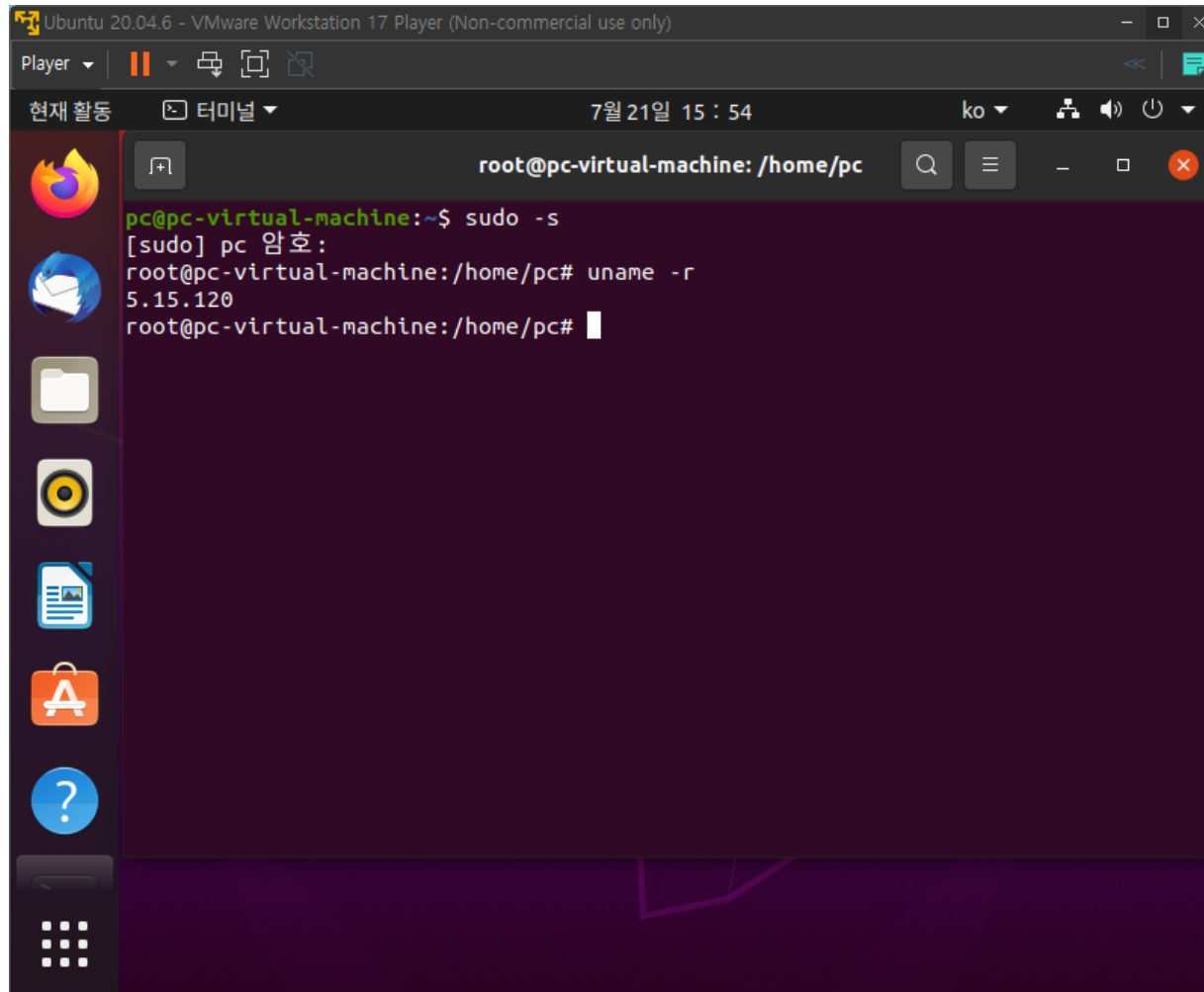


과제2 참고자료

Hello World! 시스템콜 추가

커널 환경 확인



Ubuntu 20.04.6 - VMware Workstation 17 Player (Non-commercial use only)

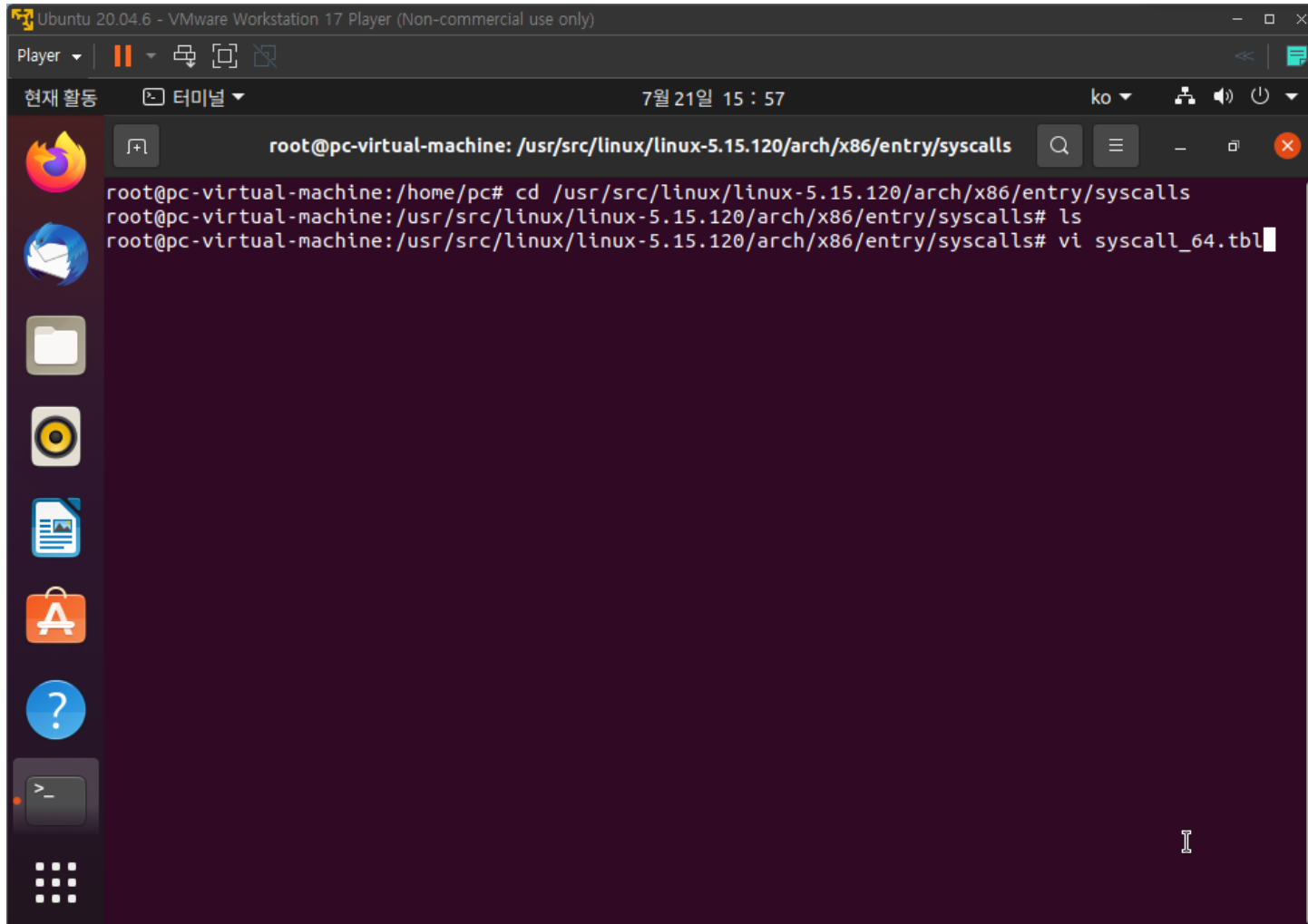
Player | 7월 21일 15 : 54 | ko

현재 활동 | 터미널

root@pc-virtual-machine: /home/pc

```
pc@pc-virtual-machine:~$ sudo -s
[sudo] pc 암호:
root@pc-virtual-machine:/home/pc# uname -r
5.15.120
root@pc-virtual-machine:/home/pc#
```

1. 시스템콜 추가 - 시스템콜 테이블 등록



```
root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls
root@pc-virtual-machine:/home/pc# cd /usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls
root@pc-virtual-machine:/usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls# ls
root@pc-virtual-machine:/usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls# vi syscall_64.tbl
```

/usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls

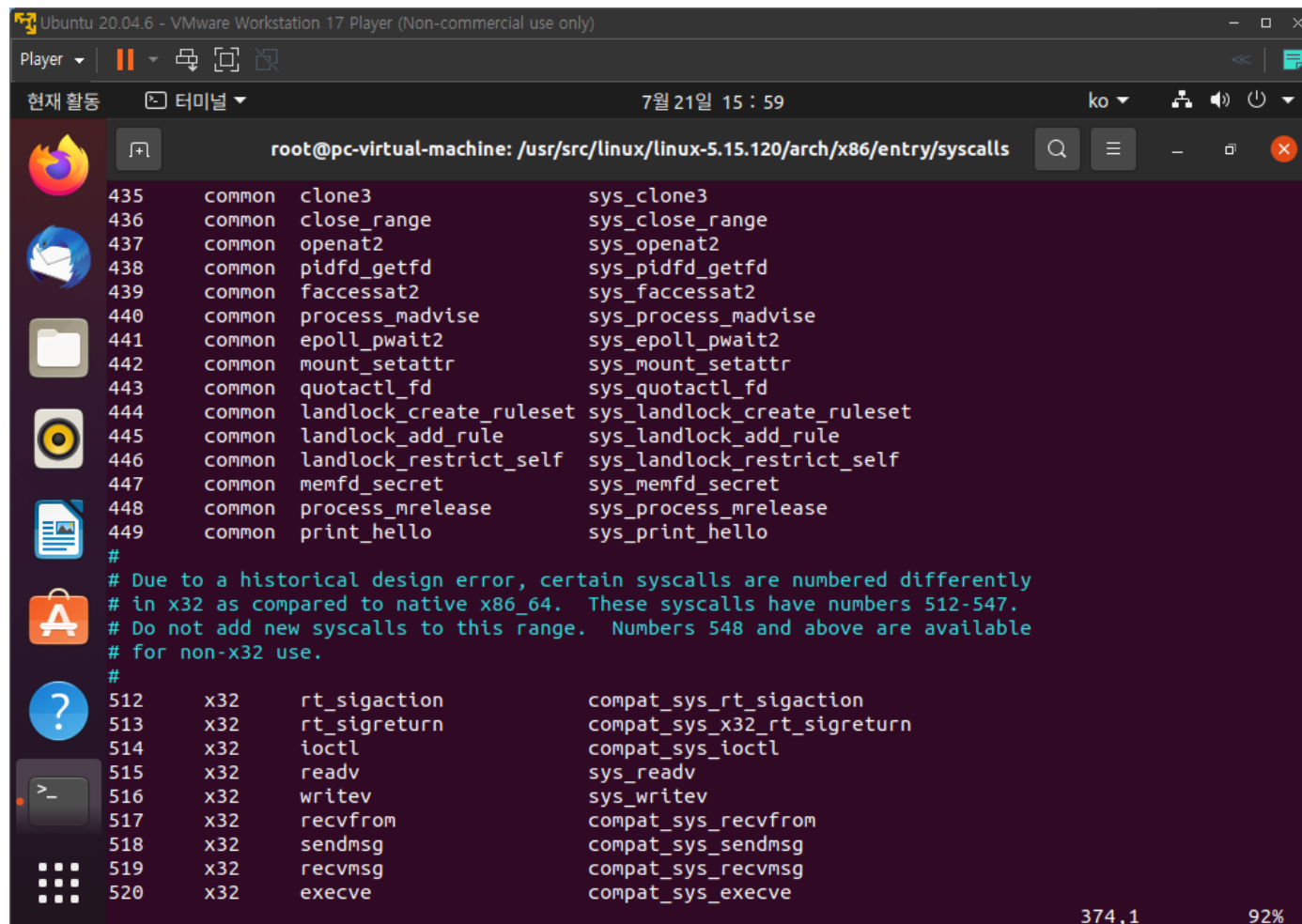
디렉토리로 이동 후

운영체제 종류 (64bit 혹은 32bit)에 맞는 syscall_xx.tbl

파일 편집

* 본 참고자료의 경우 64bit로 진행함

1. 시스템콜 추가 - 시스템콜 테이블 등록



```
root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls

435 common clone3 sys_clone3
436 common close_range sys_close_range
437 common openat2 sys_openat2
438 common pidfd_getfd sys_pidfd_getfd
439 common faccessat2 sys_faccessat2
440 common process_madvise sys_process_madvise
441 common epoll_pwait2 sys_epoll_pwait2
442 common mount_setattr sys_mount_setattr
443 common quotactl_fd sys_quotactl_fd
444 common landlock_create_ruleset sys_landlock_create_ruleset
445 common landlock_add_rule sys_landlock_add_rule
446 common landlock_restrict_self sys_landlock_restrict_self
447 common memfd_secret sys_memfd_secret
448 common process_mrelease sys_process_mrelease
449 common print_hello sys_print_hello
#
# Due to a historical design error, certain syscalls are numbered differently
# in x32 as compared to native x86_64. These syscalls have numbers 512-547.
# Do not add new syscalls to this range. Numbers 548 and above are available
# for non-x32 use.
#
512 x32 rt_sigaction compat_sys_rt_sigaction
513 x32 rt_sigreturn compat_sys_x32_rt_sigreturn
514 x32 ioctl compat_sys_ioctl
515 x32 readv sys_readv
516 x32 writev 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
```

시스템콜 테이블은 <number> <abi> <name> <entry point>

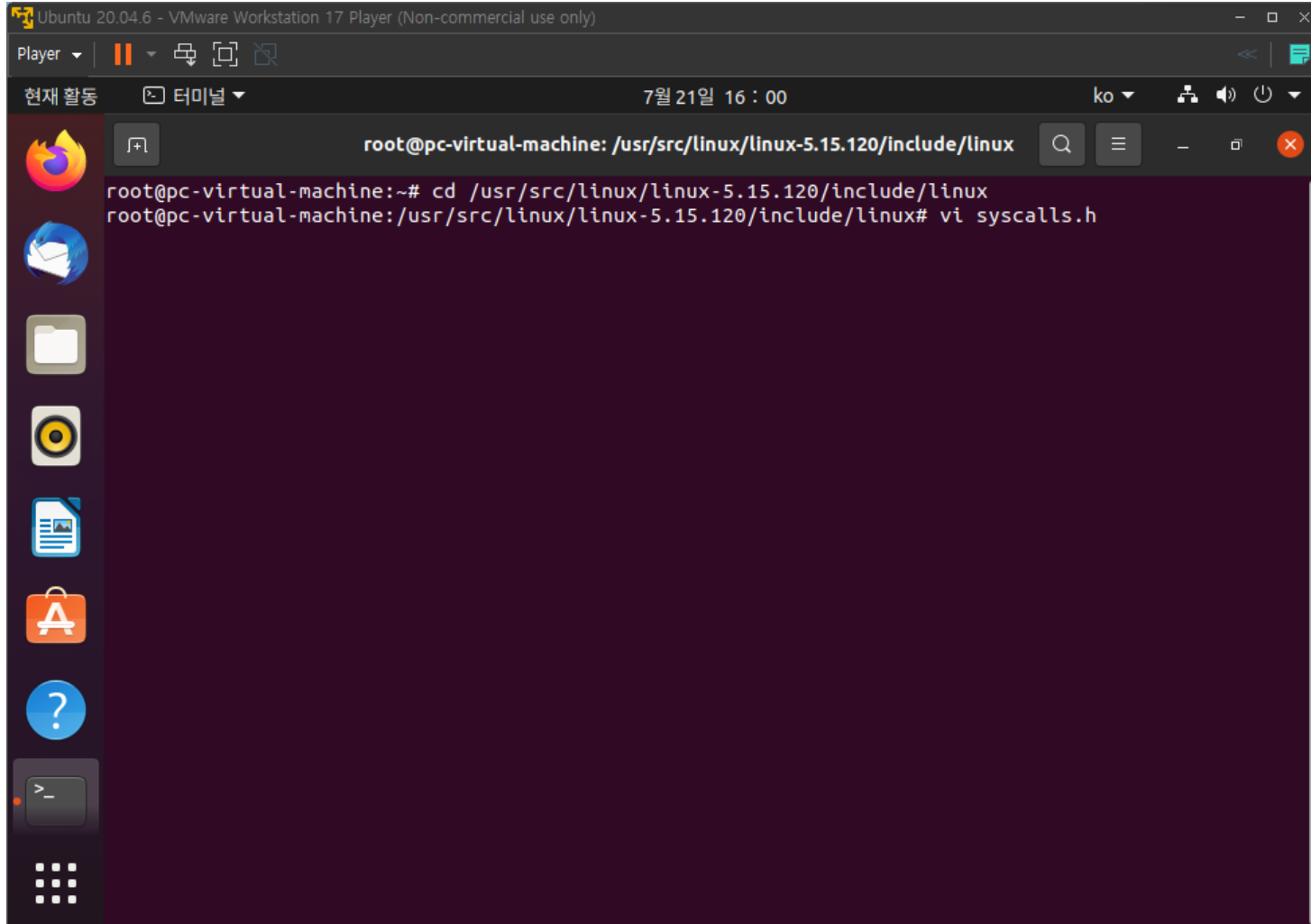
순으로 구성되어 있음

64bit 시스템 콜 테이블의 마지막에 이어지도록 작성함

도중에 사용 불가능한 구간이 존재하므로 주석을 잘 읽어보고 작성

추가한 시스템 콜의 번호는 기억해야 함

1. 시스템콜 추가 - 시스템콜 헤더 파일에 등록

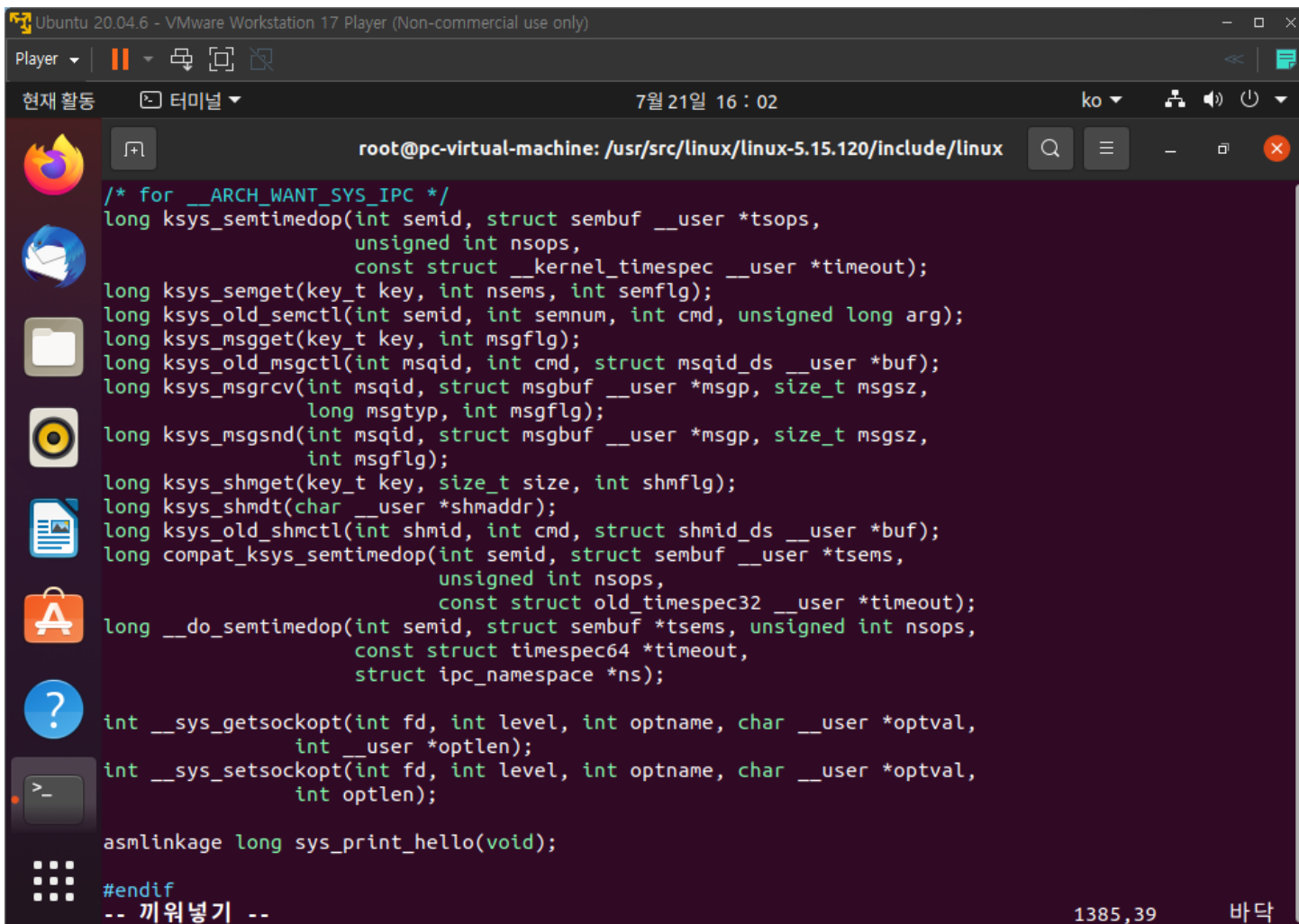


The screenshot shows a terminal window titled "Ubuntu 20.04.6 - VMware Workstation 17 Player (Non-commercial use only)". The terminal is running as root on a virtual machine named "pc-virtual-machine". The current directory is `/usr/src/linux/linux-5.15.120/include/linux`. The user has executed the command `cd /usr/src/linux/linux-5.15.120/include/linux` and then `vi syscalls.h` to edit the file. The terminal interface includes a top bar with "현재 활동" (Current Activity) and "터미널" (Terminal) tabs, a date and time display of "7월 21일 16:00", and a language dropdown set to "ko". The left sidebar shows various application icons, and the top of the terminal window displays the current path and search, menu, and window control icons.

```
root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/include/linux
root@pc-virtual-machine:~# cd /usr/src/linux/linux-5.15.120/include/linux
root@pc-virtual-machine:/usr/src/linux/linux-5.15.120/include/linux# vi syscalls.h
```

/usr/src/linux/linux-5.15.120/include/linux
디렉토리로 이동 후 syscalls.h 편집

1. 시스템콜 추가 - 시스템콜 헤더 파일에 등록



The screenshot shows a terminal window titled "root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/include/linux". The terminal displays a list of system call prototypes for IPC-related functions. The functions listed include `ksys_sem`, `ksys_msg`, `ksys_shm`, and `__do_semtimedop` functions, as well as `__sys_getsockopt` and `__sys_setsockopt`. At the bottom, there is a line `asmlinkage long sys_print_hello(void);` and a `#endif` line. The terminal also shows a status bar at the bottom with "1385,39" and "바닥".

```
/* for __ARCH_WANT_SYS_IPC */
long ksys_semtimedop(int semid, struct sembuf __user *tsops,
                    unsigned int nsops,
                    const struct __kernel_timespec __user *timeout);
long ksys_semget(key_t key, int nsems, int semflg);
long ksys_old_semctl(int semid, int semnum, int cmd, unsigned long arg);
long ksys_msgget(key_t key, int msgflg);
long ksys_old_msgctl(int msqid, int cmd, struct msqid_ds __user *buf);
long ksys_msgrcv(int msqid, struct msgbuf __user *msgp, size_t msgsz,
                long msgtyp, int msgflg);
long ksys_msgsnd(int msqid, struct msgbuf __user *msgp, size_t msgsz,
                int msgflg);
long ksys_shmget(key_t key, size_t size, int shmflg);
long ksys_shmdt(char __user *shmaddr);
long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
long compat_ksys_semtimedop(int semid, struct sembuf __user *tsems,
                            unsigned int nsops,
                            const struct old_timespec32 __user *timeout);
long __do_semtimedop(int semid, struct sembuf *tsems, unsigned int nsops,
                    const struct timespec64 *timeout,
                    struct ipc_namespace *ns);

int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
                    int __user *optlen);
int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
                    int optlen);

asmlinkage long sys_print_hello(void);

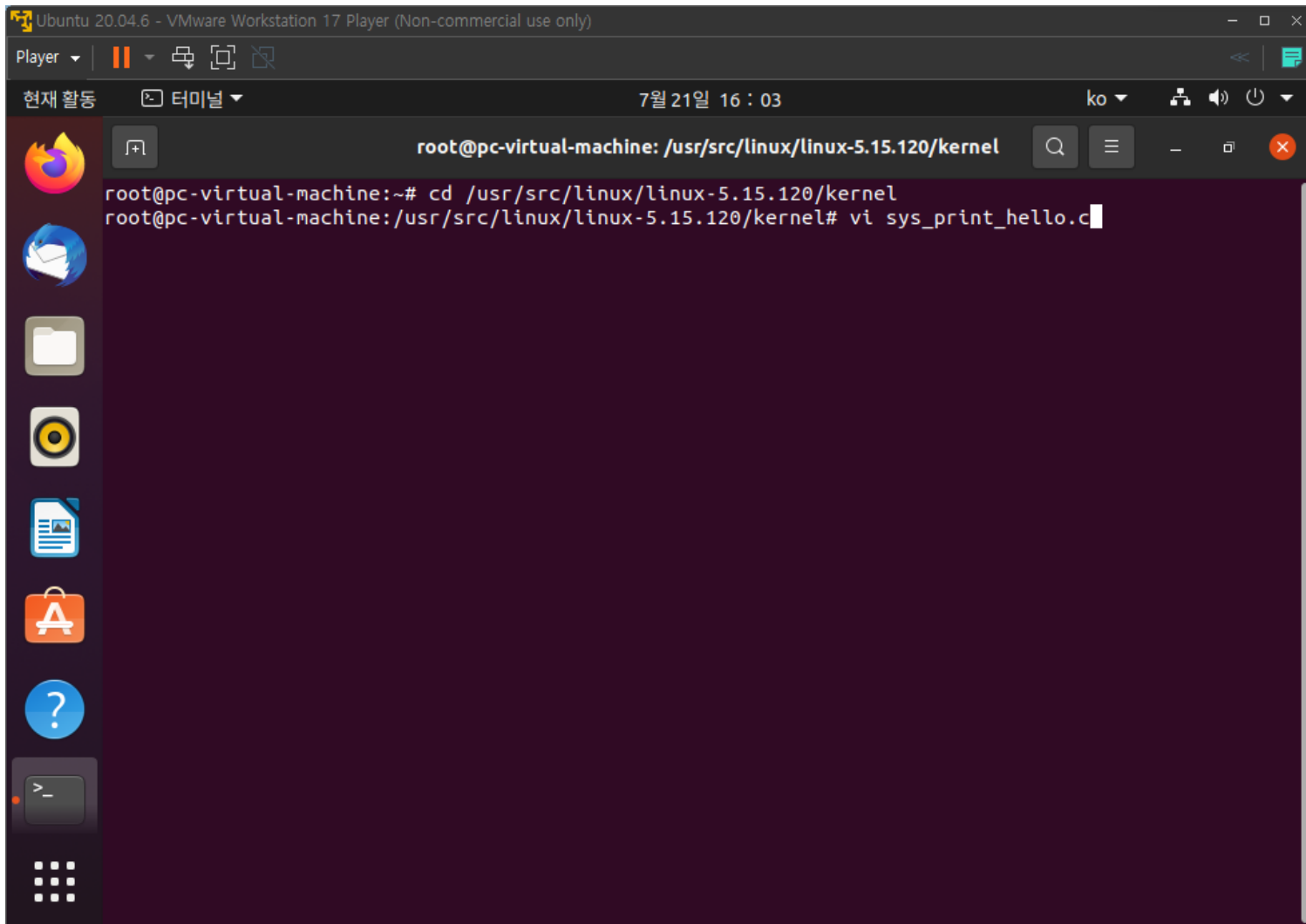
#endif
-- 끼워넣기 --
```

함수의 프로토타입을 정의함

asmlinkage를 앞에 붙힘으로서 어셈블리 코드에서도

C 함수 호출이 가능해짐

1. 시스템콜 추가 - 시스템콜 함수 구현

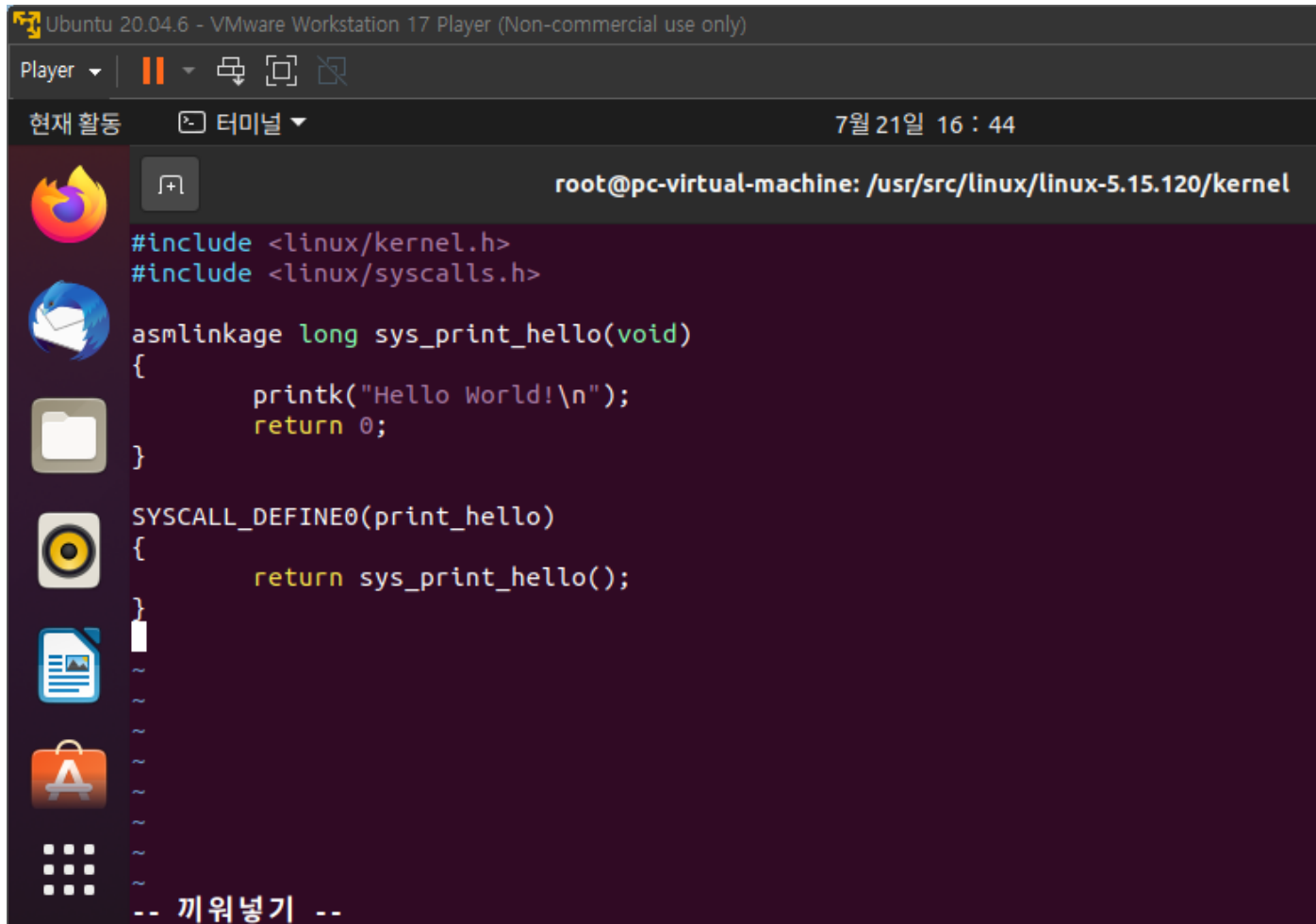


The screenshot shows a terminal window titled "Ubuntu 20.04.6 - VMware Workstation 17 Player (Non-commercial use only)". The terminal prompt is "root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/kernel". The user has entered the command "cd /usr/src/linux/linux-5.15.120/kernel" and the prompt has changed to "root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/kernel#". The user has then entered the command "vi sys_print_hello.c" and the prompt is now "root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/kernel# vi sys_print_hello.c". The terminal window is running on a system with the locale "ko".

```
root@pc-virtual-machine:~# cd /usr/src/linux/linux-5.15.120/kernel
root@pc-virtual-machine:/usr/src/linux/linux-5.15.120/kernel# vi sys_print_hello.c
```

/usr/src/linux/linux-5.15.120/kernel 디렉토리 이동 후
추가할 시스템콜 구현 파일을 편집

1. 시스템콜 추가 - 시스템콜 함수 구현



The screenshot shows a terminal window titled "Ubuntu 20.04.6 - VMware Workstation 17 Player (Non-commercial use only)". The terminal prompt is "root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/kernel". The code being edited is as follows:

```
#include <linux/kernel.h>
#include <linux/syscalls.h>

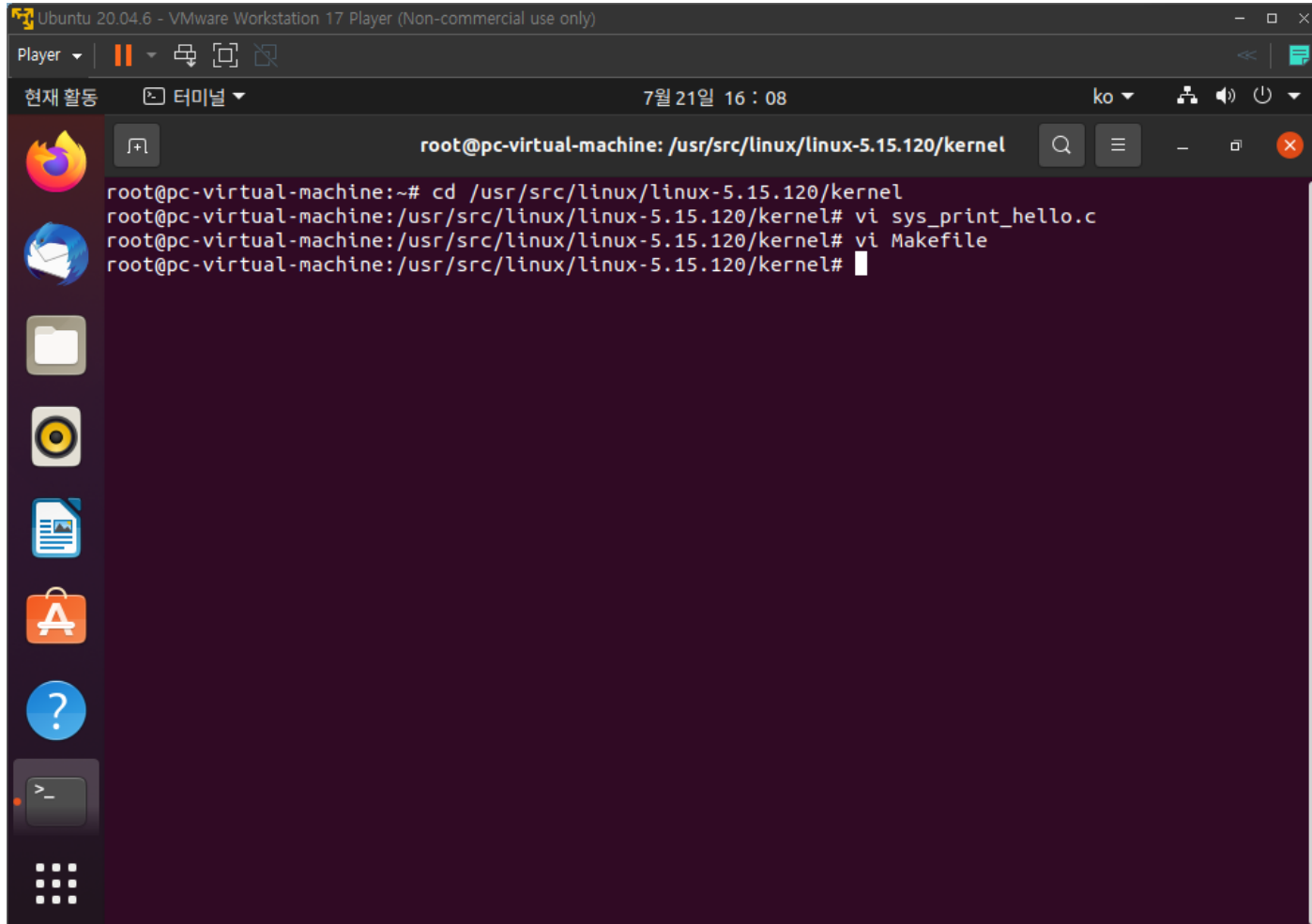
asmlinkage long sys_print_hello(void)
{
    printk("Hello World!\n");
    return 0;
}

SYSCALL_DEFINE0(print_hello)
{
    return sys_print_hello();
}
```

At the bottom of the terminal, there is a comment: "-- 끼워넣기 --".

파일명은 시스템콜 이름과 달라도 되지만 파일 내부의 함수는 sys_시스템콜 이름으로 작성해야 함
printk 명령어를 통해 커널 공간에서 메시지 출력이 되도록 작성함

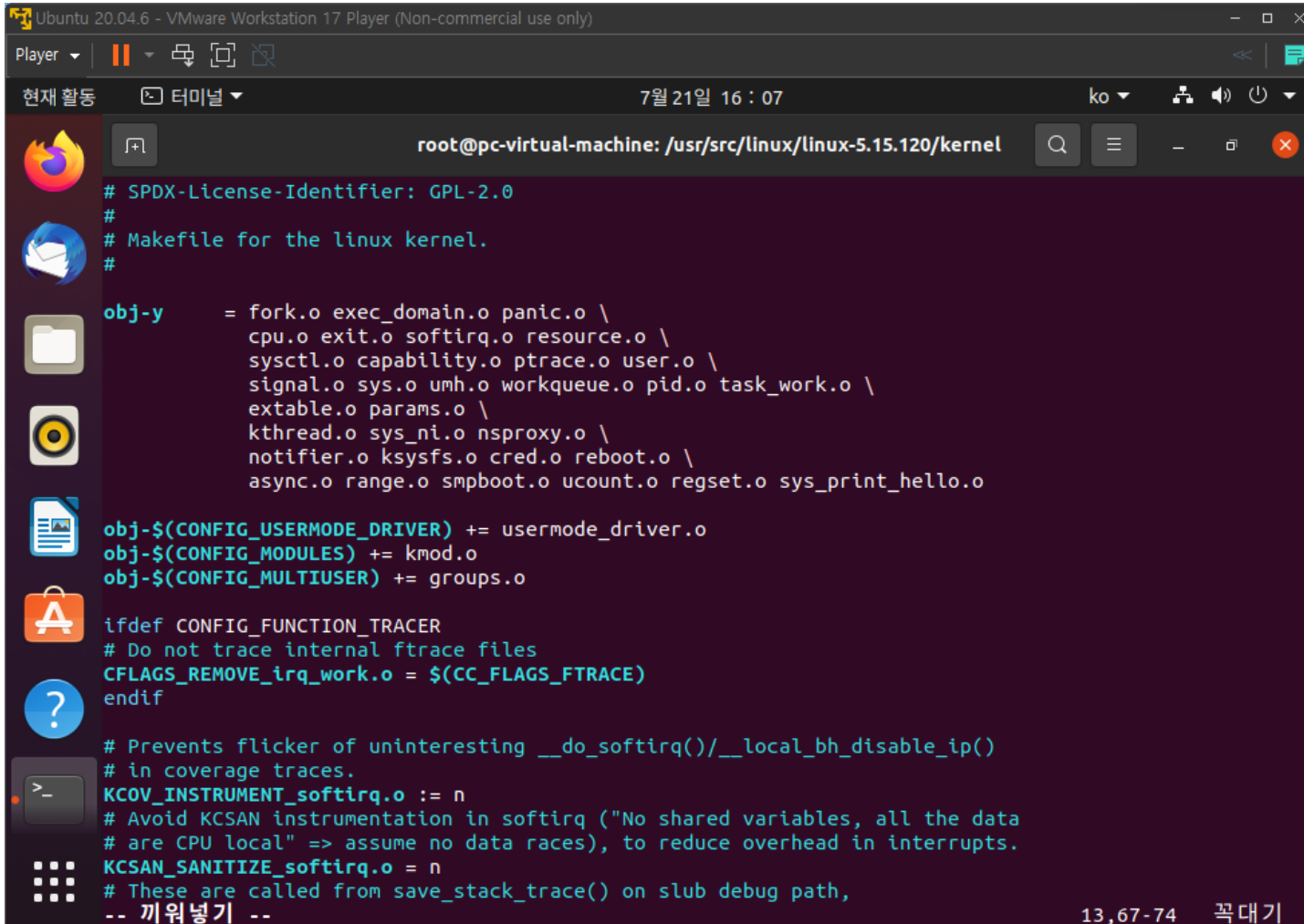
1. 시스템콜 추가 - Makefile에 등록



```
root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/kernel
root@pc-virtual-machine:~# cd /usr/src/linux/linux-5.15.120/kernel
root@pc-virtual-machine:/usr/src/linux/linux-5.15.120/kernel# vi sys_print_hello.c
root@pc-virtual-machine:/usr/src/linux/linux-5.15.120/kernel# vi Makefile
root@pc-virtual-machine:/usr/src/linux/linux-5.15.120/kernel#
```

이전 단계와 동일한 디렉토리 내에서 Makefile 편집

1. 시스템콜 추가 - Makefile에 등록



```
root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/kernel
# SPDX-License-Identifier: GPL-2.0
#
# Makefile for the linux kernel.
#
obj-y      = fork.o exec_domain.o panic.o \
             cpu.o exit.o softirq.o resource.o \
             sysctl.o capability.o ptrace.o user.o \
             signal.o sys.o umh.o workqueue.o pid.o task_work.o \
             extable.o params.o \
             kthread.o sys_ni.o nsproxy.o \
             notifier.o ksyzfs.o cred.o reboot.o \
             async.o range.o smpboot.o ucount.o regset.o sys_print_hello.o

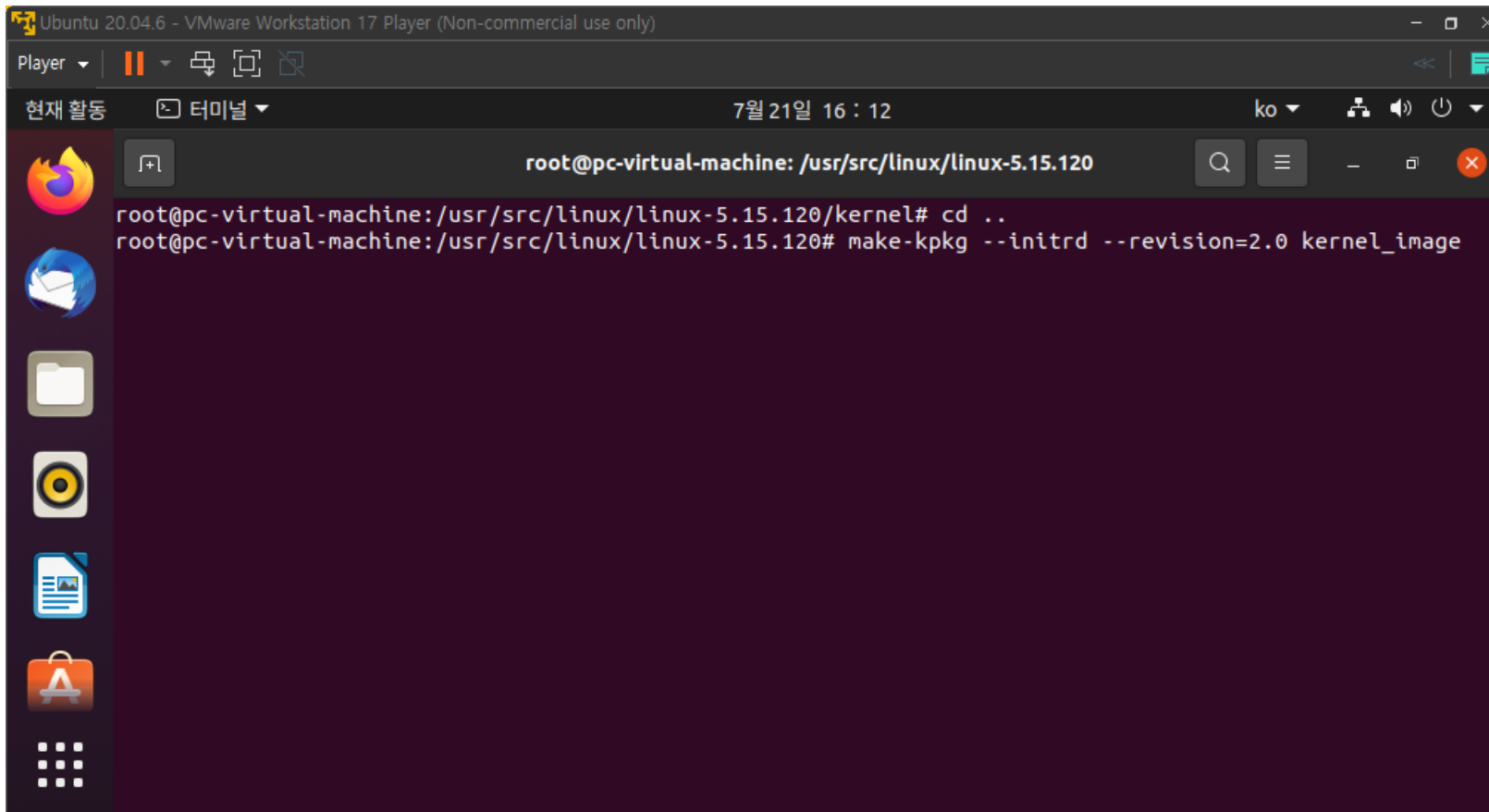
obj-$(CONFIG_USERMODE_DRIVER) += usermode_driver.o
obj-$(CONFIG_MODULES) += kmod.o
obj-$(CONFIG_MULTIUSER) += groups.o

ifdef CONFIG_FUNCTION_TRACER
# Do not trace internal ftrace files
CFLAGS_REMOVE_irq_work.o = $(CC_FLAGS_FTRACE)
endif

# Prevents flicker of uninteresting __do_softirq()/__local_bh_disable_ip()
# in coverage traces.
KCOV_INSTRUMENT_softirq.o := n
# Avoid KCSAN instrumentation in softirq ("No shared variables, all the data
# are CPU local" => assume no data races), to reduce overhead in interrupts.
KCSAN_SANITIZE_softirq.o = n
# These are called from save_stack_trace() on slub debug path,
-- 끼워넣기 --
```

추가한 시스템콜이 다른 시스템콜과 함께 컴파일될 수 있도록
Makefile 편집

1. 시스템콜 추가 - Makefile에 등록

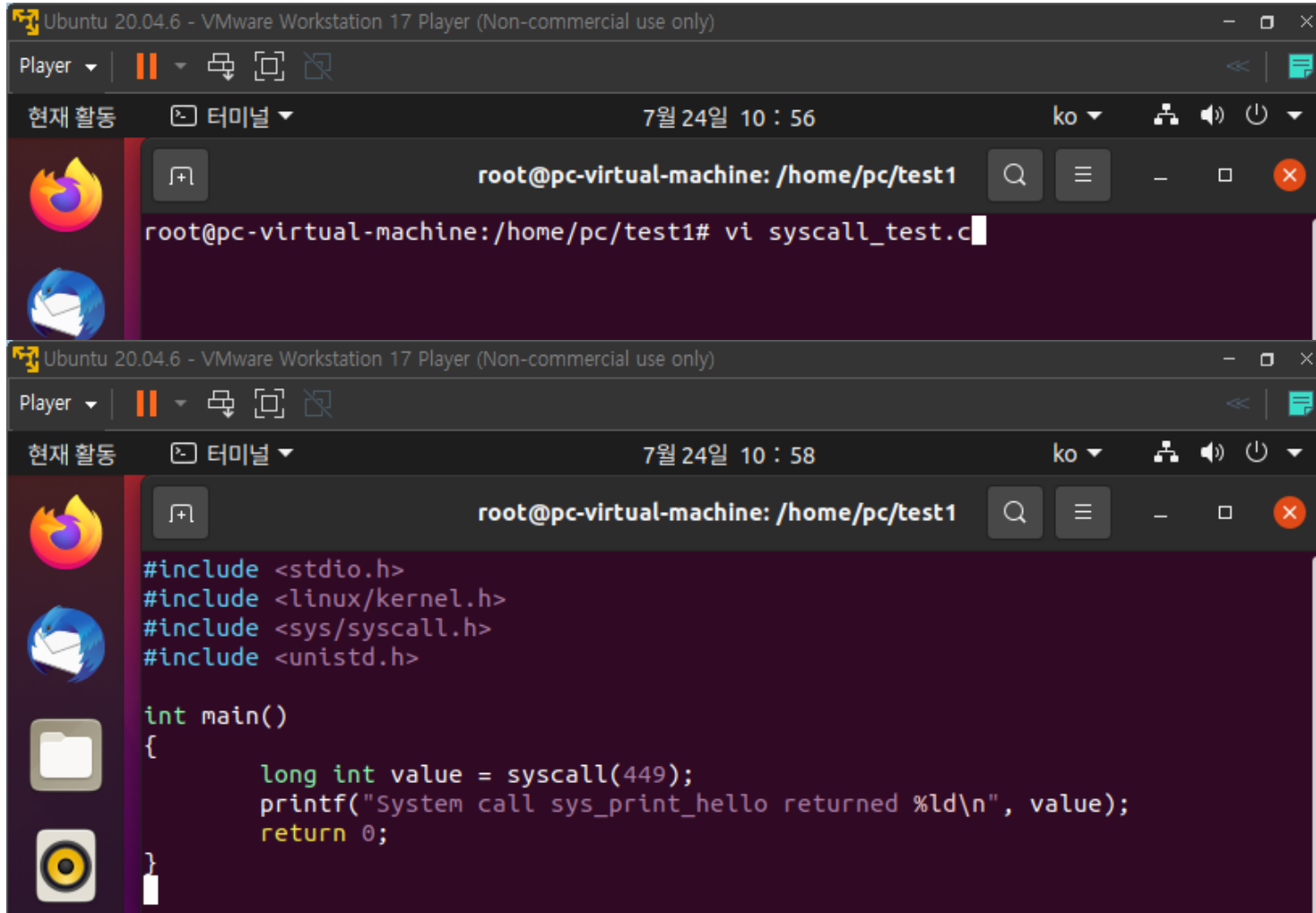


```
root@pc-virtual-machine: /usr/src/linux/linux-5.15.120/kernel# cd ..
root@pc-virtual-machine: /usr/src/linux/linux-5.15.120# make-kpkg --initrd --revision=2.0 kernel_image
```

커널 소스 디렉토리로 이동하여 새로 컴파일 후
재부팅 실시함

이때 revision의 값은 기존의 1.0과 구분을 할 수
있도록 다른 값을 입력하여 컴파일 실시 (정수
값으로 설정)

2. 추가된 시스템콜 호출



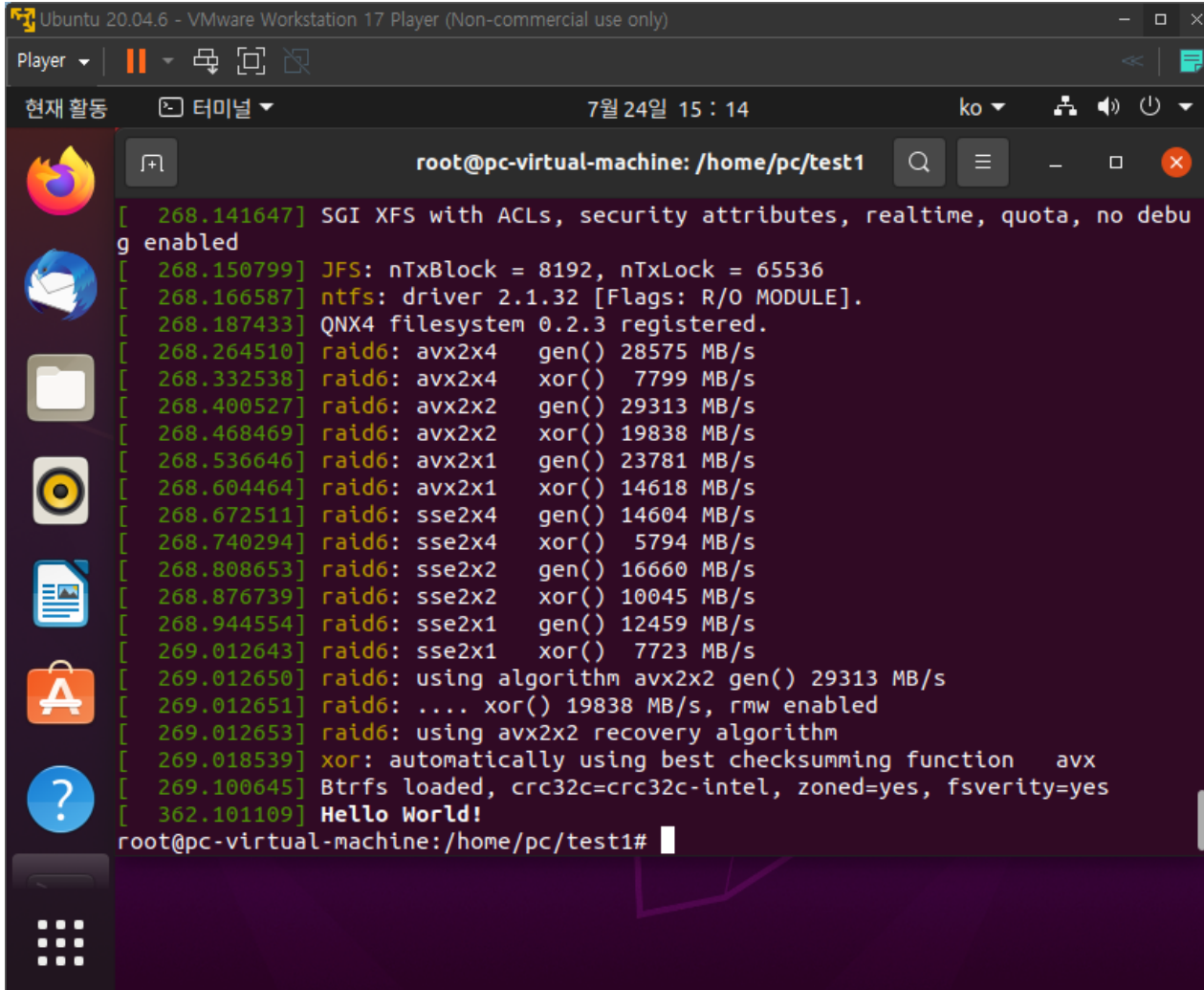
```
root@pc-virtual-machine: /home/pc/test1
root@pc-virtual-machine:/home/pc/test1# vi syscall_test.c

#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>

int main()
{
    long int value = syscall(449);
    printf("System call sys_print_hello returned %ld\n", value);
    return 0;
}
```

재부팅 후 추가한 시스템콜을 호출하는 함수를 생성함
앞서 시스템콜을 테이블에 등록할 때 생성한 번호를
사용하여 호출

2. 추가된 시스템콜 호출



```
Ubuntu 20.04.6 - VMware Workstation 17 Player (Non-commercial use only)
Player
현재 활동 7월 24일 15 : 14 ko
root@pc-virtual-machine: /home/pc/test1
[ 268.141647] SGI XFS with ACLs, security attributes, realtime, quota, no debug enabled
[ 268.150799] JFS: nTxBlock = 8192, nTxLock = 65536
[ 268.166587] ntfs: driver 2.1.32 [Flags: R/O MODULE].
[ 268.187433] QNX4 filesystem 0.2.3 registered.
[ 268.264510] raid6: avx2x4 gen() 28575 MB/s
[ 268.332538] raid6: avx2x4 xor() 7799 MB/s
[ 268.400527] raid6: avx2x2 gen() 29313 MB/s
[ 268.468469] raid6: avx2x2 xor() 19838 MB/s
[ 268.536646] raid6: avx2x1 gen() 23781 MB/s
[ 268.604464] raid6: avx2x1 xor() 14618 MB/s
[ 268.672511] raid6: sse2x4 gen() 14604 MB/s
[ 268.740294] raid6: sse2x4 xor() 5794 MB/s
[ 268.808653] raid6: sse2x2 gen() 16660 MB/s
[ 268.876739] raid6: sse2x2 xor() 10045 MB/s
[ 268.944554] raid6: sse2x1 gen() 12459 MB/s
[ 269.012643] raid6: sse2x1 xor() 7723 MB/s
[ 269.012650] raid6: using algorithm avx2x2 gen() 29313 MB/s
[ 269.012651] raid6: .... xor() 19838 MB/s, rmw enabled
[ 269.012653] raid6: using avx2x2 recovery algorithm
[ 269.018539] xor: automatically using best checksumming function avx
[ 269.100645] Btrfs loaded, crc32c=crc32c-intel, zoned=yes, fsverity=yes
[ 362.101109] Hello World!
root@pc-virtual-machine:/home/pc/test1#
```

테스트 프로그램 실행 후 dmesg 명령어를 통해 커널 로그를 출력하면
Hello World! 가 출력된 것을 확인할 수 있음