

[운영체제 실습 1. System Call, Process, Thread 과제]

학과 : 정보대학 컴퓨터학과

이름 : 정경륜

과제 1. 시스템 콜 실습

과제 1-1. 시스템 콜 과정 이해하기(5 점)

시스템 콜이란 응용 프로그램이 운영체제의 커널이 제공하는 서비스를 사용하고 싶을 때 커널에 접근하기 위한 인터페이스이다. 응용 프로그램이 직접 커널을 조작하는 것은 불가능 하기 때문에 시스템 콜의 도움을 받아 커널의 서비스를 이용할 수 있다.

사용자의 응용 프로그램에서 시스템 콜을 발생시키면 인터럽트가 발생하게 되는데 인터럽트는 CPU 가 프로그램을 실행하고 있을 때 예외상황이 발생하여 처리가 필요한 경우 이를 CPU 에 알려 처리를 할 수 있도록 하는 것을 의미한다. 응용 프로그램은 사용자 모드에서 돌아가는 반면에 시스템 콜은 커널에 접근해야 하기 때문에 커널 모드에서 진행되어야 한다. 따라서 이러한 모드 변환이 이루어져야 하기 때문에 응용 프로그램에서 시스템 콜 호출은 인터럽트를 발생 시킨다. 여기서의 인터럽트는 소프트웨어에 의한 트랩이다.

응용 프로그램에서 read 함수가 발생했을 때 곧 바로 시스템 콜을 호출하는 것은 아니다. 주로 라이브러리 내 함수를 사용하게 된다. C에서는 The GNU C 라이브러리를 사용한다. 라이브러리 내 read 함수는 0x80 번 인터럽트를 소프트웨어적으로 발생시킨다. 0x80 번 인터럽트는 커널 내의 IDT, Interrupt Descriptor Table 에 적힌 번호를 확인해보면 어떤 인터럽트를 뜻하는지 알 수 있다. 커널 내의 테이블에 접근해야 하기 때문에 CPU 는 사용자 모드에서 커널 모드로 바뀌게 되고 이제 IDT 에 접근할 수 있다.

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/* Vector 128 : legacy int80 syscall interface
 * Vectors 129 ... INVALIDATE_TLB_VECTOR_START-1 except 204 : device interrupts
 * Vectors INVALIDATE_TLB_VECTOR_START ... 255 : special interrupts
 *
 * 64-bit x86 has per CPU IDT tables, 32-bit has one shared IDT table.
 *
 * This file enumerates the exact layout of them:
 */

#define NMI_VECTOR                0x02
#define MCE_VECTOR                0x12

/*
 * IDT vectors usable for external interrupt sources start at 0x20.
 * (0x80 is the syscall vector, 0x30-0x3f are for ISA)
 */
#define FIRST_EXTERNAL_VECTOR    0x20
/*
 * We start allocating at 0x21 to spread out vectors evenly between
 * priority levels. (0x80 is the syscall vector)
 */
#define VECTOR_OFFSET_START      1

/*
 * Reserve the lowest usable vector (and hence lowest priority) 0x20 for
 * triggering cleanup after irq migration. 0x21-0x2f will still be used
 * for device interrupts.
 */
#define IRQ_MOVE_CLEANUP_VECTOR  FIRST_EXTERNAL_VECTOR

#define IA32_SYSCALL_VECTOR      0x80

/*
 * Vectors 0x30-0x3f are used for ISA interrupts.
 * round up to the next 16-vector boundary
 */
/ SYSCALL_VECTOR                50,14                17%

os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

    INTG(X86_TRAP_MF,                coprocessor_error),
    INTG(X86_TRAP_AC,                alignment_check),
    INTG(X86_TRAP_XF,                simd_coprocessor_error),

#ifdef CONFIG_X86_32
    TSKG(X86_TRAP_DF,                GDT_ENTRY_DOUBLEFAULT_TSS),
#else
    INTG(X86_TRAP_DF,                double_fault),
#endif
    INTG(X86_TRAP_DB,                debug),

#ifdef CONFIG_X86_MCE
    INTG(X86_TRAP_MC,                &machine_check),
#endif

    SYSG(X86_TRAP_OF,                overflow),
#ifdef defined(CONFIG_IA32_EMULATION)
    SYSG(IA32_SYSCALL_VECTOR,        entry_INT80_compat),
#elif defined(CONFIG_X86_32)
    SYSG(IA32_SYSCALL_VECTOR,        entry_INT80_32),
#endif
};

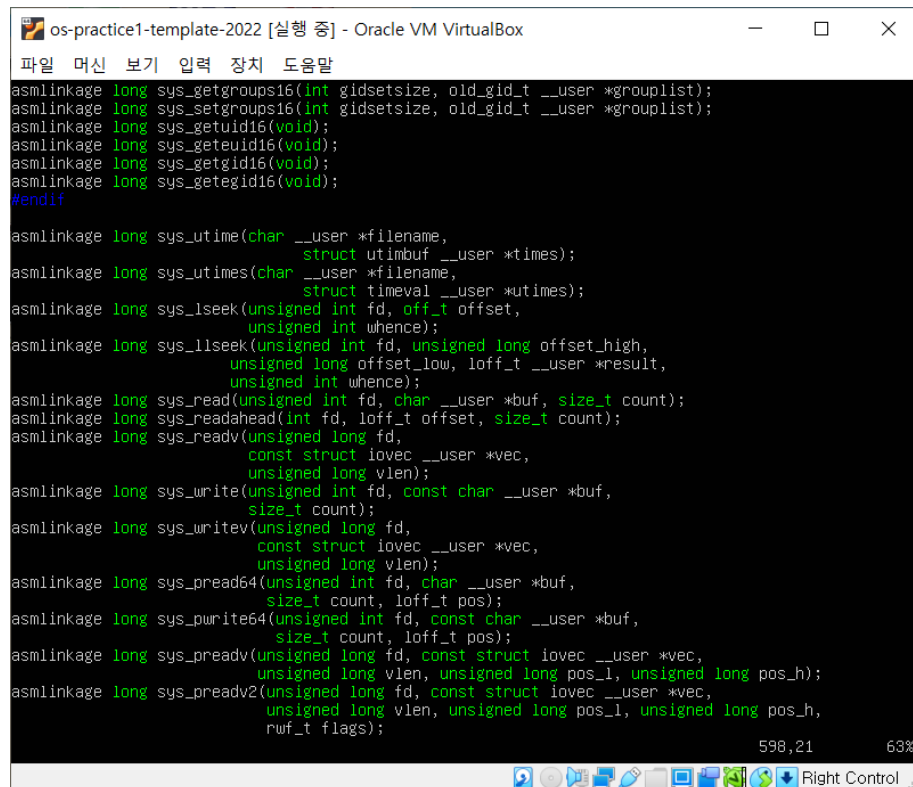
/*
 * The APIC and SMP idt entries
 */
static const __initconst struct idt_data apic_idts[] = {
#ifdef CONFIG_SMP
    INTG(RECHEDULE_VECTOR,            reschedule_interrupt),
    INTG(CALL_FUNCTION_VECTOR,        call_function_interrupt),
    INTG(CALL_FUNCTION_SINGLE_VECTOR, call_function_single_interrupt),
    INTG(IRQ_MOVE_CLEANUP_VECTOR,     irq_move_cleanup_interrupt),
    INTG(REBOOT_VECTOR,               reboot_interrupt),
#endif

#ifdef CONFIG_X86_THERMAL_VECTOR
search hit BOTTOM, continuing at TOP
102,13-20                25%
```

커널 내 IDT의 값을 살펴보면 0x80 번은 시스템 콜을 뜻하는 것을 알 수 있다. IDT의 0x80은 시스템 콜 테이블을 가리키는데 여기에 나타난 시스템 콜 번호를 살펴보면 어떤 시스템 콜을 호출하는지 파악할 수 있다.

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
#
# 64-bit system call numbers and entry vectors
#
# The format is:
# <number> <abi> <name> <entry point>
#
# The abi is "common", "64" or "x32" for this file.
#
0      common read          sys_read
1      common write         sys_write
2      common open          sys_open
3      common close         sys_close
4      common stat           sys_newstat
5      common fstat          sys_newfstat
6      common lstat          sys_newlstat
7      common poll           sys_poll
8      common lseek          sys_lseek
9      common mmap           sys_mmap
10     common mprotect       sys_mprotect
11     common munmap         sys_munmap
12     common brk            sys_brk
13     64 rt_sigaction        sys_rt_sigaction
14     common rt_sigprocmask  sys_rt_sigprocmask
15     64 rt_sigreturn        sys_rt_sigreturn/ptregs
16     64 ioctl              sys_ioctl
17     common pread64         sys_pread64
18     common pwrite64        sys_pwrite64
19     64 readv              sys_readv
20     64 writev             sys_writev
21     common access         sys_access
22     common pipe           sys_pipe
23     common select         sys_select
24     common sched_yield     sys_sched_yield
25     common mremap          sys_mremap
26     common msync           sys_msync
27     common mincore         sys_mincore
"syscall_64.tbl" 382L, 13259C
1,1 Top
```

앞서 말한대로 0x80 인터럽트는 시스템 콜 호출을 의미한다. 어떤 시스템 콜이 호출 됐는지 확인하기 위한 시스템 콜 번호와 시스템 콜 함수 이름이 적힌 테이블은 syscall_64.tbl 파일에 적혀있다. 시스템 콜 테이블을 보면 read 함수는 0 번으로 시스템 콜 테이블 번호가 지정되어 있는 것을 살펴볼 수 있다.



```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
asm linkage long sys_getgroups16(int gidsetsize, old_gid_t __user *grouplist);
asm linkage long sys_setgroups16(int gidsetsize, old_gid_t __user *grouplist);
asm linkage long sys_getuid16(void);
asm linkage long sys_geteuid16(void);
asm linkage long sys_getgid16(void);
asm linkage long sys_getegid16(void);
#endif

asm linkage long sys_utime(char __user *filename,
                          struct utimbuf __user *times);
asm linkage long sys_utimes(char __user *filename,
                          struct timeval __user *utimes);
asm linkage long sys_lseek(unsigned int fd, off_t offset,
                          unsigned int whence);
asm linkage long sys_llseek(unsigned int fd, unsigned long offset_high,
                          unsigned long offset_low, loff_t __user *result,
                          unsigned int whence);
asm linkage long sys_read(unsigned int fd, char __user *buf, size_t count);
asm linkage long sys_readahead(int fd, loff_t offset, size_t count);
asm linkage long sys_readv(unsigned long fd,
                          const struct iovec __user *vec,
                          unsigned long vlen);
asm linkage long sys_write(unsigned int fd, const char __user *buf,
                          size_t count);
asm linkage long sys_writev(unsigned long fd,
                          const struct iovec __user *vec,
                          unsigned long vlen);
asm linkage long sys_pread64(unsigned int fd, char __user *buf,
                          size_t count, loff_t pos);
asm linkage long sys_pwrite64(unsigned int fd, const char __user *buf,
                          size_t count, loff_t pos);
asm linkage long sys_preadv(unsigned long fd, const struct iovec __user *vec,
                          unsigned long vlen, unsigned long pos_l, unsigned long pos_h);
asm linkage long sys_preadv2(unsigned long fd, const struct iovec __user *vec,
                          unsigned long vlen, unsigned long pos_l, unsigned long pos_h,
                          rwf_t flags);
598,21 63%
```

위에서 read 시스템 콜은 0 번으로 지정되어 있고 함수명은 sys_read 인 것을 확인할 수 있었다. sys_read 가 어떻게 정의되어 있는 지 확인하기 위해 syscalls.h 를 참고하였다. sys_read 는 위의 화면에서 볼 수 있듯이 3 개의 인자를 받는 함수로 구성되어있는 것을 확인할 수 있다. 따라서 read 함수 호출 시 시스템 콜은 시스템 콜 테이블 번호에 적힌 read 시스템 콜 함수를 syscalls.h 을 참고해 실행하게 되는 것을 알 수 있다.

과제 1-2. 새로운 시스템 콜 추가하기(7 점)

a. 시스템 콜 번호를 등록한 부분 캡처 (syscall_64.tbl)

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말
322    64    execveat    sys_execveat/ptregs
323    common userfaultfd  sys_userfaultfd
324    common membarrier  sys_membarrier
325    common mlock2    sys_mlock2
326    common copy_file_range sys_copy_file_range
327    64    preadv2    sys_preadv2
328    64    pwritev2   sys_pwritev2
329    common pkey_mprotect sys_pkey_mprotect
330    common pkey_alloc  sys_pkey_alloc
331    common pkey_free   sys_pkey_free
332    common statx     sys_statx
333    common print_student_id sys_print_student_id
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
512    x32    rt_sigaction  compat_sys_rt_sigaction
513    x32    rt_sigreturn sys32_x32_rt_sigreturn
514    x32    ioctl       compat_sys_ioctl
515    x32    readv       compat_sys_readv
516    x32    writev      compat_sys_writev
517    x32    recvfrom    compat_sys_recvfrom
518    x32    sendmsg     compat_sys_sendmsg
519    x32    recvmsg     compat_sys_recvmsg
520    x32    execve      compat_sys_execve/ptregs
521    x32    ptrace      compat_sys_ptrace
522    x32    rt_sigpending compat_sys_rt_sigpending
523    x32    rt_sigtimedwait compat_sys_rt_sigtimedwait
524    x32    rt_sigqueueinfo compat_sys_rt_sigqueueinfo
525    x32    sigaltstack compat_sys_sigaltstack
526    x32    timer_create compat_sys_timer_create
527    x32    mq_notify   compat_sys_mq_notify
528    x32    kexec_load  compat_sys_kexec_load
529    x32    waitid      compat_sys_waitid
530    x32    set_robust_list compat_sys_set_robust_list
531    x32    get_robust_list compat_sys_get_robust_list
342,60    95%
```

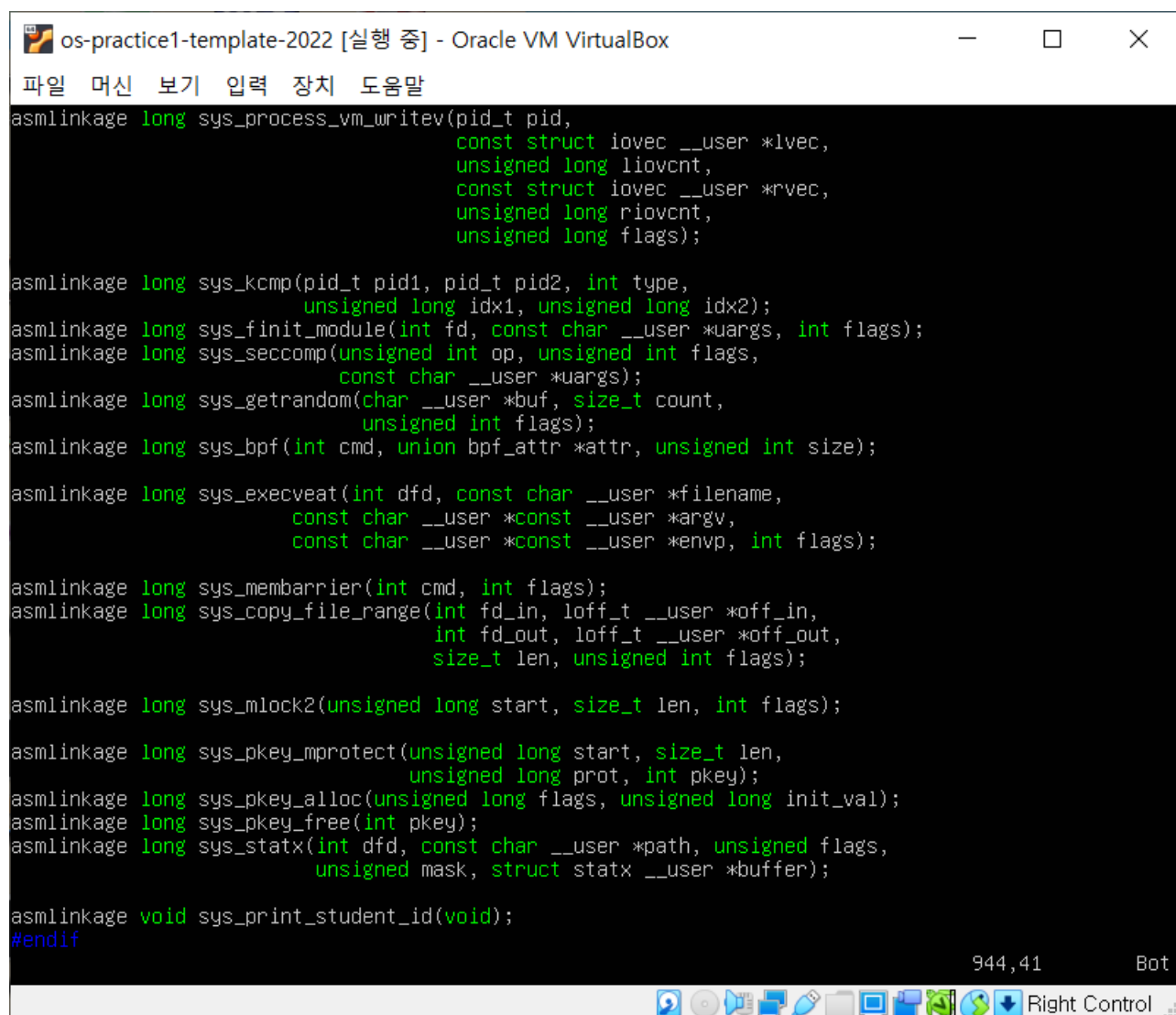
→ 새로 추가할 시스템 콜 번호를 333 으로 등록하였다.

b. 새로 추가한 시스템 콜 함수 코드 캡처 (new_syscall.c)

[illegible]

→ 새로 추가한 시스템 콜 함수를 구현하였다.

c. 시스템 콜 함수를 선언한 부분 캡처 (syscalls.h)



The screenshot shows a window titled "os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox". The window contains a code editor with C code for system call declarations. The code is as follows:

```
asmlinkage long sys_process_vm_writev(pid_t pid,
                                     const struct iovec __user *lvec,
                                     unsigned long liovcnt,
                                     const struct iovec __user *rvec,
                                     unsigned long riovcnt,
                                     unsigned long flags);

asmlinkage long sys_kcmp(pid_t pid1, pid_t pid2, int type,
                        unsigned long idx1, unsigned long idx2);
asmlinkage long sys_finit_module(int fd, const char __user *uargs, int flags);
asmlinkage long sys_seccomp(unsigned int op, unsigned int flags,
                           const char __user *uargs);
asmlinkage long sys_getrandom(char __user *buf, size_t count,
                             unsigned int flags);
asmlinkage long sys_bpf(int cmd, union bpf_attr *attr, unsigned int size);

asmlinkage long sys_execveat(int dfd, const char __user *filename,
                             const char __user *const __user *argv,
                             const char __user *const __user *envp, int flags);

asmlinkage long sys_membarrier(int cmd, int flags);
asmlinkage long sys_copy_file_range(int fd_in, loff_t __user *off_in,
                                    int fd_out, loff_t __user *off_out,
                                    size_t len, unsigned int flags);

asmlinkage long sys_mlock2(unsigned long start, size_t len, int flags);

asmlinkage long sys_pkey_mprotect(unsigned long start, size_t len,
                                  unsigned long prot, int pkey);
asmlinkage long sys_pkey_alloc(unsigned long flags, unsigned long init_val);
asmlinkage long sys_pkey_free(int pkey);
asmlinkage long sys_statx(int dfd, const char __user *path, unsigned flags,
                          unsigned mask, struct statx __user *buffer);

asmlinkage void sys_print_student_id(void);
#endif
```

The bottom right corner of the window shows the coordinates "944,41" and the text "Bot". The bottom status bar contains icons for various system functions and the text "Right Control".

→ 새로 추가할 시스템 콜 함수를 선언하였다.

d. 사용자 어플리케이션 코드 캡처 (assignment.c)

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

#include <linux/unistd.h>

int main(void)
{
    // TODO: Write your code here
    syscall(333);
    return 0;
}

"assignment.c" 8L, 104C
6,12-19  A11
```

→ 앞서 구현한 시스템 콜 함수를 사용한 사용자 영역 프로그램을 작성하였다.

e. 결과 캡처

```
[ 484.325026] My student id is 2018320142
os-practice: ~
```

→ 학번의 잘 출력되는 것을 볼 수 있다.

과제 2. 프로세스, 스레드 실습

과제 2-1. 프로세스 퀴즈(5 문제, 각 1 점)

quiz 01

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/**
 * Basic fork() Usage 1.
 *
 * By walking through this example you'll learn:
 * - How to use fork().
 * - How to distinguish parent and child process.
 * - What happens to variables that reside in various scopes.
 */

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

const int SEVEN_AND_A_HALF_MILLION_YEARS = 3;
const int A_DAY = 1;

// Allocated in data segment.
static int the_answer = 0;

int main(int argc, char* argv[]){
    // Allocated in stack segment.
    int arthur = 0;

    pid_t pid;

    switch(pid = fork()){
        default:
            // HINT: The parent process should fall into this scope.
            the_answer = 42;
            arthur = 6 * 9;
            sleep(SEVEN_AND_A_HALF_MILLION_YEARS);
            break;
        case 0:
            // HINT: The child process should fall into this scope.
            sleep(A_DAY * 2);
    }

    printf("My pid is %d (0 is parent, non-0 is child), the answer is %d, and arthur is %d.\n",
           (long) getpid(), the_answer, arthur);
    return 0;
}
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

// Allocated in data segment.
static int the_answer = 0;

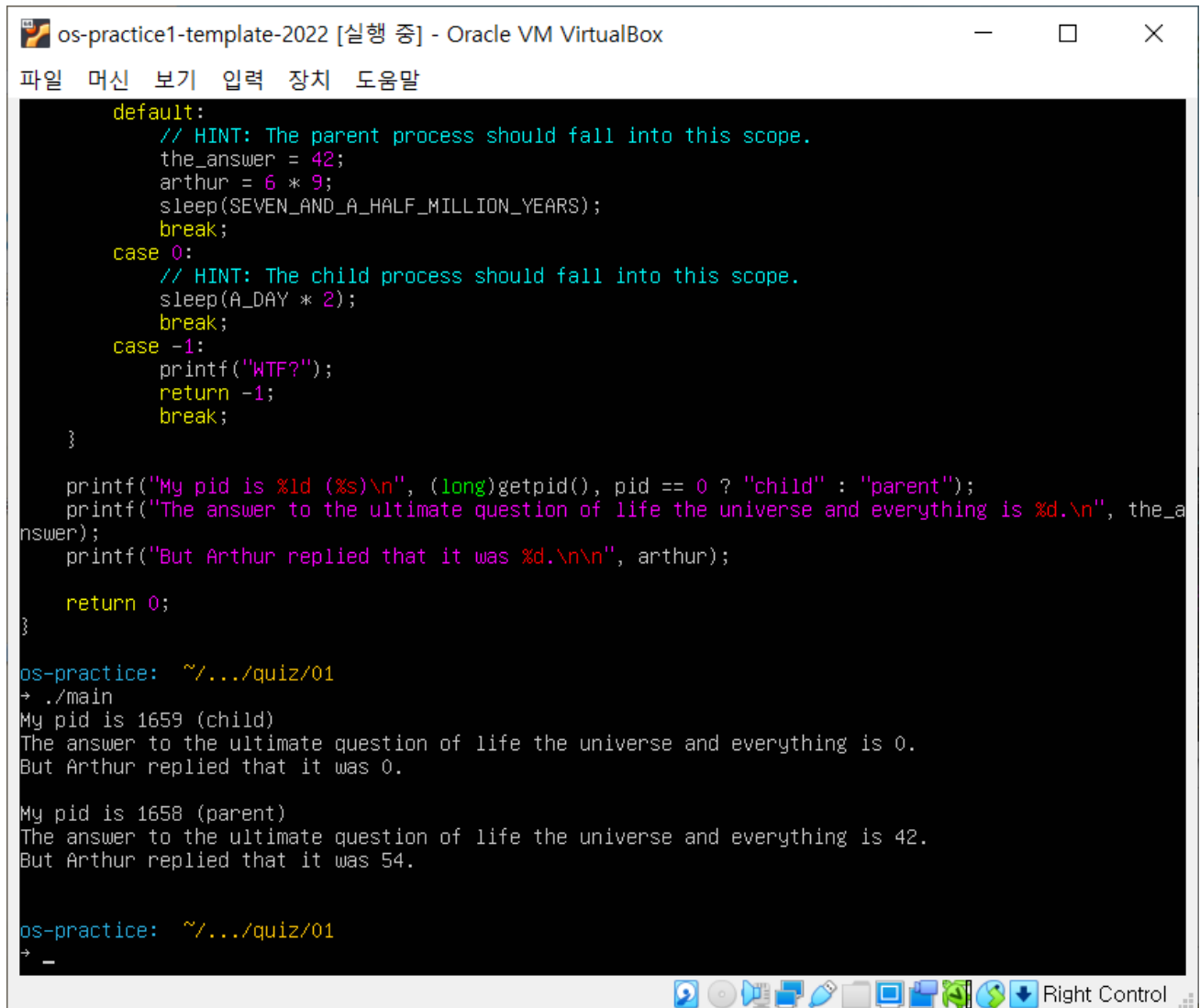
int main(int argc, char* argv[]){
    // Allocated in stack segment.
    int arthur = 0;

    pid_t pid;

    switch(pid = fork()){
        default:
            // HINT: The parent process should fall into this scope.
            the_answer = 42;
            arthur = 6 * 9;
            sleep(SEVEN_AND_A_HALF_MILLION_YEARS);
            break;
        case 0:
            // HINT: The child process should fall into this scope.
            sleep(A_DAY * 2);
            break;
        case -1:
            printf("WTF?\n");
            return -1;
            break;
    }

    printf("My pid is %d (0 is parent, non-0 is child), the answer is %d, and arthur is %d.\n",
           (long) getpid(), the_answer, arthur);
    return 0;
}
```

Result



```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

default:
    // HINT: The parent process should fall into this scope.
    the_answer = 42;
    arthur = 6 * 9;
    sleep(SEVEN_AND_A_HALF_MILLION_YEARS);
    break;
case 0:
    // HINT: The child process should fall into this scope.
    sleep(A_DAY * 2);
    break;
case -1:
    printf("WTF?");
    return -1;
    break;
}

printf("My pid is %ld (%s)\n", (long) getpid(), pid == 0 ? "child" : "parent");
printf("The answer to the ultimate question of life the universe and everything is %d.\n", the_answer);
printf("But Arthur replied that it was %d.\n\n", arthur);

return 0;
}

os-practice: ~/.../quiz/01
→ ./main
My pid is 1659 (child)
The answer to the ultimate question of life the universe and everything is 0.
But Arthur replied that it was 0.

My pid is 1658 (parent)
The answer to the ultimate question of life the universe and everything is 42.
But Arthur replied that it was 54.

os-practice: ~/.../quiz/01
→ _
```

quiz 02

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/**
 * Basic fork() Usage 2.
 *
 * By walking through this example you'll learn:
 * - How to use fork().
 * - How to distinguish parent and child process.
 * - What happens to variables that reside in various scopes.
 */

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(int argc, char* argv[]){
    pid_t pid;
    int val = 1;

    printf("The value is %d\n", val);

    pid = fork();

    if(pid > 0){
        // HINT: The parent process should fall into this scope.
        val++;
    } else if(pid == 0) {
        // HINT: The child process should fall into this scope.
        sleep(1);
        val--;
    } else {
        printf("WTF?");
        return -1;
    }

    printf("The value is %d in %s.\n", val, pid == 0 ? "child" : "parent");
}

1,1 Top
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

int main(int argc, char* argv[]){
    pid_t pid;
    int val = 1;

    printf("The value is %d\n", val);

    pid = fork();

    if(pid > 0){
        // HINT: The parent process should fall into this scope.
        val++;
    } else if(pid == 0) {
        // HINT: The child process should fall into this scope.
        sleep(1);
        val--;
    } else {
        printf("WTF?");
        return -1;
    }

    printf("The value is %d in %s.\n", val, pid == 0 ? "child" : "parent");

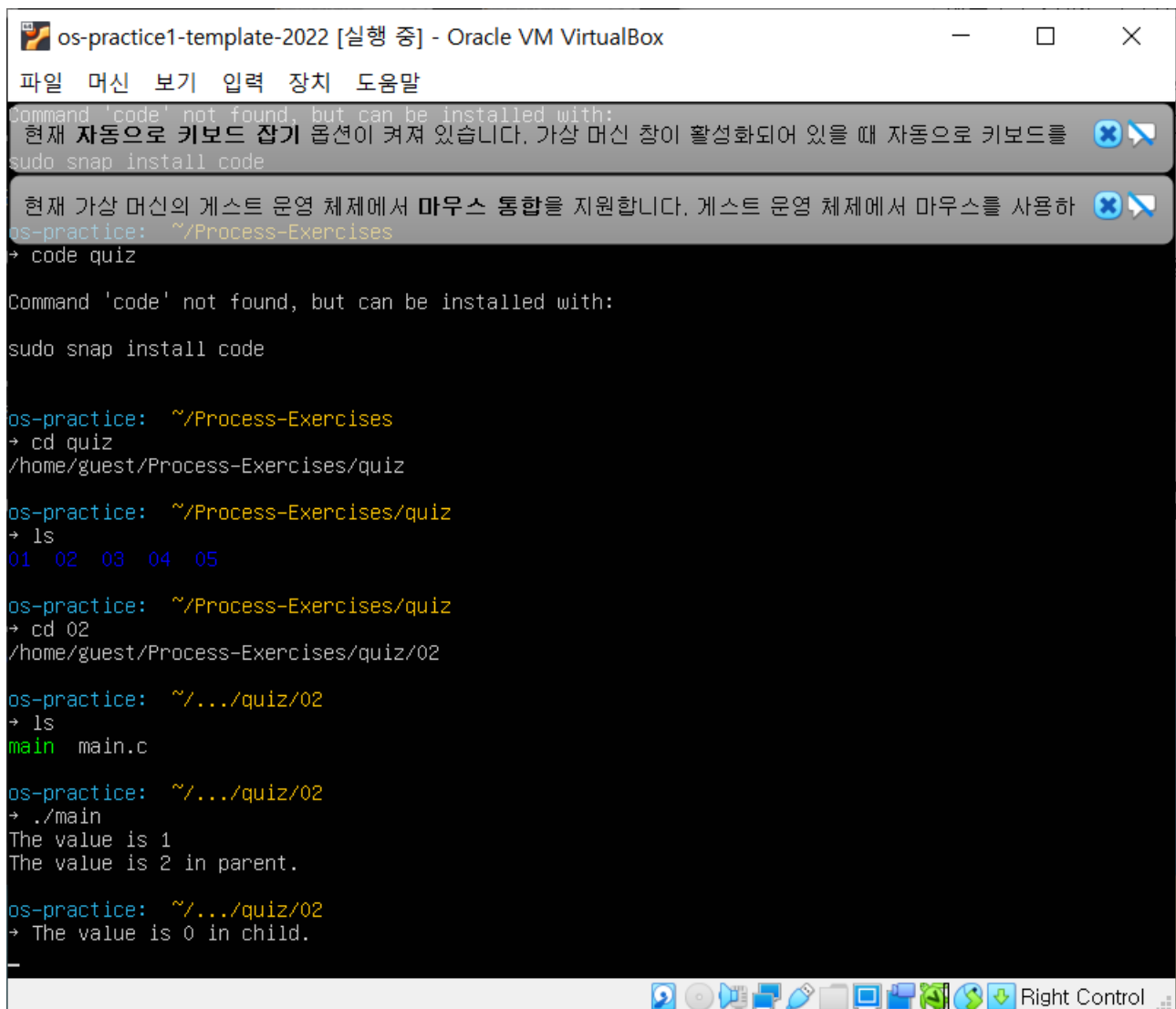
    return 0;
}

/**
Expected output:

The value is 1
The value is 2 in parent.
The value is 0 in child. // It will be printed on the command line.
*/

49,1 Bot
```

Result



```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

Command 'code' not found, but can be installed with:
현재 자동으로 키보드 잡기 옵션이 켜져 있습니다. 가상 머신 창이 활성화되어 있을 때 자동으로 키보드를
sudo snap install code

현재 가상 머신의 게스트 운영 체제에서 마우스 통합을 지원합니다. 게스트 운영 체제에서 마우스를 사용하
os-practice: ~/Process-Exercises
→ code quiz

Command 'code' not found, but can be installed with:
sudo snap install code

os-practice: ~/Process-Exercises
→ cd quiz
/home/guest/Process-Exercises/quiz

os-practice: ~/Process-Exercises/quiz
→ ls
01 02 03 04 05

os-practice: ~/Process-Exercises/quiz
→ cd 02
/home/guest/Process-Exercises/quiz/02

os-practice: ~/.../quiz/02
→ ls
main main.c

os-practice: ~/.../quiz/02
→ ./main
The value is 1
The value is 2 in parent.

os-practice: ~/.../quiz/02
→ The value is 0 in child.
```

quiz 03

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/**
 * Basic execl() Usage.
 * By walking through this example you'll learn:
 * - How to use execl().
 * - What happens to the process that invoked execl().
 */

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

int main(int argc, char* argv[]){
    printf("%s executing `ls -l`.\n", "Before");

    // HINT: The /bin/ls -l should be executed.
    execl("/bin/ls", "ls", "-l", NULL);

    printf("%s executing `ls -l`.\n", "After");

    return 0;
}

/*
Expected output:
Before executing `ls -l`.
total 20
-rwxr-xr-x 1 root root 15484 Apr 25 01:37 main
-rw-r--r-- 1 root root  453 Apr 25 01:37 main.c
*/
"main.c" 34L, 608C 1,1 All
```

Result

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

    val--;
} else {
    printf("WTF?");
    return -1;
}

printf("The value is %d in %s.\n", val, pid == 0 ? "child" : "parent");

return 0;
}

os-practice: ~/.../quiz/02
→ cd ..

os-practice: ~/Process-Exercises/quiz
→ ls
01 02 03 04 05

os-practice: ~/Process-Exercises/quiz
→ cd 03
/home/guest/Process-Exercises/quiz/03

os-practice: ~/.../quiz/03
→ ls
main main.c

os-practice: ~/.../quiz/03
→ ./main
Before executing `ls -l`.
total 16
-rwxrwxr-x 1 guest guest 8344 Apr 13 15:51 main
-rw-rw-r-- 1 guest guest  608 Apr 13 15:51 main.c

os-practice: ~/.../quiz/03
→
```

quiz 04

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일 머신 보기 입력 장치 도움말

/**
 * Keep Running Your Process.
 *
 * By walking through this example you'll learn:
 * - How to use execl() while keep your process using fork().
 */

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(int argc, char* argv[]){
    pid_t pid = fork();

    switch (pid)
    {
        default:
            // HINT: The parent process should fall into this scope.
            printf("I'm your father.\n");
            sleep(3);
            break;

        case 0:
            sleep(1);
            // HINT: The child process should fall into this scope.
            printf("I'm sorry, but I'm not Luke. I'm...");
            fflush(stdout);

            sleep(1); // for dramatic effect

            // HINT: The /usr/bin/whoami should be executed.
            execl("/usr/bin/whoami", "whoami", NULL);

        case -1:
            // Error handling
    }

    return 0;
}
```

"main.c" 51L, 1013C 13,1 Top

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일 머신 보기 입력 장치 도움말

switch (pid)
{
    default:
        // HINT: The parent process should fall into this scope.
        printf("I'm your father.\n");
        sleep(3);
        break;

    case 0:
        sleep(1);
        // HINT: The child process should fall into this scope.
        printf("I'm sorry, but I'm not Luke. I'm...");
        fflush(stdout);

        sleep(1); // for dramatic effect

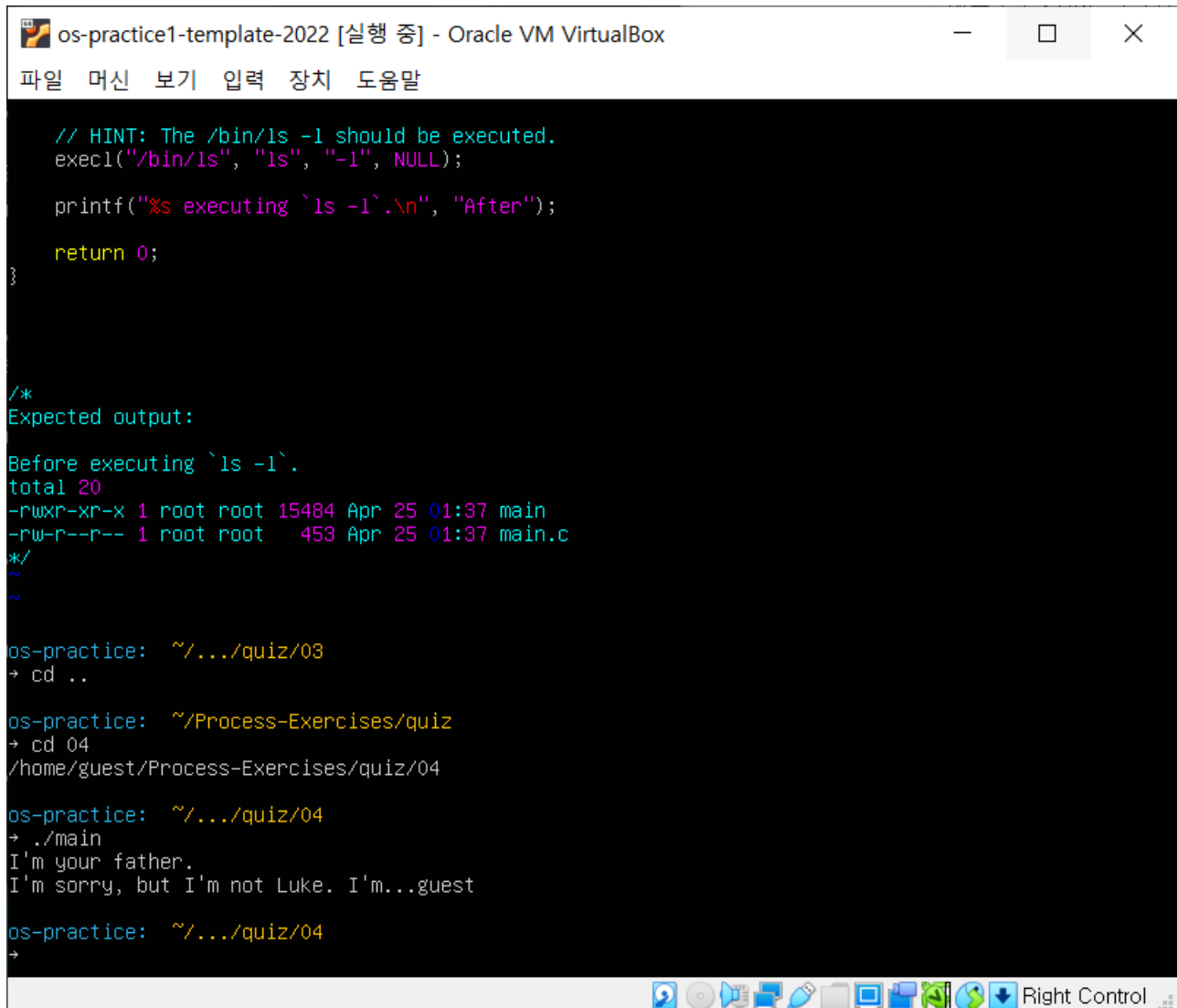
        // HINT: The /usr/bin/whoami should be executed.
        execl("/usr/bin/whoami", "whoami", NULL);

    case -1:
        printf("WTF?");
        return -1;
        break;
}

/**
Expected output:
I'm your father.
I'm sorry, but I'm not Luke. I'm...ubuntu
*/
```

51,1 Bot

Result



```
// HINT: The /bin/ls -l should be executed.
execl("/bin/ls", "ls", "-l", NULL);

printf("%s executing `ls -l`.\\n", "After");

return 0;
}

/*
Expected output:

Before executing `ls -l`.
total 20
-rwxr-xr-x 1 root root 15484 Apr 25 01:37 main
-rw-r--r-- 1 root root 453 Apr 25 01:37 main.c
*/
~
~

os-practice: ~/.../quiz/03
→ cd ..

os-practice: ~/Process-Exercises/quiz
→ cd 04
/home/guest/Process-Exercises/quiz/04

os-practice: ~/.../quiz/04
→ ./main
I'm your father.
I'm sorry, but I'm not Luke. I'm...guest

os-practice: ~/.../quiz/04
→
```

quiz 05

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/**
 * Allow Perseverance to Finish Its Work.
 *
 * By walking through this example you'll learn:
 * - In the parent process, wait for the child process to complete its work.
 * - How to use wait().
 */

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char* argv[]){
    pid_t pid;
    int status;

    printf("It breaks my heart to see my fellow zealots suffer on the battlefield.\n");
    printf("But what if we dragoons went to their rescue?\n");

    printf("Duh! ");
    fflush(stdout);

    pid = fork();

    if(pid > 0){
        // HINT: The parent process should fall into this scope.
        wait(&status);
        printf("Goon!\n");
    } else if(pid == 0){
        // HINT: The child process should fall into this scope.
        printf("Ra! ");
    } else {
        printf("WTF?");
        return -1;
    }

    "main.c" 51L, 1058C                                     14,0-1      Top
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

    pid_t pid;
    int status;

    printf("It breaks my heart to see my fellow zealots suffer on the battlefield.\n");
    printf("But what if we dragoons went to their rescue?\n");

    printf("Duh! ");
    fflush(stdout);

    pid = fork();

    if(pid > 0){
        // HINT: The parent process should fall into this scope.
        wait(&status);
        printf("Goon!\n");
    } else if(pid == 0){
        // HINT: The child process should fall into this scope.
        printf("Ra! ");
    } else {
        printf("WTF?");
        return -1;
    }

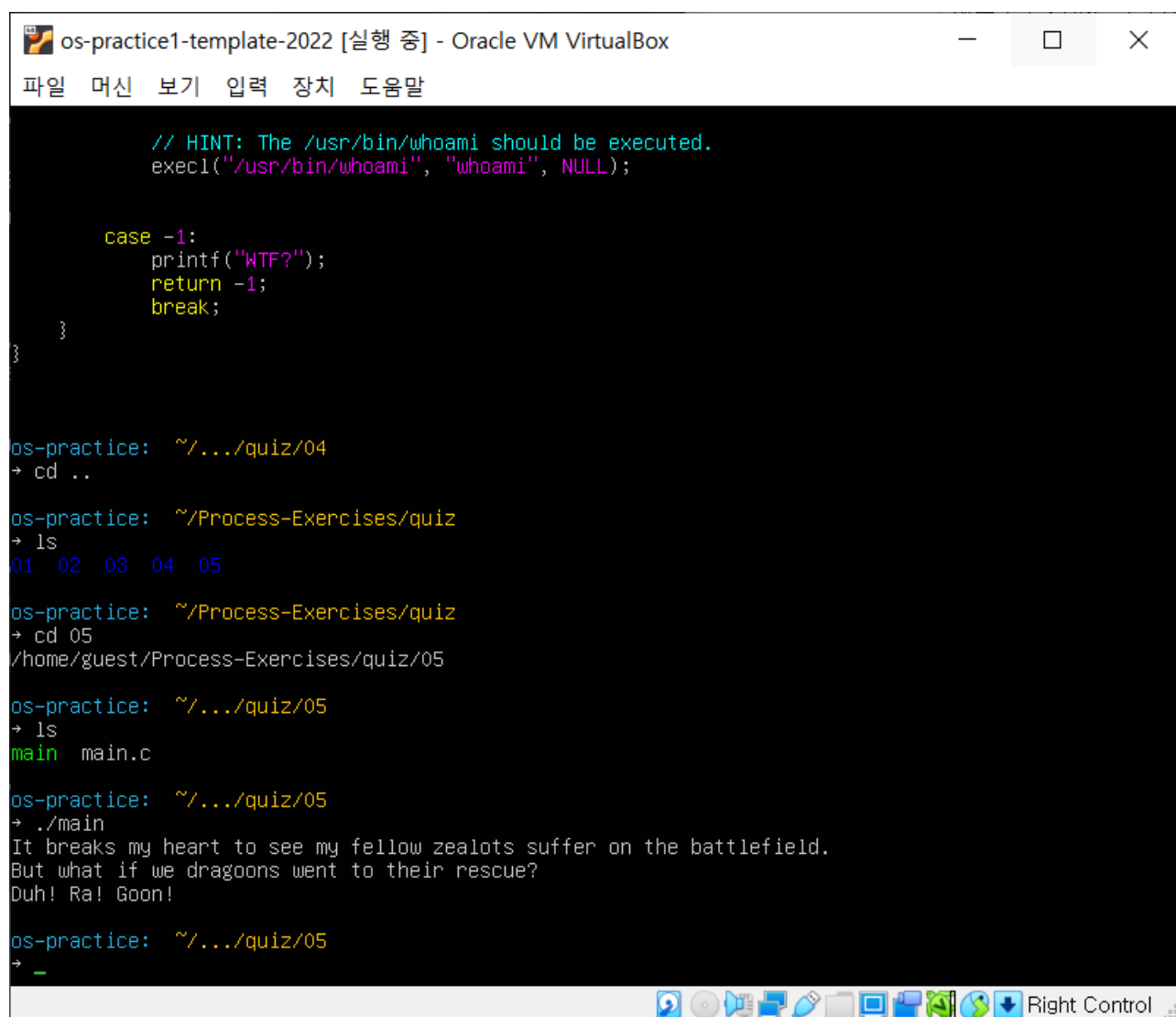
    return 0;
}

/**
Expected output:

It breaks my heart to see my fellow zealots suffer on the battlefield.
But what if we dragoons went to their rescue?
Duh! Ra! Goon!
*/

51,1      Bot
```


Result



```
// HINT: The /usr/bin/whoami should be executed.
execl("/usr/bin/whoami", "whoami", NULL);

case -1:
    printf("WTF?");
    return -1;
    break;
}
}
```

os-practice: ~/.../quiz/04
→ cd ..

os-practice: ~/Process-Exercises/quiz
→ ls
01 02 03 04 05

os-practice: ~/Process-Exercises/quiz
→ cd 05
/home/guest/Process-Exercises/quiz/05

os-practice: ~/.../quiz/05
→ ls
main main.c

os-practice: ~/.../quiz/05
→ ./main
It breaks my heart to see my fellow zealots suffer on the battlefield.
But what if we dragoons went to their rescue?
Duh! Ra! Goon!

os-practice: ~/.../quiz/05
→ -

과제 2-2. 스레드 퀴즈(3 문제, 각 1 점)

quiz 01

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/**
 * Basic POSIX Thread (pthread) Usage 1.
 *
 * By walking through this example you'll learn:
 * - How to use pthread_create().
 * - How to use pthread_exit().
 * - How to use pthread_join().
 */

#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/wait.h>

void* ninja(void* arg){
    printf("Who's there?");
    fflush(stdout);

    pthread_exit("ninja");
}

int main(int argc, char* argv[]){
    pthread_t tid;
    char* from = "";

    printf("knock knock.\n");

    // HINT: The thread that runs `ninja` should be created.
    int status = pthread_create(&tid, NULL, ninja, NULL);

    if(status != 0){
        printf("WTF?");
        return -1;
    }
}
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

int main(int argc, char* argv[]){
    pthread_t tid;
    char* from = "";

    printf("knock knock.\n");

    // HINT: The thread that runs `ninja` should be created.
    int status = pthread_create(&tid, NULL, ninja, NULL);

    if(status != 0){
        printf("WTF?");
        return -1;
    }

    // HINT: The main thread should not be exited until `ninja` has finished.
    pthread_join(tid, (void *)&from);

    // HINT: The variable `from` should not be empty.
    printf(" - from %s\n", (char *)&from);

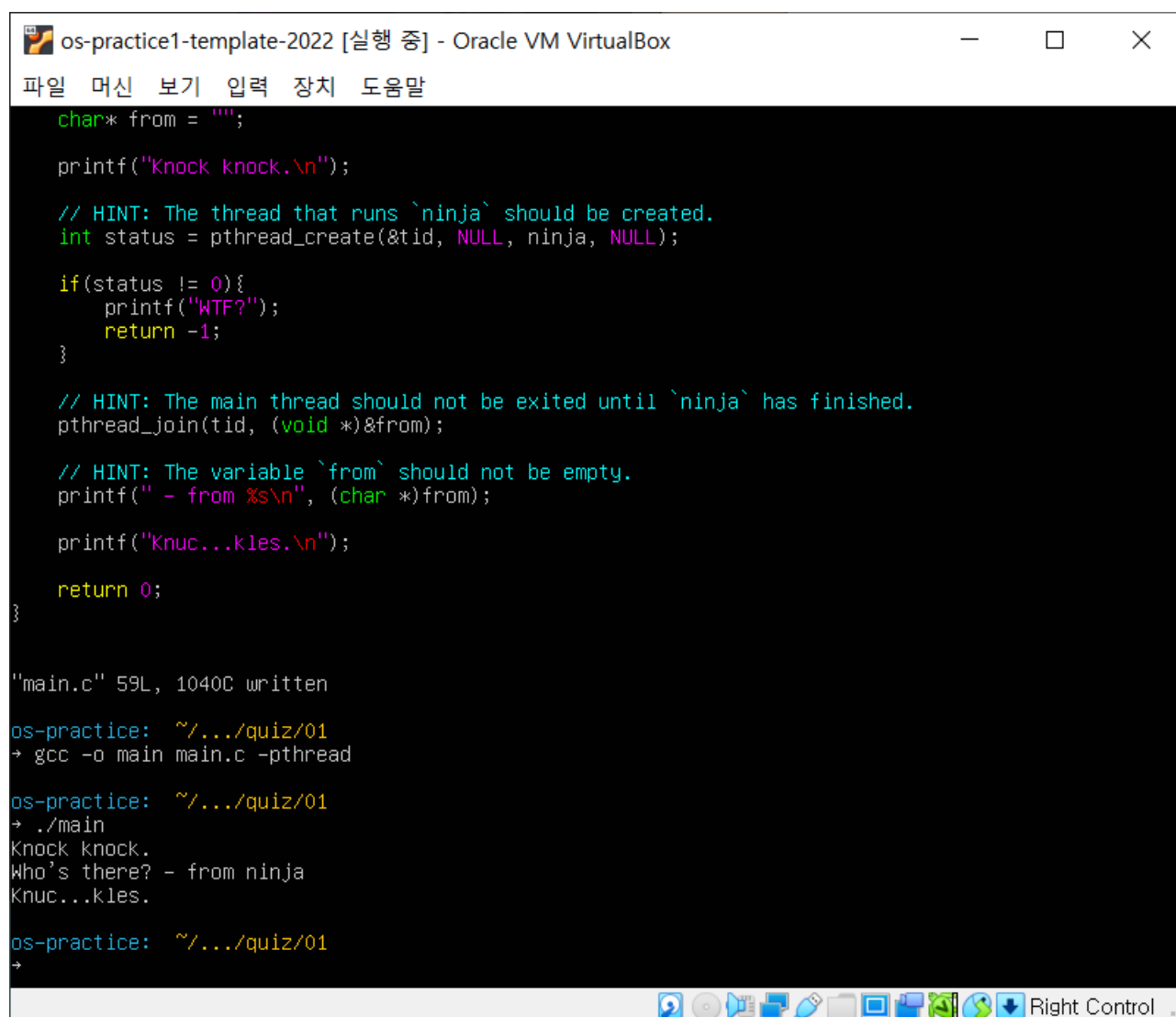
    printf("Knuc...kles.\n");

    return 0;
}

/**
Expected output:

Knock knock.
Who's there? - from ninja
Knuc...kles.
*/
```

Result



```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

char* from = "";

printf("Knock knock.\n");

// HINT: The thread that runs `ninja` should be created.
int status = pthread_create(&tid, NULL, ninja, NULL);

if(status != 0){
    printf("WTF?");
    return -1;
}

// HINT: The main thread should not be exited until `ninja` has finished.
pthread_join(tid, (void *)&from);

// HINT: The variable `from` should not be empty.
printf("- from %s\n", (char *)from);

printf("Knuc...kles.\n");

return 0;
}

"main.c" 59L, 1040C written
os-practice: ~/.../quiz/01
→ gcc -o main main.c -pthread

os-practice: ~/.../quiz/01
→ ./main
Knock knock.
Who's there? - from ninja
Knuc...kles.

os-practice: ~/.../quiz/01
→
```

quiz 02

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/**
 * Basic POSIX Thread (pthread) Usage 2.
 *
 * By walking through this example you'll learn:
 * - How to pass value to thread when create it.
 * - How to handles multiple threads.
 * - What happens to variables that reside in various scopes.
 */

#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/wait.h>

#define NUM_THREADS 3

static int global;

void print_addr(void* g, void* m, void* t, void* ts){
    printf("%p\t%p\t%p\t%p\n", g, m, t, ts);
}

void* worker(void* arg){
    int thread;
    static int thread_static;

    print_addr(&global, arg, &thread, &thread_static);

    pthread_exit(NULL);
}

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

    pthread_t tids[NUM_THREADS];

    1,1      Top
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

}

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

    pthread_t tids[NUM_THREADS];
    int status;

    printf("global\t\tmain\t\tthread\t\tthread-static\n");
    print_addr(&global, &main, 0, 0);

    for(int i = 0; i < NUM_THREADS; i++){
        // HINT: The thread that runs `worker` should be created.
        // HINT: The address of variable `main_static` should be passed
        //       when thread created.
        // HINT: Each thread descriptor should be stored appropriately.
        status = pthread_create(&tids[i], NULL, worker, &main_static);

        if(status != 0){
            printf("WTF?");
            return -1;
        }
    }

    // HINT: The main thread should not be exited until all `worker`s have finished.
    for(int i=0; i < NUM_THREADS; i++){
        pthread_join(tids[i], NULL);
    }

    return 0;
}

38,0-1      76%
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

print_addr(&global, &main, 0, 0);

for(int i = 0; i < NUM_THREADS; i++){
    // HINT: The thread that runs `worker` should be created.
    // HINT: The address of variable `main_static` should be passed
    //        when thread created.
    // HINT: Each thread descriptor should be stored appropriately.
    status = pthread_create(&tids[i], NULL, worker, &main_static);

    if(status != 0){
        printf("WTF?");
        return -1;
    }
}

// HINT: The main thread should not be exited until all `worker`s have finished.
for(int i=0; i < NUM_THREADS; i++){
    pthread_join(tids[i], NULL);
}

return 0;
}

/*
Expected output:
global      main      thread      thread-static
0x4dd030     0x4da249     (nil) (nil)
0x4dd030     0x4dd038     0xb74db34c  0x4dd034
0x4dd030     0x4dd038     0xb6cda34c  0x4dd034
0x4dd030     0x4dd038     0xb7cdc34c  0x4dd034
*/
75,1 Bot
```

Result

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

printf("- from %s\n", (char *)from);

printf("Knuc...kles.\n");

return 0;
}

"main.c" 59L, 1039C written

os-practice: ~/.../quiz/01
+ gcc -o main main.c
/tmp/cc8AgpCt.o: In function `main':
main.c:(.text+0x86): undefined reference to `pthread_create'
main.c:(.text+0xba): undefined reference to `pthread_join'
collect2: error: ld returned 1 exit status

os-practice: ~/.../quiz/01
+ cd ..

os-practice: ~/Threads-Exercises/quiz
+ cd 02
/home/guest/Threads-Exercises/quiz/02

os-practice: ~/.../quiz/02
+ ls
main main.c

os-practice: ~/.../quiz/02
+ ./main
global      main      thread      thread-static
0x5579d80d4014 0x5579d7ed389f (nil) (nil)
0x5579d80d4014 0x5579d80d401c 0x7f9d982ddee4 0x5579d80d4018
0x5579d80d4014 0x5579d80d401c 0x7f9d98adeee4 0x5579d80d4018
0x5579d80d4014 0x5579d80d401c 0x7f9d97adcee4 0x5579d80d4018

os-practice: ~/.../quiz/02
+ -
```

quiz 03

Code

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

/**
 * There's Always a Slight Duplication of Efforts 1.
 *
 * By walking through this example you'll learn:
 * - How to use pthread_join().
 * - What happens when multiple threads try to reference the same memory block.
 */

#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sched.h>

int stick_this_thread_to_core(int core_id);

#define NUM_THREADS 100
#define NUM_TASKS 100000

static int cnt = 0;

void* worker(void* arg){
    stick_this_thread_to_core((int)arg);
    int progress;

    for(int i = 0; i < NUM_TASKS; i++){
        progress = cnt++;
    }

    pthread_exit((void*)progress);
}

int main(int argc, char* argv[]){
    pthread_t tids[NUM_THREADS];
    int status;

    2,52      Top
Right Control
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

int main(int argc, char* argv[]){
    pthread_t tids[NUM_THREADS];
    int status;
    int progress = 0;

    for(int i = 0; i < NUM_THREADS; i++){
        // HINT: The thread that runs `worker` should be created.
        // HINT: The address of variable `i` should be passed when thread created.
        // HINT: Each thread descriptor should be stored appropriately.
        status = pthread_create(&tids[i], NULL, worker, (void *)&i);

        if(status != 0){
            printf("WTF?");
            return -1;
        }
    }

    // HINT: The main thread should not be exited until all `worker`s have finished.
    for(int i = 0; i < NUM_THREADS; i++){
        pthread_join(tids[i], (void *)&progress);
        // HINT: The variable `progress` should not be 0.
        printf("\r%d ", progress);

        fflush(stdout);
        usleep(10*1000); // 10ms
    }

    printf("\nexpected: %d\n", NUM_THREADS * NUM_TASKS);
    printf("result: %d\n", cnt);

    return 0;
}

63,0-1      52%
Right Control
```

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

printf("result: %d\n", cnt);

return 0;
}

int stick_this_thread_to_core(int core_id) {
    int num_cores = sysconf(_SC_NPROCESSORS_ONLN);

    cpu_set_t cpuset;
    CPU_ZERO(&cpuset);
    CPU_SET(core_id % num_cores, &cpuset);

    pthread_t current_thread = pthread_self();
    return pthread_setaffinity_np(current_thread, sizeof(cpu_set_t), &cpuset);
}

/*
1.Expected output:
8463775
expectd: 10000000
result: 9063776

2.Expected output:
6267912
expectd: 10000000
result: 10000000
*/
94,8 Bot
```

Result

```
os-practice1-template-2022 [실행 중] - Oracle VM VirtualBox
파일  머신  보기  입력  장치  도움말

pthread_t current_thread = pthread_self();
return pthread_setaffinity_np(current_thread, sizeof(cpu_set_t), &cpuset);
}

/*
1.Expected output:
8463775
expectd: 10000000
result: 9063776

2.Expected output:
6267912
expectd: 10000000
result: 10000000
*/

os-practice: ~/.../quiz/03
+ ./main
5699999
expectd: 10000000
result: 10000000

os-practice: ~/.../quiz/03
+ ./main
9999999
expectd: 10000000
result: 10000000

os-practice: ~/.../quiz/03
+ ./main_
```