      /* Device or resource busy */
    21  #define EEXIST          17      /* File exists */
```

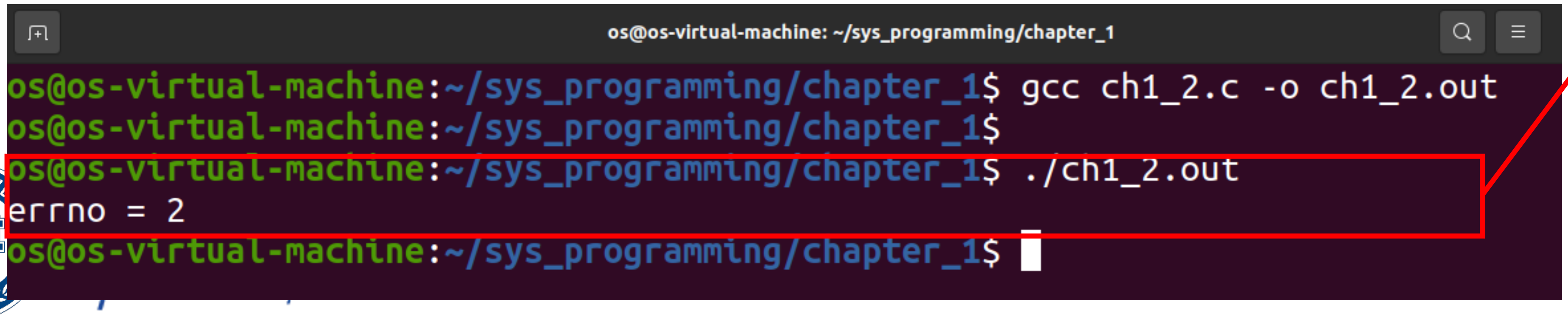
예제 풀이

예제 1-2

라이브러리 함수의 오류 처리하기 (ch1_2.c)

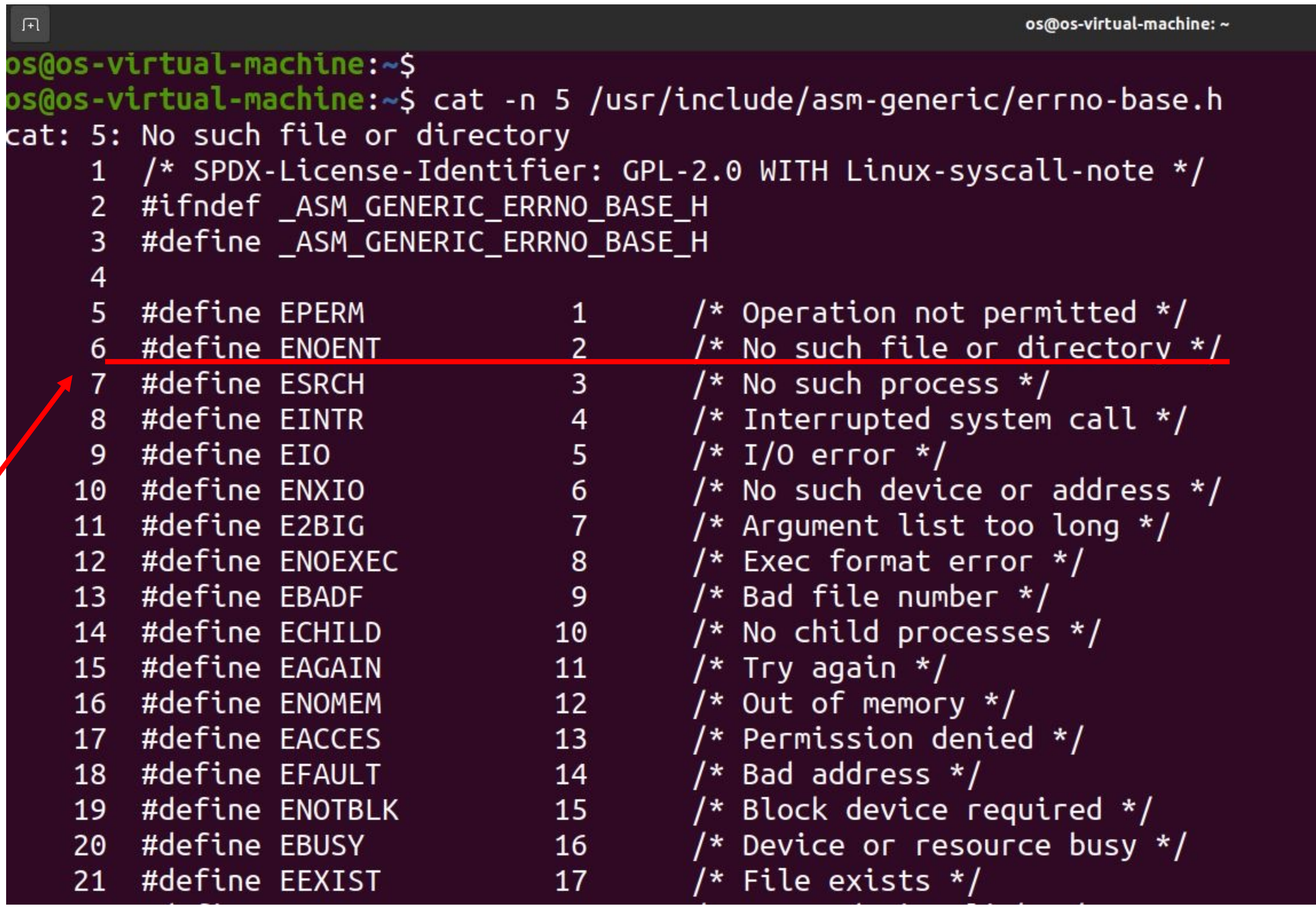


```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

int main(void) {

        FILE *fp = NULL;

        if( (fp = fopen("test.txt", "r")) == NULL) {
                printf("errno = %d\n", errno);
                exit(1);
        }

        fclose(fp);

        return 0;
}
~
```



os@os-virtual-machine: ~/sys_programming/chapter_1

```
os@os-virtual-machine:~/sys_programming/chapter_1$ gcc ch1_2.c -o ch1_2.out
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_2.out
errno = 2
os@os-virtual-machine:~/sys_programming/chapter_1$
```



os@os-virtual-machine: ~

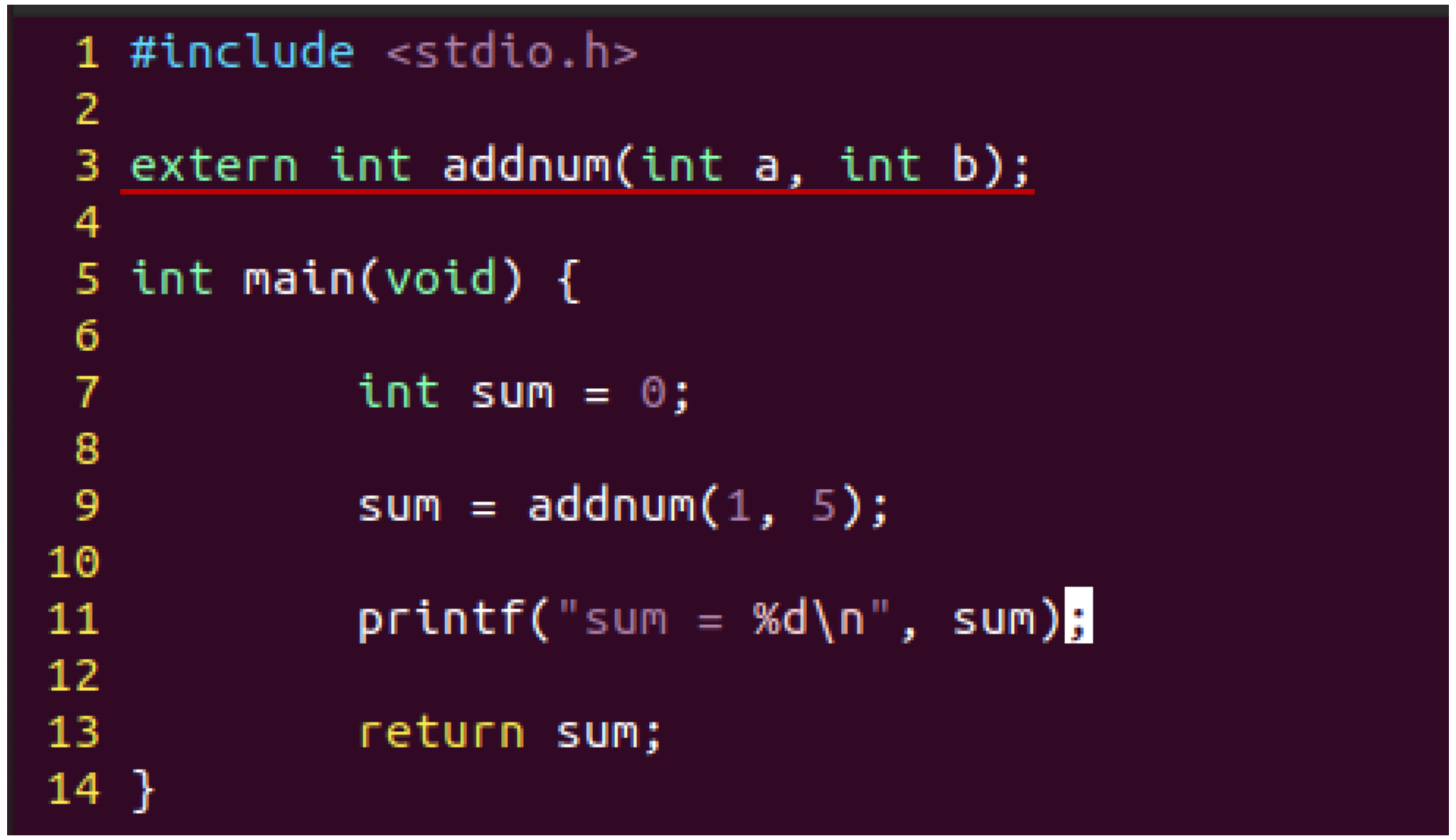
```
os@os-virtual-machine:~$
os@os-virtual-machine:~$ cat -n 5 /usr/include/asm-generic/errno-base.h
cat: 5: No such file or directory
     1  /* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
     2  #ifndef _ASM_GENERIC_ERRNO_BASE_H
     3  #define _ASM_GENERIC_ERRNO_BASE_H
     4
     5  #define EPERM            1      /* Operation not permitted */
     6  #define ENOENT           2      /* No such file or directory */
     7  #define ESRCH            3      /* No such process */
     8  #define EINTR            4      /* Interrupted system call */
     9  #define EIO              5      /* I/O error */
    10  #define ENXIO            6      /* No such device or address */
    11  #define E2BIG            7      /* Argument list too long */
    12  #define ENOEXEC          8      /* Exec format error */
    13  #define EBADF            9      /* Bad file number */
    14  #define ECHILD          10      /* No child processes */
    15  #define EAGAIN          11      /* Try again */
    16  #define ENOMEM          12      /* Out of memory */
    17  #define EACCES          13      /* Permission denied */
    18  #define EFAULT          14      /* Bad address */
    19  #define ENOTBLK         15      /* Block device required */
    20  #define EBUSY           16      /* Device or resource busy */
    21  #define EEXIST          17      /* File exists */
```

예제 풀이

예제 1-3

make 명령 사용하기 (**ch1_3_main.c, ch1_3_addnum.c, Makefile**)

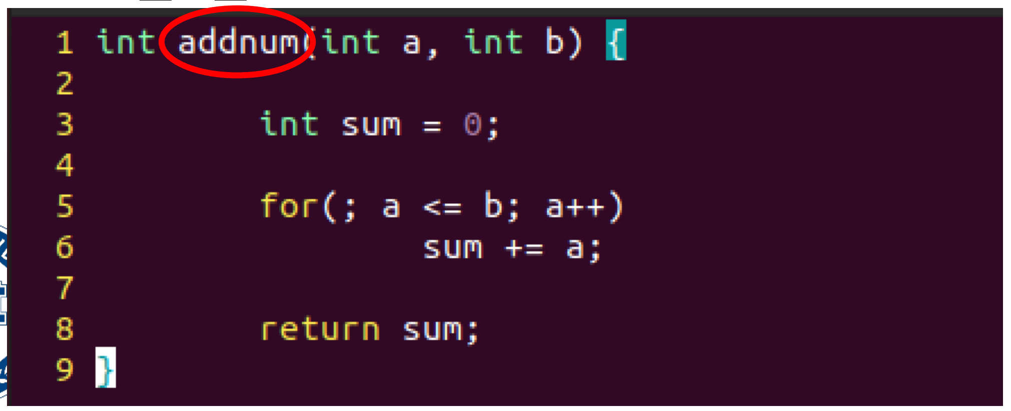
ch1_3_main.c



```c
#include <stdio.h>

extern int addnum(int a, int b);

int main(void) {

        int sum = 0;

        sum = addnum(1, 5);

        printf("sum = %d\n", sum);

        return sum;
}
```

함수를 공유하기 위해 extern 키워드를 사용한다. 헤더 파일을 따로 만들지 않고도 두 개의 c 파일이 함수를 공유할 수 있다.

ch1_3_addnum.c



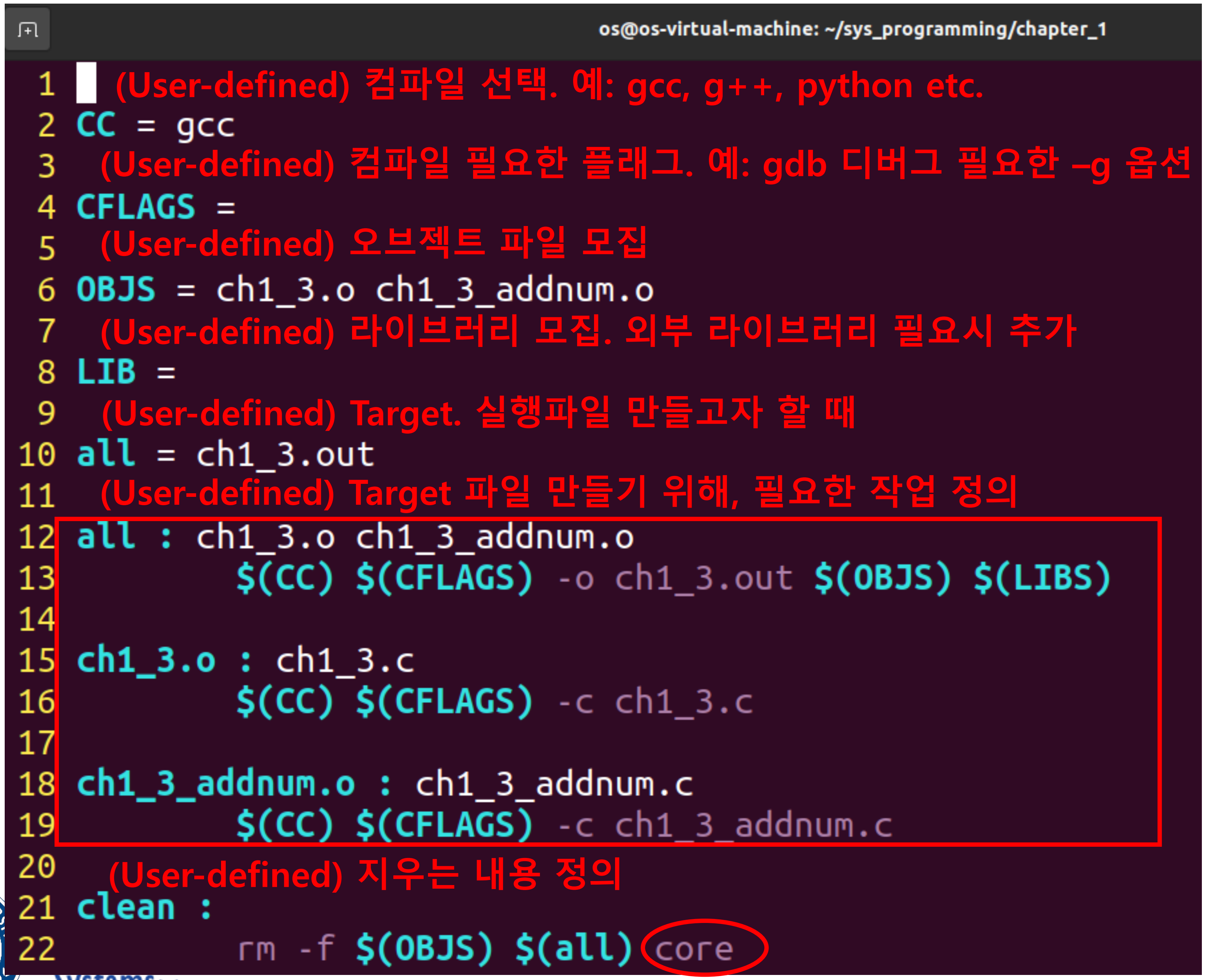
```c
int addnum(int a, int b) {

        int sum = 0;

        for(; a <= b; a++)
                sum += a;

        return sum;
}
```

예제 풀이

예제 1-3

make 명령 사용하기 (**ch1_3_main.c, ch1_3_addnum.c, Makefile**)

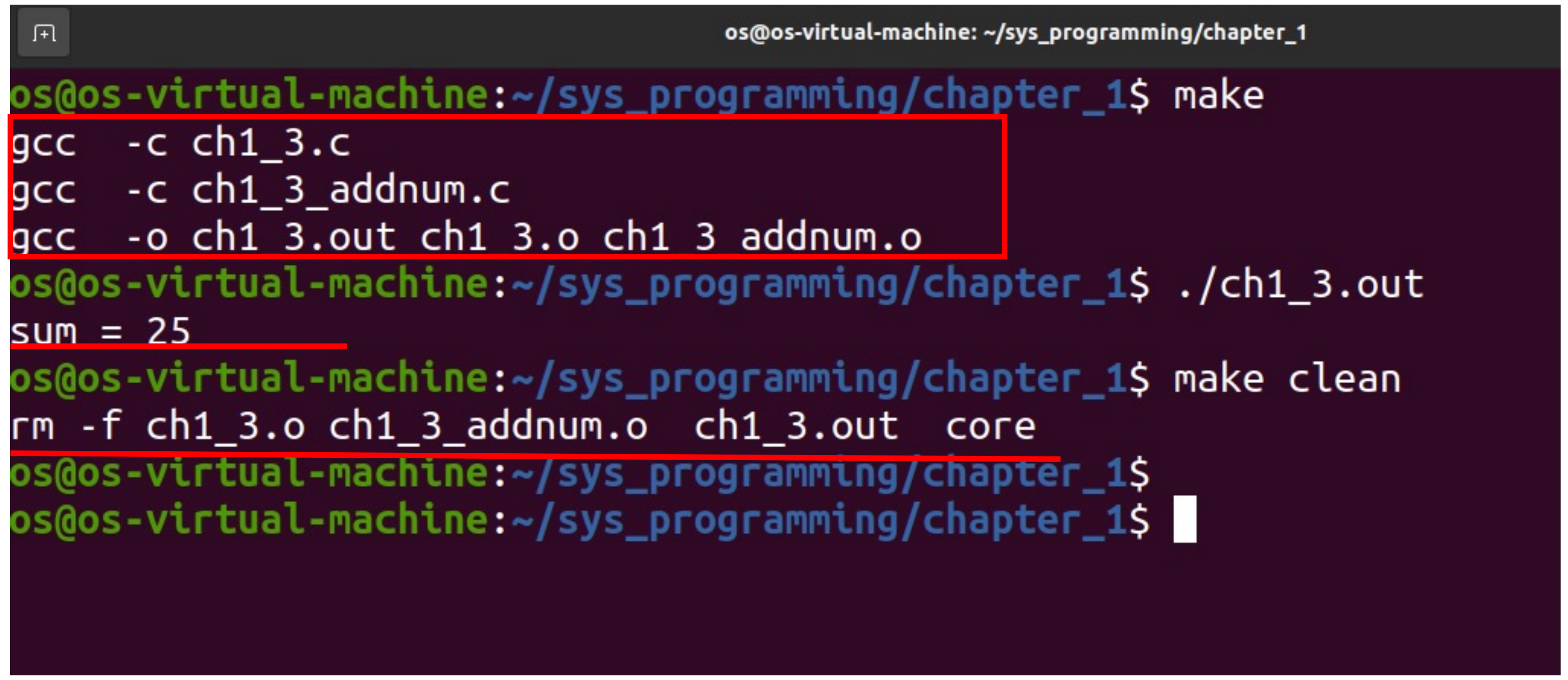
Makefile



os@os-virtual-machine: ~/sys_programming/chapter_1

```
 (User-defined) 컴파일 선택. 예: gcc, g++, python etc.
CC = gcc
 (User-defined) 컴파일 필요한 플래그. 예: gdb 디버그 필요한 –g 옵션
CFLAGS =
 (User-defined) 오브젝트 파일 모집
OBJS = ch1_3.o ch1_3_addnum.o
 (User-defined) 라이브러리 모집. 외부 라이브러리 필요시 추가
LIB =
 (User-defined) Target. 실행파일 만들고자 할 때
all = ch1_3.out
 (User-defined) Target 파일 만들기 위해, 필요한 작업 정의
all : ch1_3.o ch1_3_addnum.o
        $(CC) $(CFLAGS) -o ch1_3.out $(OBJS) $(LIBS)

ch1_3.o : ch1_3.c
        $(CC) $(CFLAGS) -c ch1_3.c

ch1_3_addnum.o : ch1_3_addnum.c
        $(CC) $(CFLAGS) -c ch1_3_addnum.c
  (User-defined) 지우는 내용 정의
clean :
        rm -f $(OBJS) $(all) core
```

필드 과정에서 생성된 파일



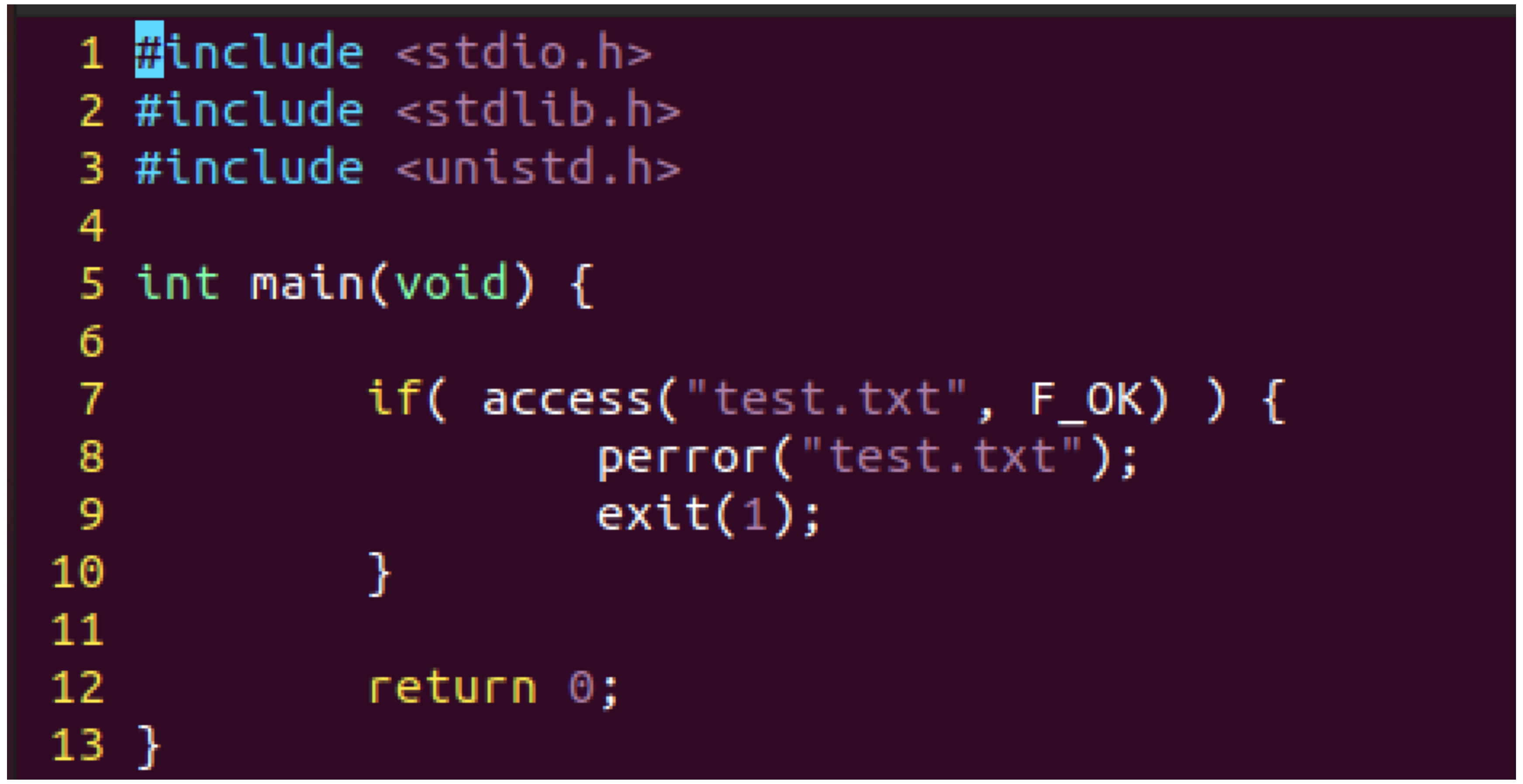
os@os-virtual-machine: ~/sys_programming/chapter_1

```
os@os-virtual-machine:~/sys_programming/chapter_1$ make
gcc   -c ch1_3.c
gcc   -c ch1_3_addnum.c
gcc   -o ch1_3.out ch1_3.o ch1_3_addnum.o
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_3.out
sum = 25
os@os-virtual-machine:~/sys_programming/chapter_1$ make clean
rm -f ch1_3.o ch1_3_addnum.o  ch1_3.out  core
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$
```

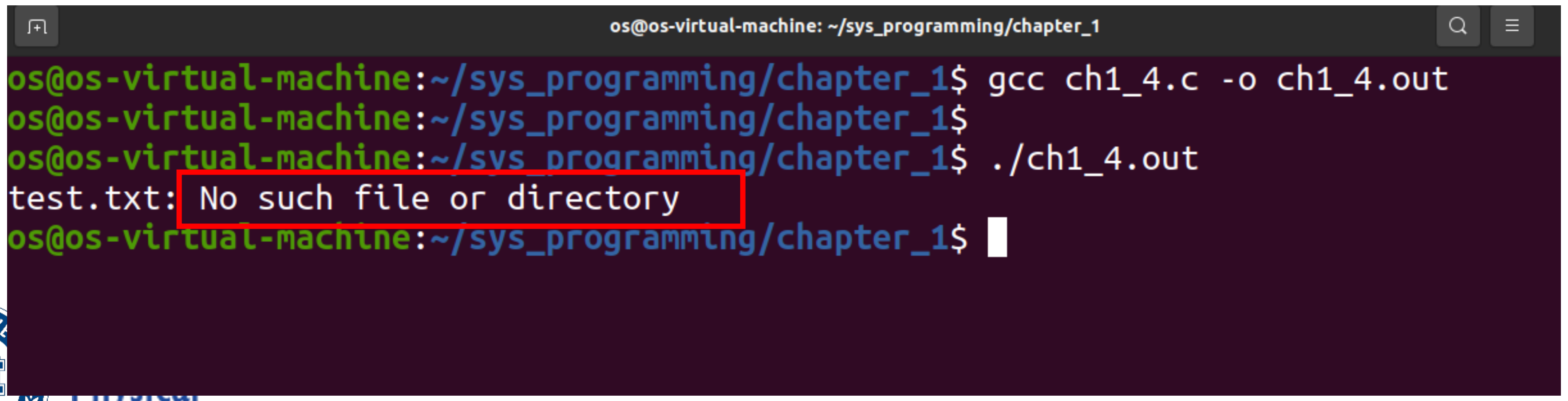
예제 풀이

예제 1-4

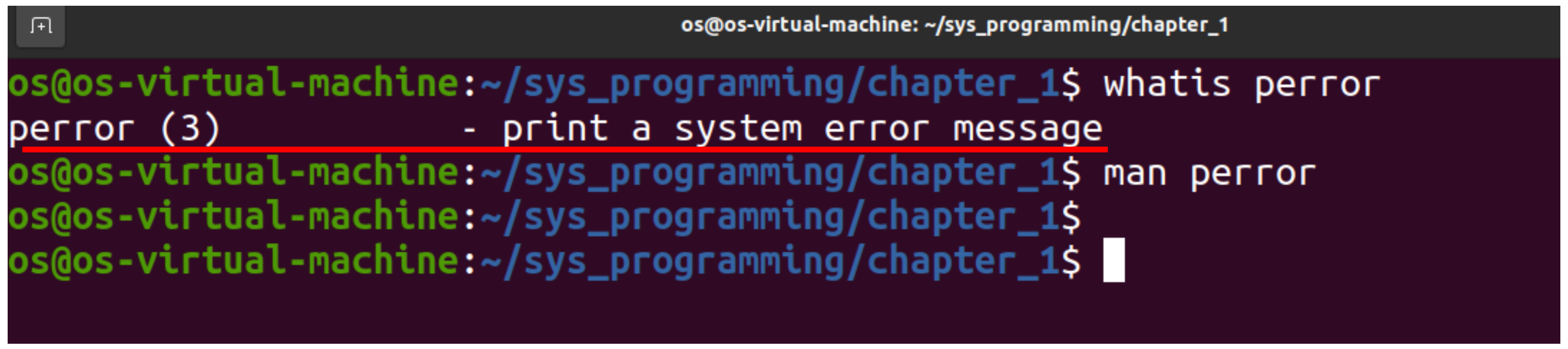
perror()함수로 오류 메시지 출력하기 (**ch1_4.c**)



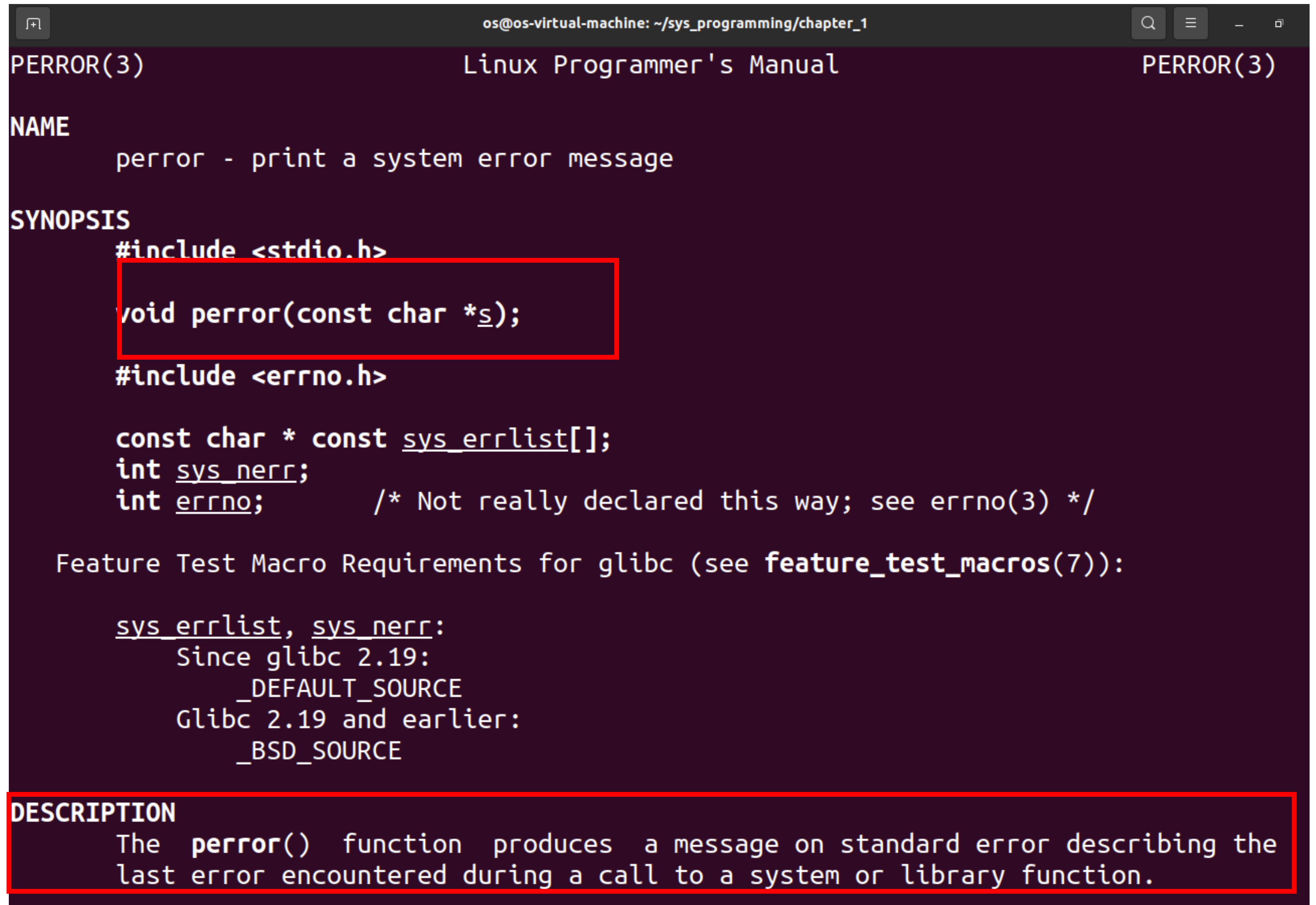
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(void) {

        if( access("test.txt", F_OK) ) {
                perror("test.txt");
                exit(1);
        }

        return 0;
}
```



```
os@os-virtual-machine:~/sys_programming/chapter_1$ gcc ch1_4.c -o ch1_4.out
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_4.out
test.txt: No such file or directory
os@os-virtual-machine:~/sys_programming/chapter_1$
```



```
os@os-virtual-machine:~/sys_programming/chapter_1$ whatis perror
perror (3)           - print a system error message
os@os-virtual-machine:~/sys_programming/chapter_1$ man perror
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$
```



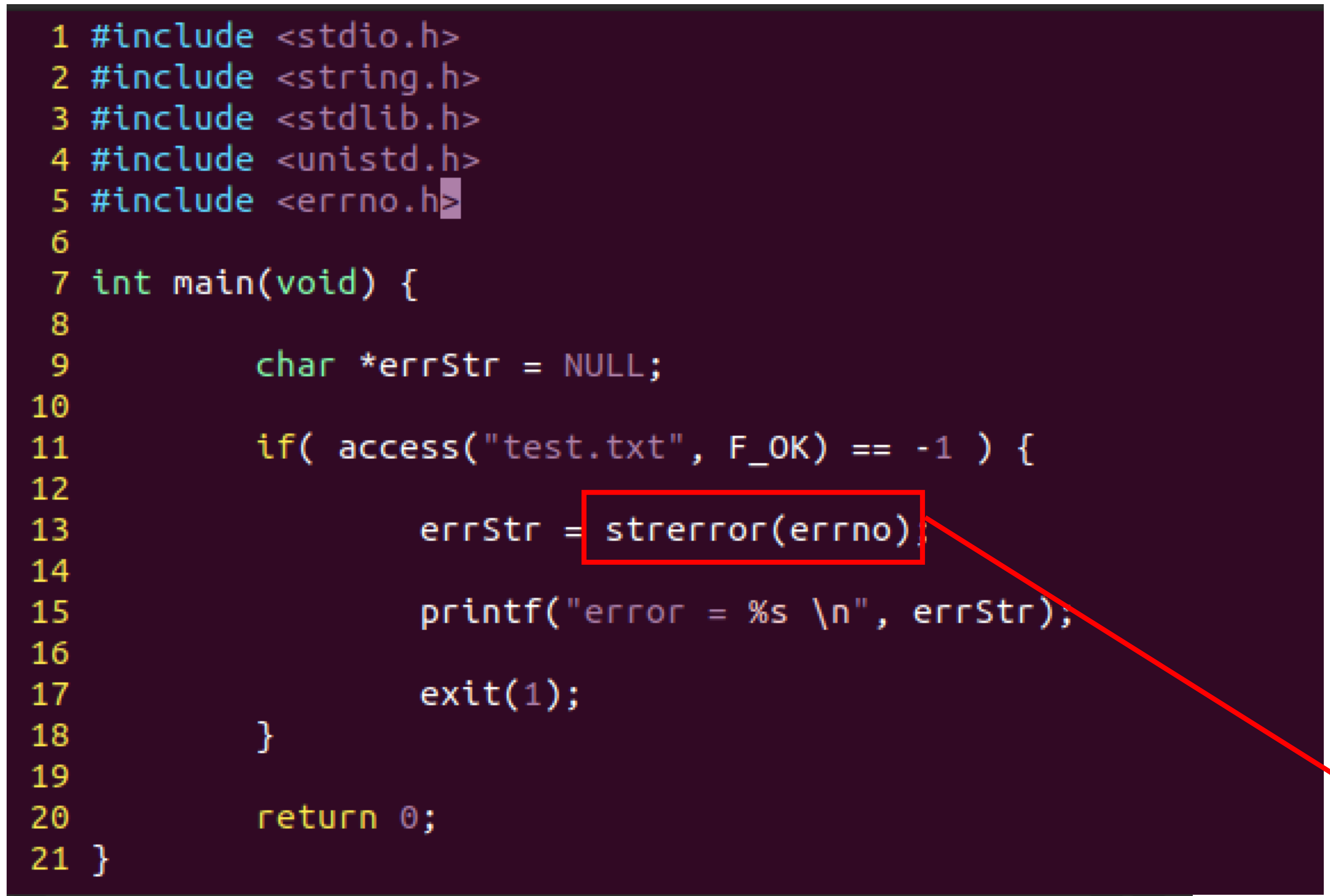
```
PERROR(3)                  Linux Programmer's Manual                  PERROR(3)

NAME
       perror - print a system error message

SYNOPSIS
       #include <stdio.h>

       void perror(const char *s);

       #include <errno.h>

       const char * const sys_errlist[];
       int sys_nerr;
       int errno;       /* Not really declared this way; see errno(3) */

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       sys_errlist, sys_nerr:
           Since glibc 2.19:
               _DEFAULT_SOURCE
           Glibc 2.19 and earlier:
               _BSD_SOURCE

DESCRIPTION
       The  perror()  function  produces  a message on standard error describing the
       last error encountered during a call to a system or library function.
```

예제 풀이

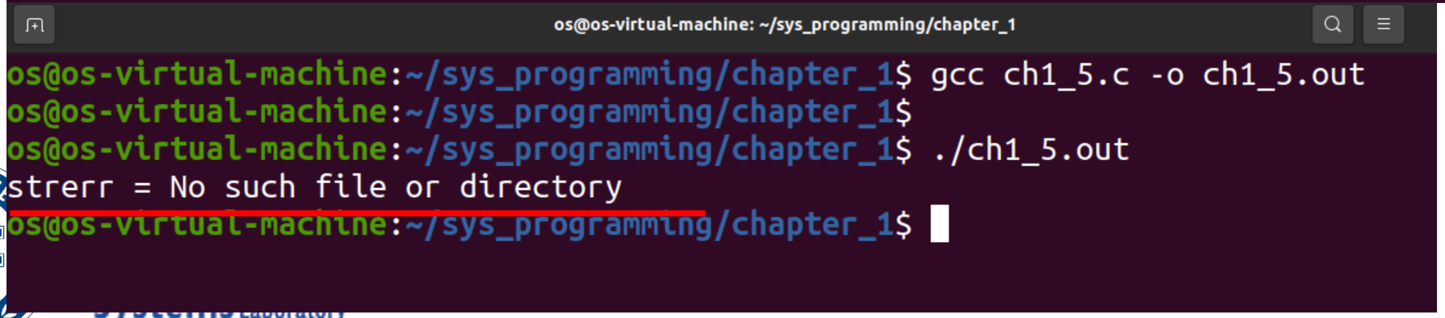
예제 1-5

strerror() 함수를 사용해 오류 메시지를 출력하는 예제 (**ch1_5.c**)

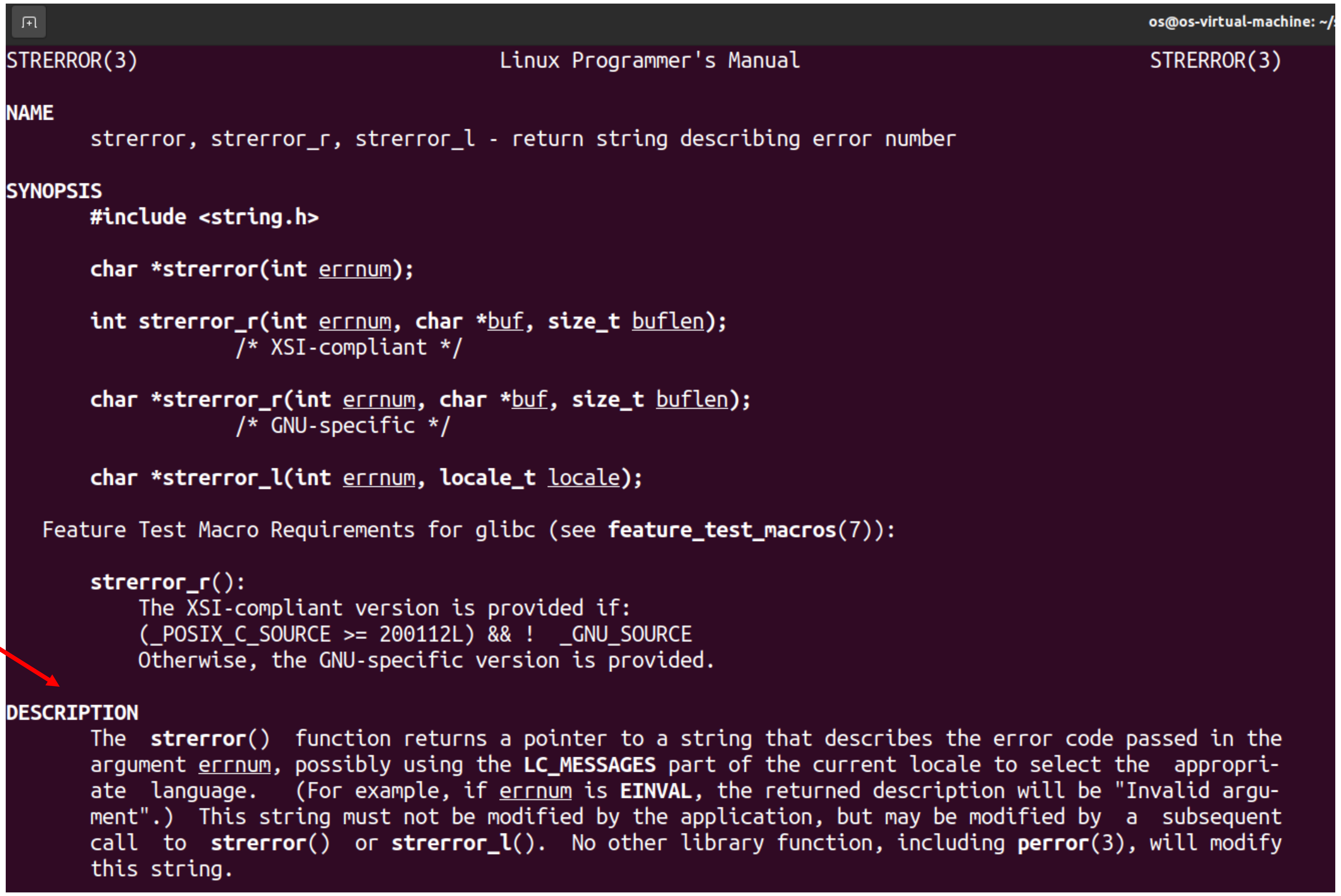
ch1_5.c



```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

int main(void) {

        char *errStr = NULL;

        if( access("test.txt", F_OK) == -1 ) {

                errStr = strerror(errno);

                printf("error = %s \n", errStr);

                exit(1);
        }

        return 0;
}
```



```
os@os-virtual-machine: ~/sys_programming/chapter_1
os@os-virtual-machine:~/sys_programming/chapter_1$ gcc ch1_5.c -o ch1_5.out
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_5.out
strerr = No such file or directory
os@os-virtual-machine:~/sys_programming/chapter_1$
```



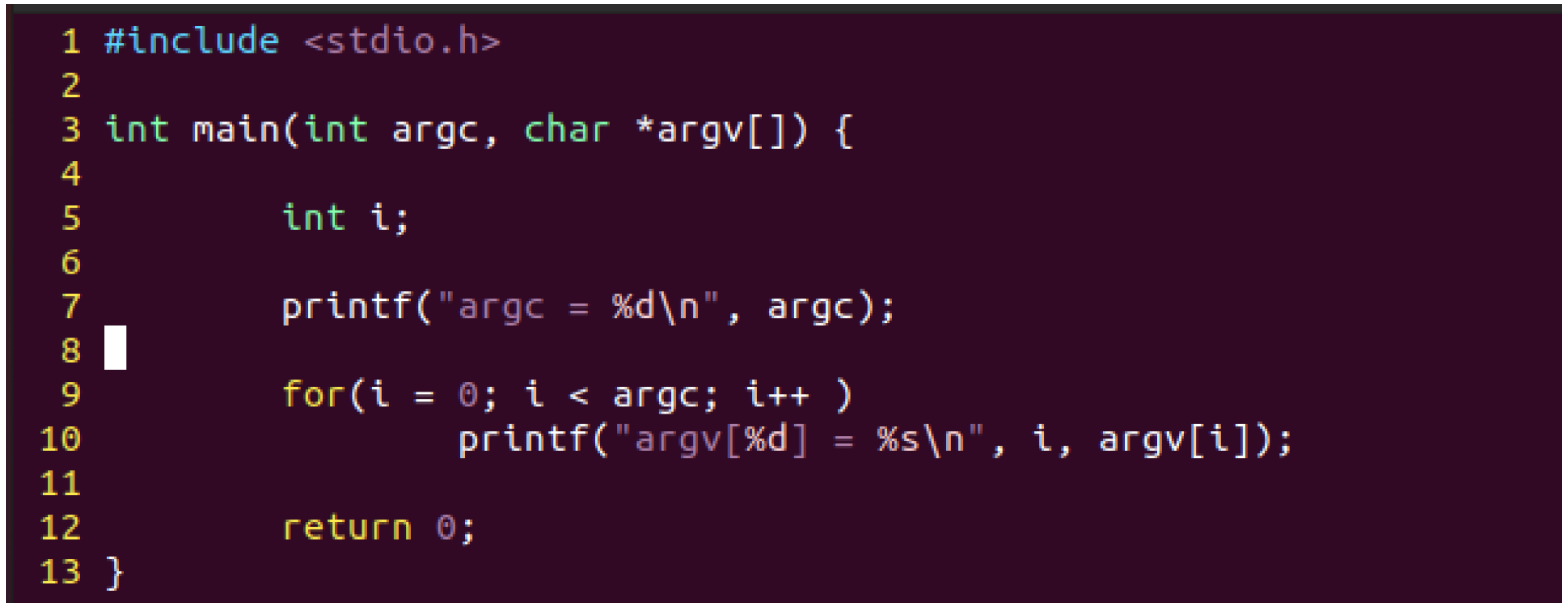
```
os@os-virtual-machine: ~/
STRERROR(3)                Linux Programmer's Manual               STRERROR(3)

NAME
       strerror, strerror_r, strerror_l - return string describing error number

SYNOPSIS
       #include <string.h>

       char *strerror(int errnum);

       int strerror_r(int errnum, char *buf, size_t buflen);
                   /* XSI-compliant */

       char *strerror_r(int errnum, char *buf, size_t buflen);
                   /* GNU-specific */

       char *strerror_l(int errnum, locale_t locale);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       strerror_r():
           The XSI-compliant version is provided if:
           (_POSIX_C_SOURCE >= 200112L) && !  _GNU_SOURCE
           Otherwise, the GNU-specific version is provided.

DESCRIPTION
       The  strerror()  function returns a pointer to a string that describes the error code passed in the
       argument errnum, possibly using the LC_MESSAGES part of the current locale to select the  appropri‐
       ate  language.   (For example, if errnum is EINVAL, the returned description will be "Invalid argu‐
       ment".)  This string must not be modified by the application, but may be modified by  a  subsequent
       call  to  strerror()  or strerror_l().  No other library function, including perror(3), will modify
       this string.
```

예제 풀이

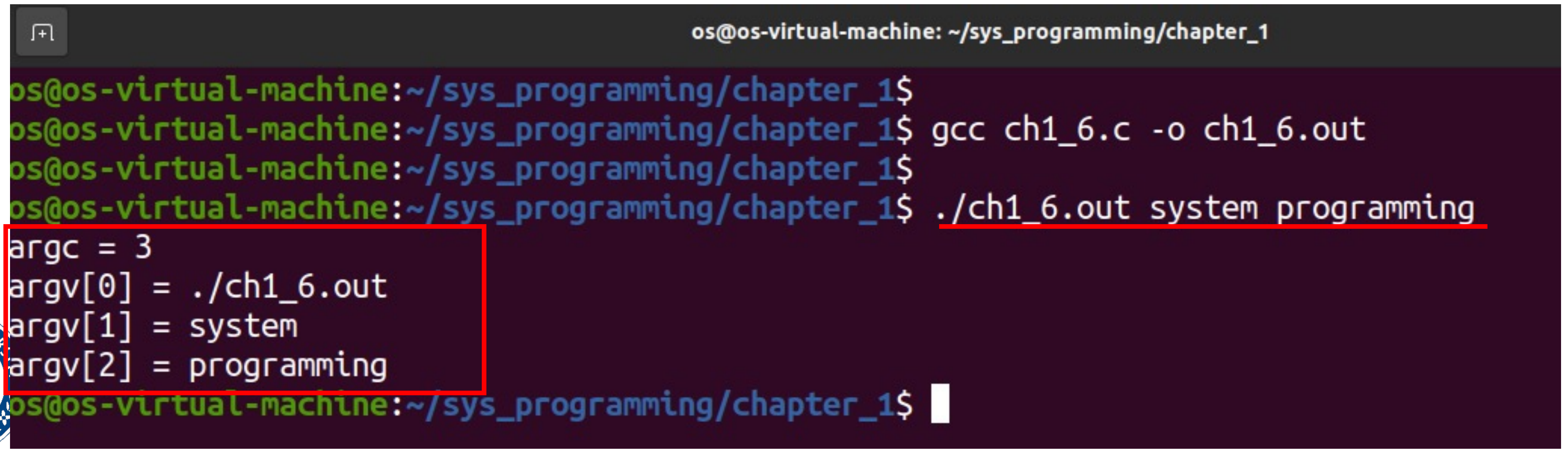
예제 1-6

명령행 인자 출력하기 (**ch1_6.c**)

ch1_6.c



```c
#include <stdio.h>

int main(int argc, char *argv[]) {

        int i;

        printf("argc = %d\n", argc);

        for(i = 0; i < argc; i++ )
                printf("argv[%d] = %s\n", i, argv[i]);

        return 0;
}
```



os@os-virtual-machine: ~/sys_programming/chapter_1

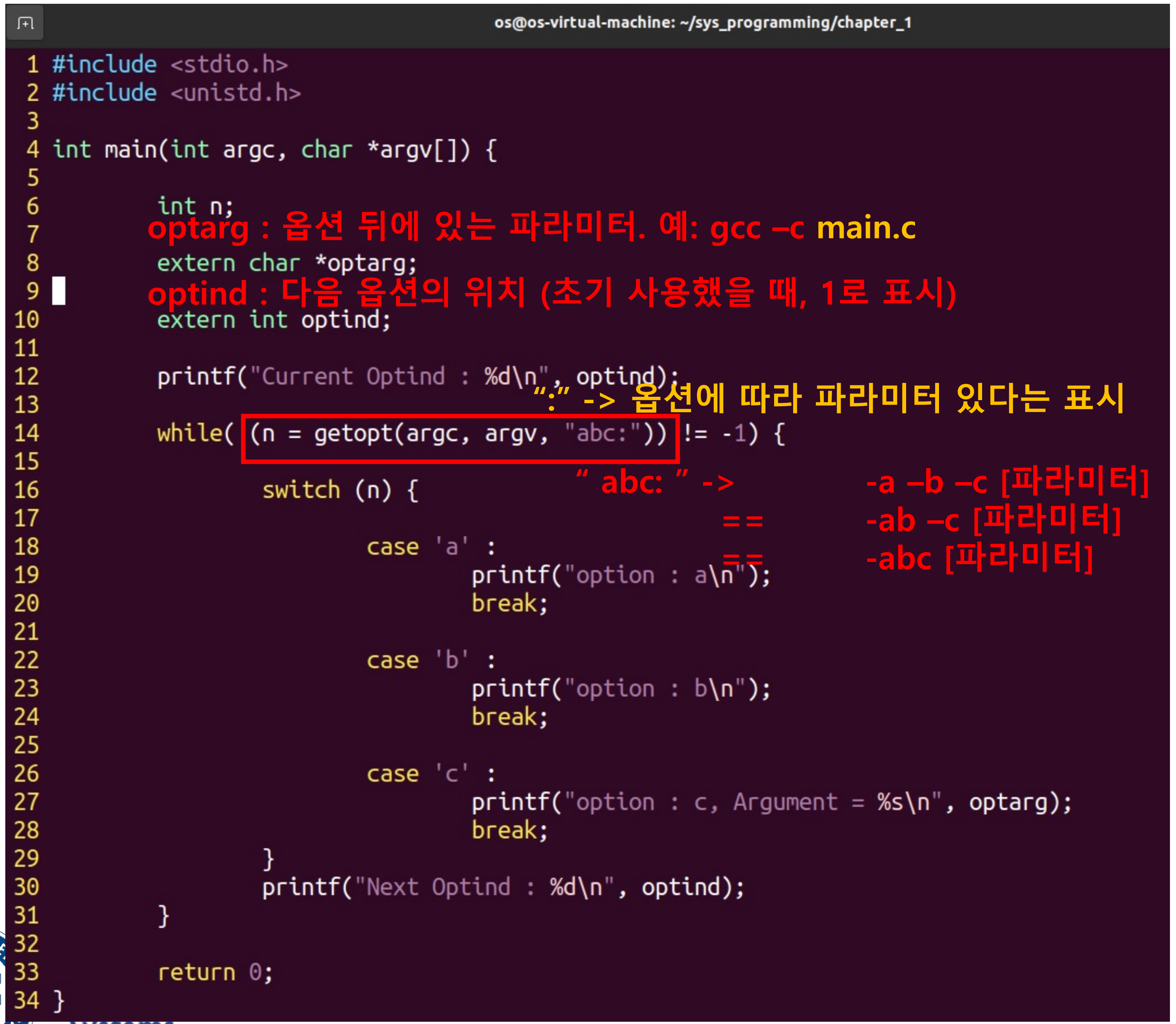
```
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$ gcc ch1_6.c -o ch1_6.out
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_6.out system programming
argc = 3
argv[0] = ./ch1_6.out
argv[1] = system
argv[2] = programming
os@os-virtual-machine:~/sys_programming/chapter_1$
```

예제 풀이

예제 1-7

getopt() 함수로 옵션 처리하기 (**ch1_7.c**)

ch1_7.c



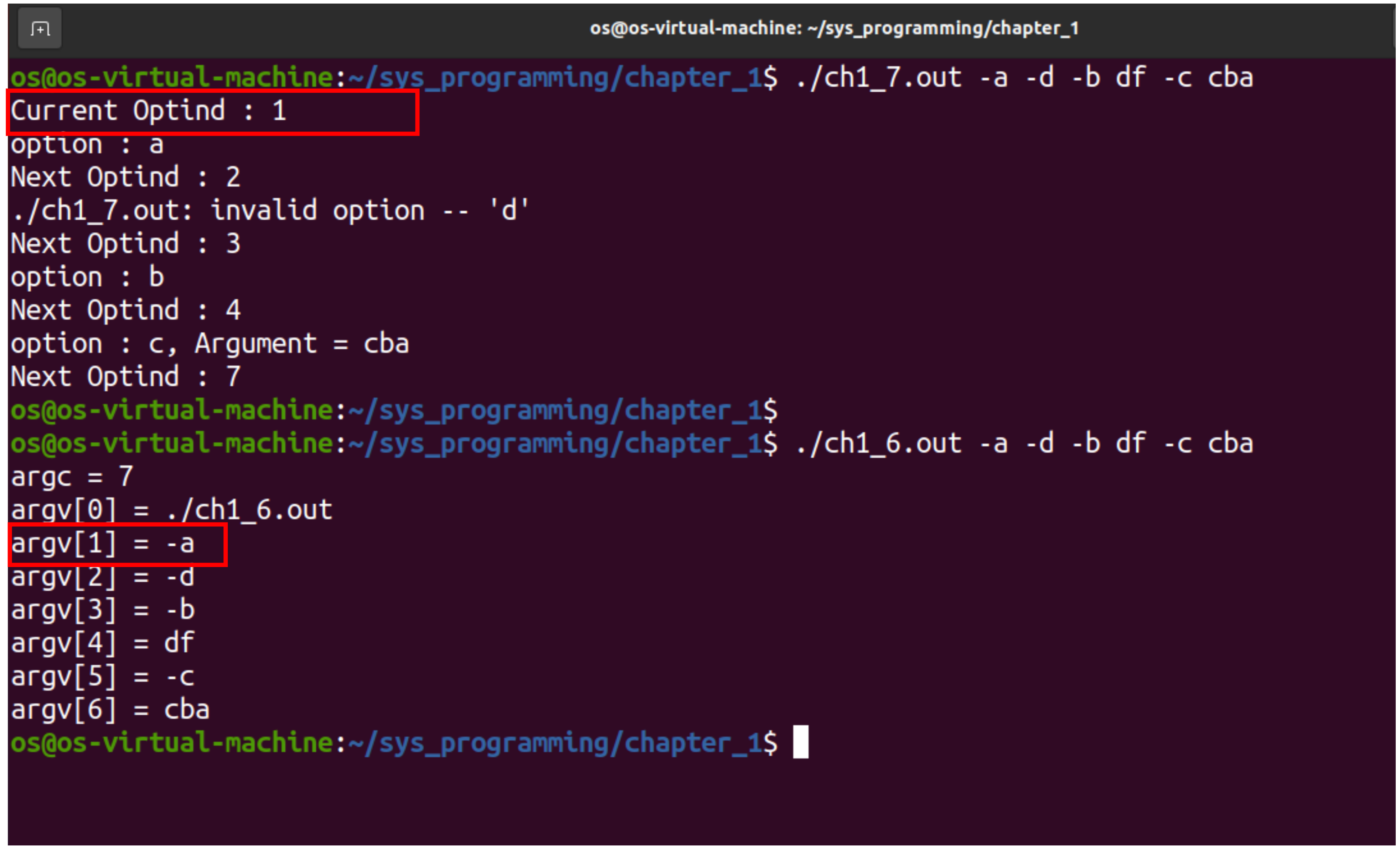
```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[]) {

        int n;

        extern char *optarg;

        extern int optind;

        printf("Current Optind : %d\n", optind);

        while( (n = getopt(argc, argv, "abc:")) != -1) {

                switch (n) {

                        case 'a' :
                                printf("option : a\n");
                                break;

                        case 'b' :
                                printf("option : b\n");
                                break;

                        case 'c' :
                                printf("option : c, Argument = %s\n", optarg);
                                break;
                }
                printf("Next Optind : %d\n", optind);
        }

        return 0;
}
```

optarg : 옵션 뒤에 있는 파라미터. 예: gcc –c main.c

optind : 다음 옵션의 위치 (초기 사용했을 때, 1로 표시)

":" -> 옵션에 따라 파라미터 있다는 표시

" abc: " -> -a –b –c [파라미터] == -ab –c [파라미터] == -abc [파라미터]



```
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_7.out -a -d -b df -c cba
Current Optind : 1
option : a
Next Optind : 2
./ch1_7.out: invalid option -- 'd'
Next Optind : 3
option : b
Next Optind : 4
option : c, Argument = cba
Next Optind : 7
os@os-virtual-machine:~/sys_programming/chapter_1$
os@os-virtual-machine:~/sys_programming/chapter_1$ ./ch1_6.out -a -d -b df -c cba
argc = 7
argv[0] = ./ch1_6.out
argv[1] = -a
argv[2] = -d
argv[3] = -b
argv[4] = df
argv[5] = -c
argv[6] = cba
os@os-virtual-machine:~/sys_programming/chapter_1$
```

예제 풀이

연습문제

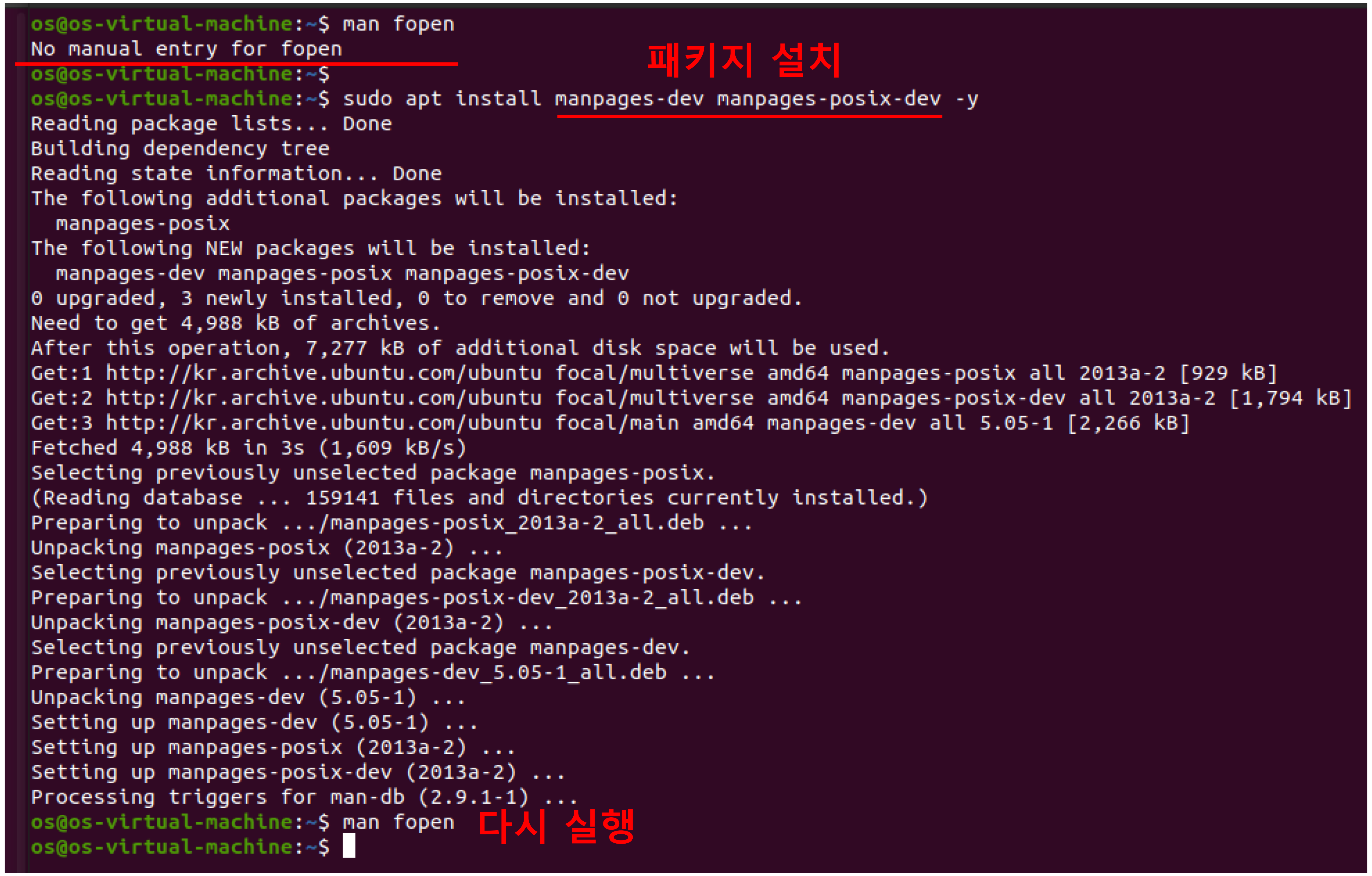
감사합니다.

CPS LAB

부록

Error 예시

man [명령어 | 라이브러리 | ..] 내용이 없을 때

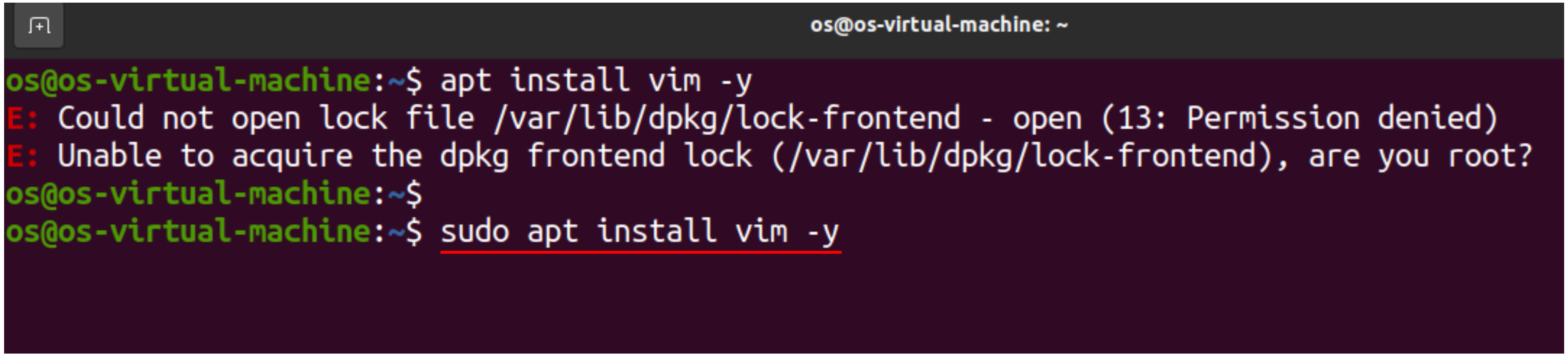


```
os@os-virtual-machine:~$ man fopen
No manual entry for fopen
os@os-virtual-machine:~$
os@os-virtual-machine:~$ sudo apt install manpages-dev manpages-posix-dev -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  manpages-posix
The following NEW packages will be installed:
  manpages-dev manpages-posix manpages-posix-dev
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,988 kB of archives.
After this operation, 7,277 kB of additional disk space will be used.
Get:1 http://kr.archive.ubuntu.com/ubuntu focal/multiverse amd64 manpages-posix all 2013a-2 [929 kB]
Get:2 http://kr.archive.ubuntu.com/ubuntu focal/multiverse amd64 manpages-posix-dev all 2013a-2 [1,794 kB]
Get:3 http://kr.archive.ubuntu.com/ubuntu focal/main amd64 manpages-dev all 5.05-1 [2,266 kB]
Fetched 4,988 kB in 3s (1,609 kB/s)
Selecting previously unselected package manpages-posix.
(Reading database ... 159141 files and directories currently installed.)
Preparing to unpack .../manpages-posix_2013a-2_all.deb ...
Unpacking manpages-posix (2013a-2) ...
Selecting previously unselected package manpages-posix-dev.
Preparing to unpack .../manpages-posix-dev_2013a-2_all.deb ...
Unpacking manpages-posix-dev (2013a-2) ...
Selecting previously unselected package manpages-dev.
Preparing to unpack .../manpages-dev_5.05-1_all.deb ...
Unpacking manpages-dev (5.05-1) ...
Setting up manpages-dev (5.05-1) ...
Setting up manpages-posix (2013a-2) ...
Setting up manpages-posix-dev (2013a-2) ...
Processing triggers for man-db (2.9.1-1) ...
os@os-virtual-machine:~$ man fopen
os@os-virtual-machine:~$
```

부록

Error 예시

sudo 권한 문제



os@os-virtual-machine: ~

```
os@os-virtual-machine:~$ apt install vim -y
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
os@os-virtual-machine:~$
os@os-virtual-machine:~$ sudo apt install vim -y
```

Cyber Physical Systems Laboratory

부록

참고내용

운영체제나 시스템 프로그래밍에서 Tool 활용하는 방법이 더 필요함

추가내용 (유용한 툴):

whatis, **man**, **whereis**,	사용설명서 (Document 읽는 능력)
strace,	프로그램과 커널 간의 시스템 호출 감시
tldr(설치필요)	간단한 명령어 사용법 출력