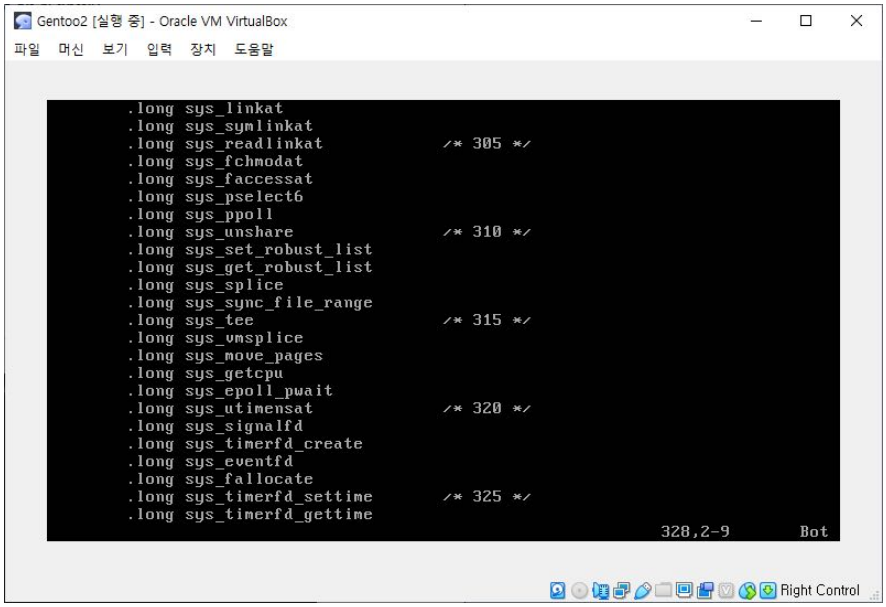
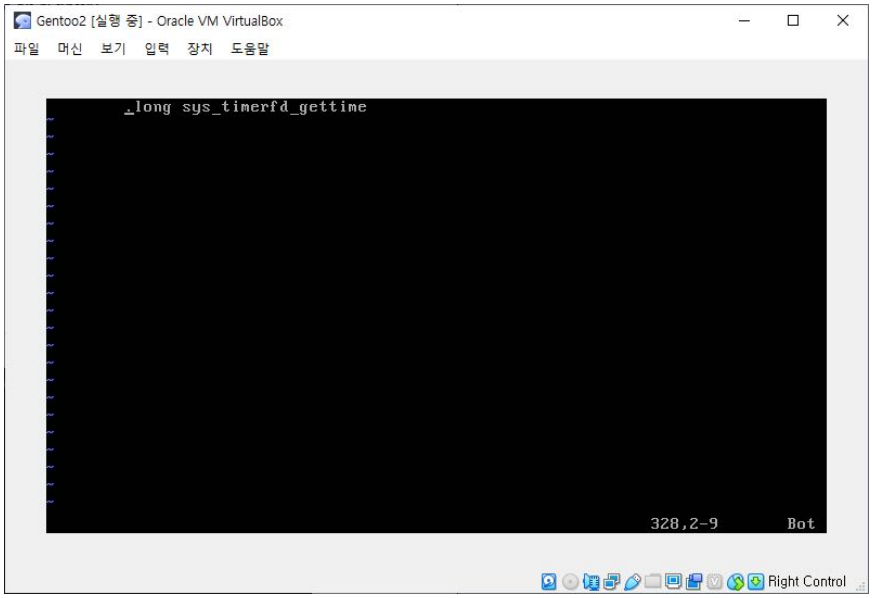
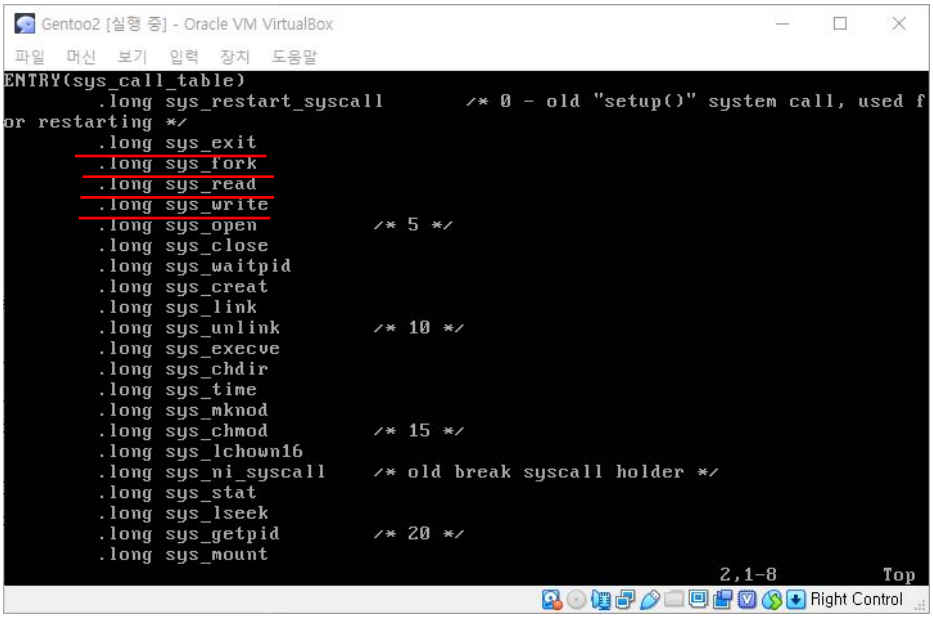
7) sys_call_table[] is in arch/x86/kernel/syscall_table_32.S. How many system calls does Linux 2.6 support? What are the system call numbers for exit, fork, execve, wait4, read, write, and mkdir? Find system call numbers for sys_ni_syscall, which is defined at kernel/sys_ni.c. What is the role of sys_ni_syscall?



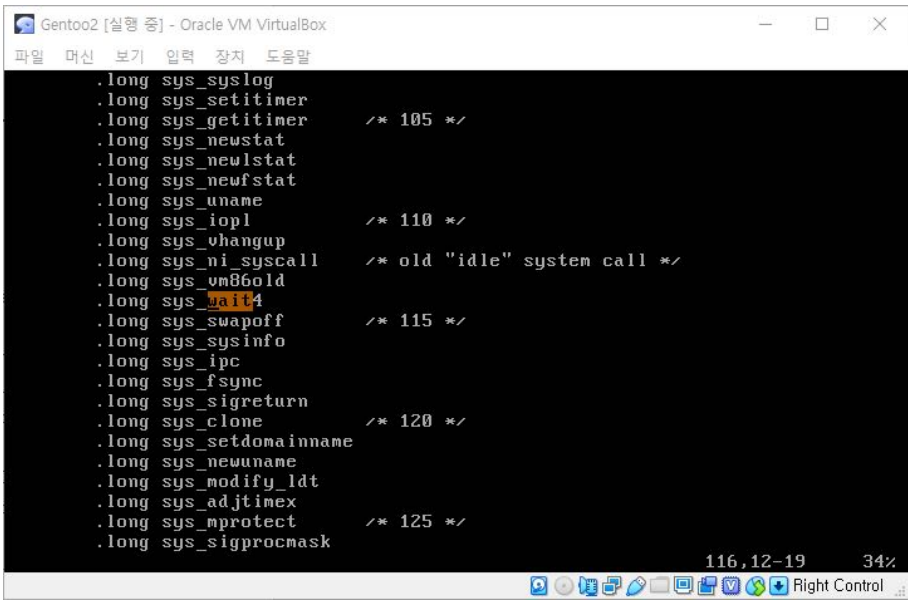
<System call table>



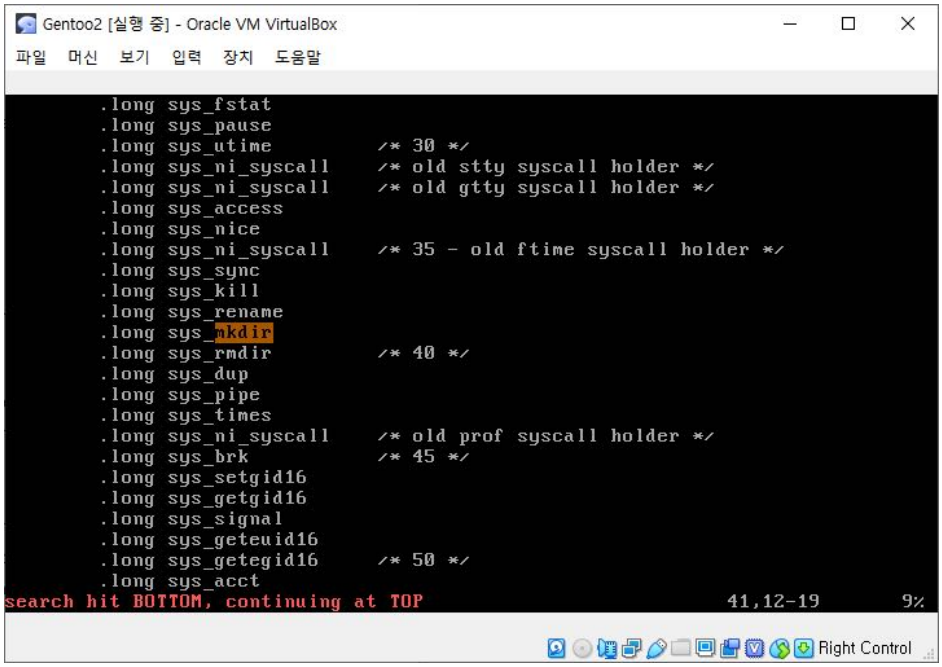
<sys_timerfd_gettime이 system call table의 마지막에 위치해있음. 즉 327번에 위치해있고, 이로써 리눅스에는 327개의 system call이 존재한다는 것을 알 수 있음>



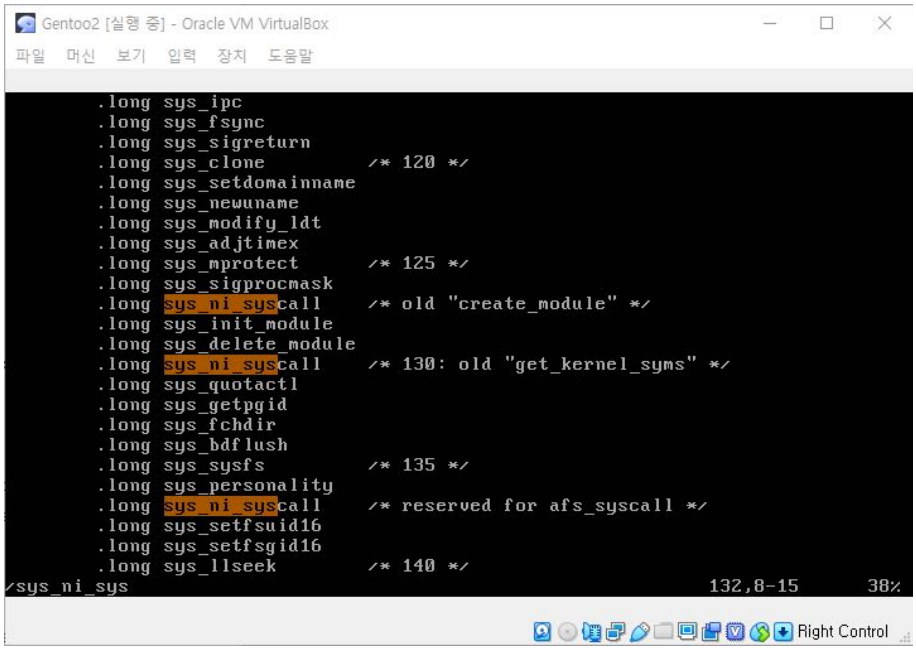
<exit(1), fork(2), read(3), write(4) 위치>



<wait4(114) 위치>



<mkdir(39) 위치>



<sys_ni_syscall에는 아무런 system call 함수가 들어있지 않음 즉, 빈칸임,
새로 선언한 system call 함수를 이 곳에다 넣으면 됨>

8) Change the kernel such that it prints "length 17 string found" for each printf(s) when the length of s is 17. Run a program that contains a printf() statement to see the effect. printf(s) calls write(1, s, strlen(s)) system call which in turn runs

mov eax, 4 ; eax<--4. 4 is system call number for "write"

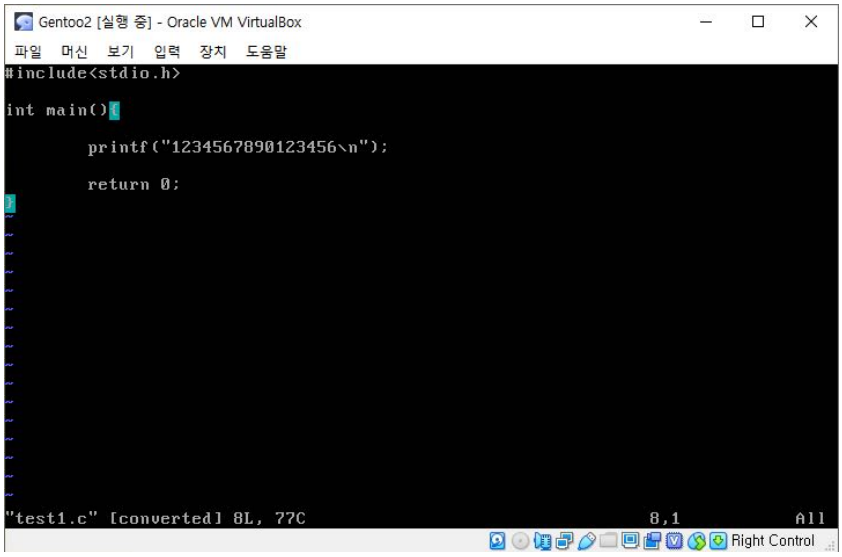
int 128

INT 128 will make the cpu stop running current process and jump to the location written in IDT[128]. IDT[128] contains the address of system_call (located in arch/x86/kernel/entry_32.S). Finally, system_call will execute

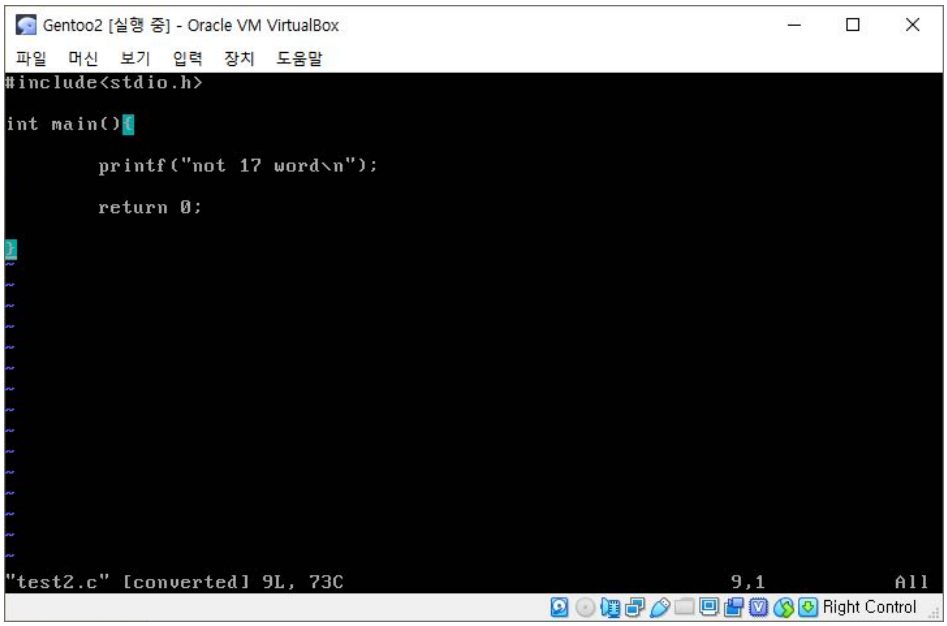
call *sys_call_table(,%eax,4)

which eventually calls sys_write() since eax=4 for write() system call (the target function location is sys_call_table+eax*4).

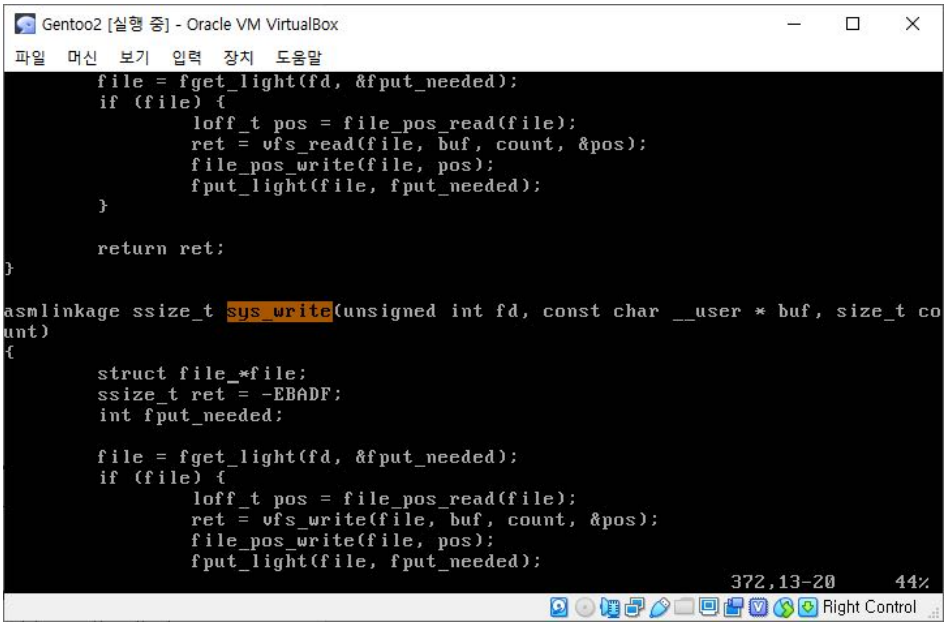
* Sometimes the compiler generate "sysenter" instead of "int 128". In this case the cpu jumps to ia32_sysenter_target (also in entry_32.S) instead of system_call.



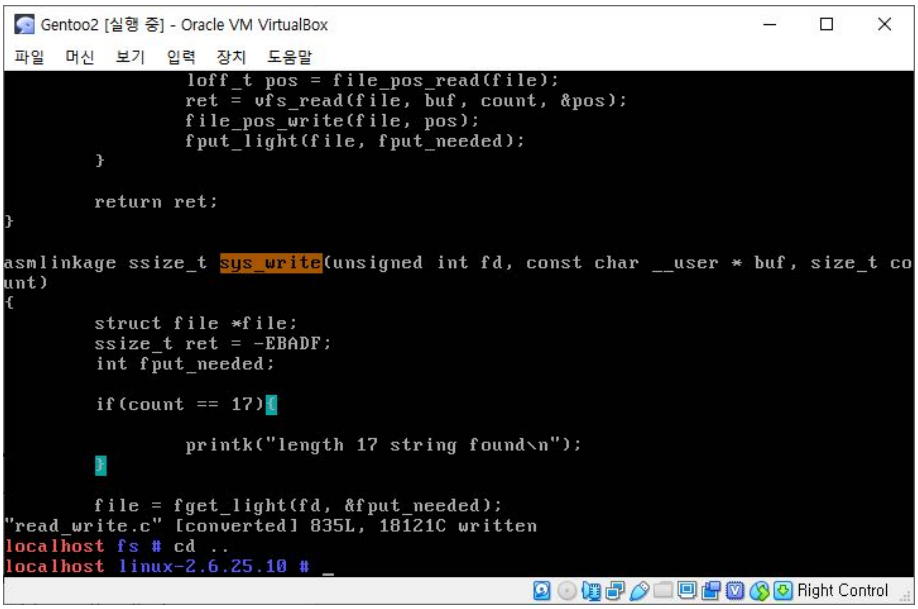
<test1.c 파일 생성>



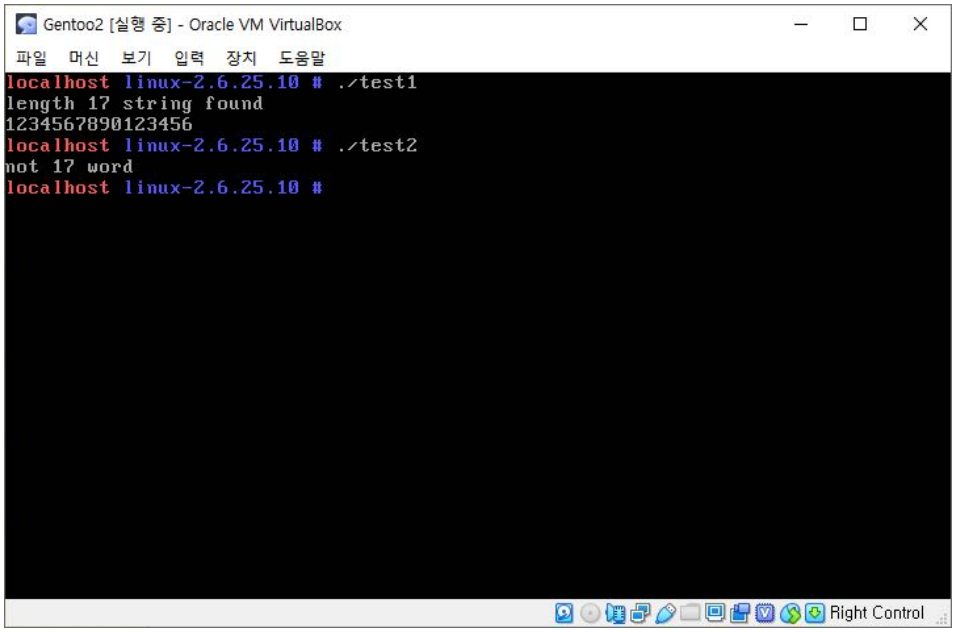
<test2.c 파일 생성>



<fs/read_write.c>에 선언되어 있는 sys_write()함수>



<sys_write()함수 변경>



<test1.c와 test2.c를 컴파일 한 결과>

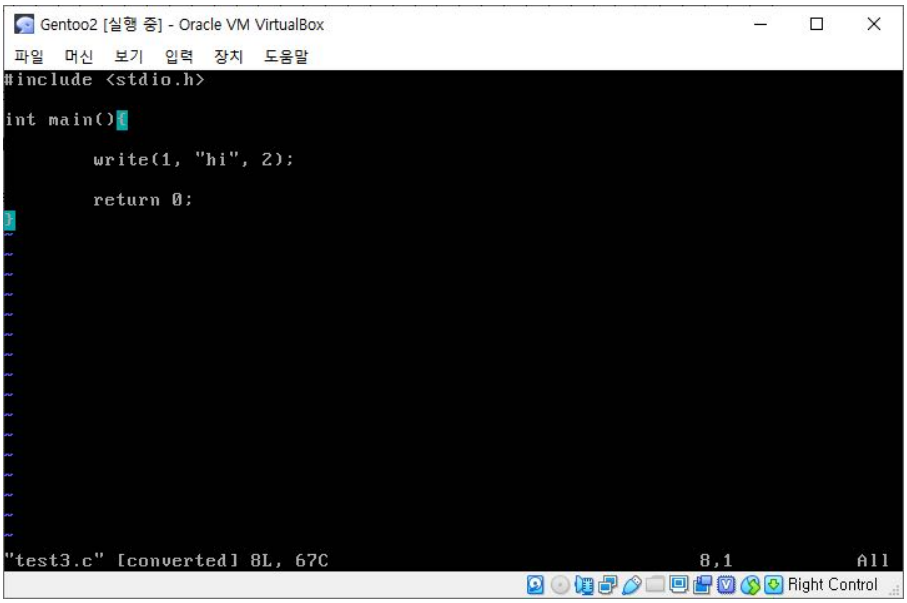
9) You can call a system call indirectly with “syscall()”.

write(1, “hi”, 2);

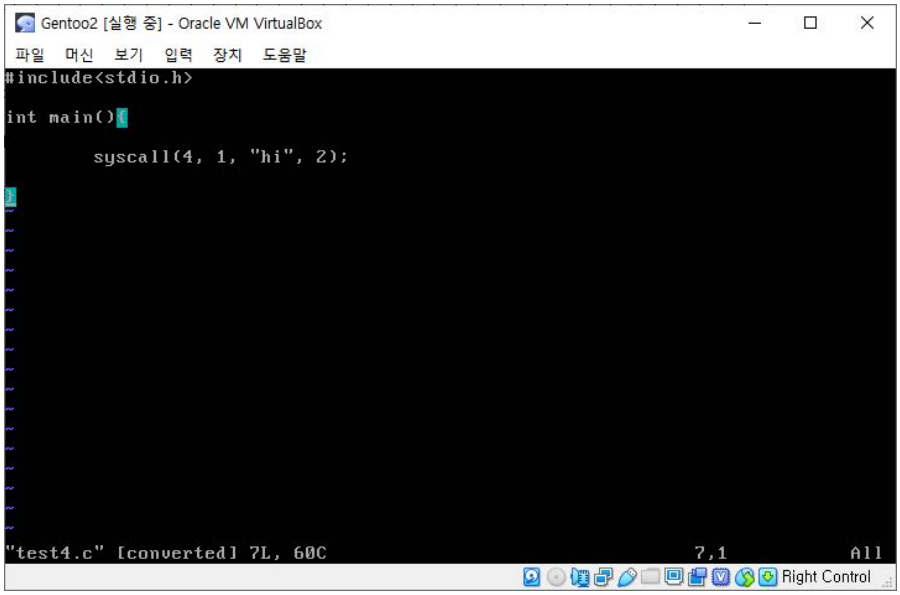
can be written as

syscall(4, 1, “hi”, 2); // 4 is the system call number for “write” system call

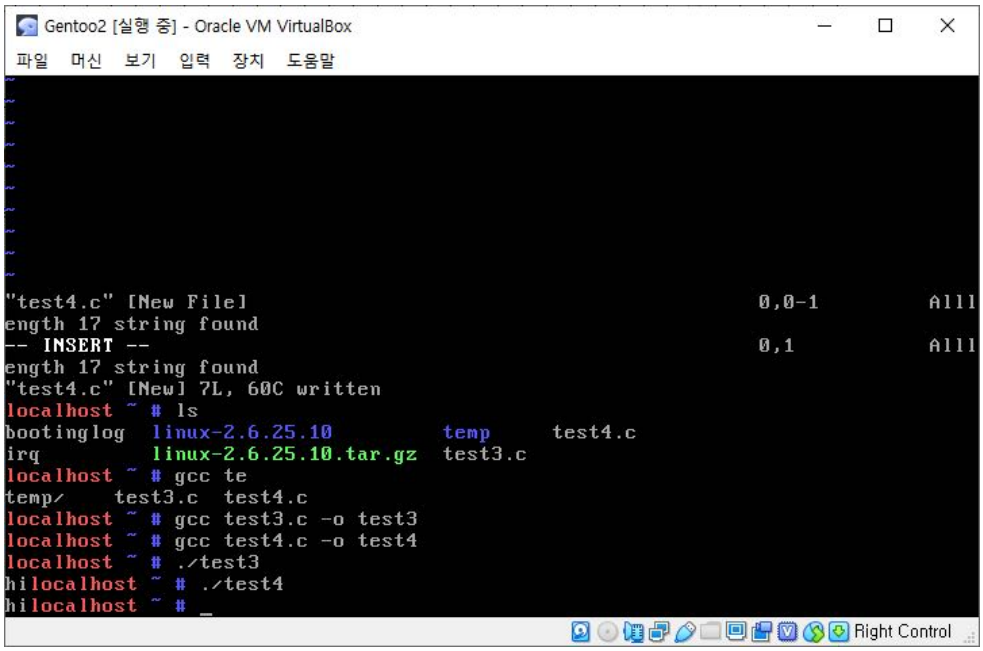
Write a program that prints “hello” in the screen using syscall.



<test3.c 파일>



<test4.c 파일>



<test3.c와 test4.c를 컴파일 한 결과>

10) Create a new system call, my_sys_call with system call number 17 (system call number 17 is one that is not being used currently). Define my_sys_call() just before sys_write() in read_write.c. Write a program that uses this system call:

```
void main(){
    syscall(17); // calls a system call with syscall number 17
}
```

When the above program runs, the kernel should display

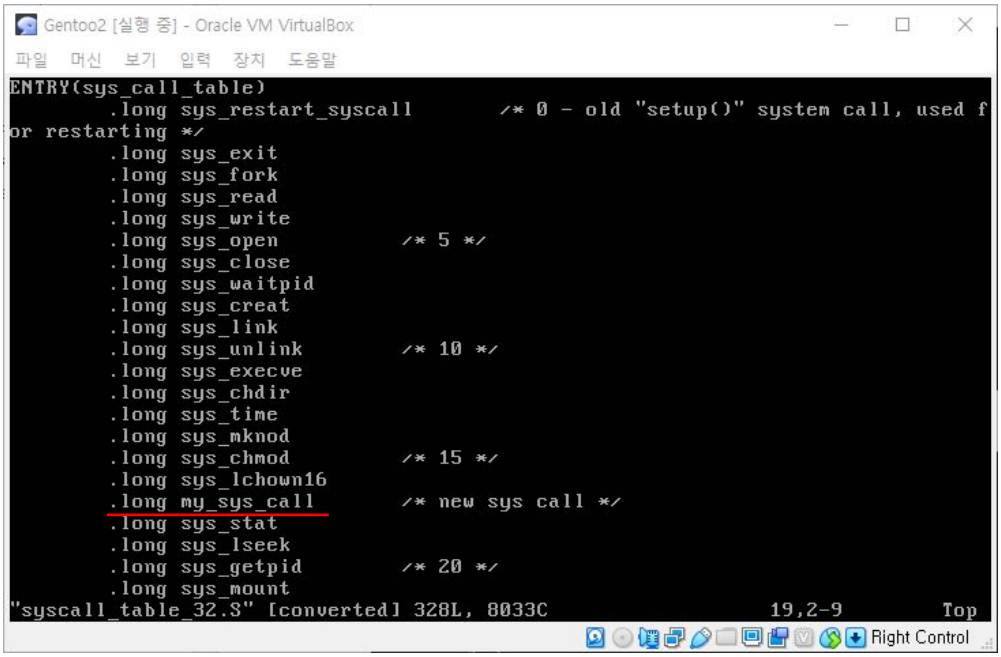
hello from my_sys_call

To define a new system call with syscall number x

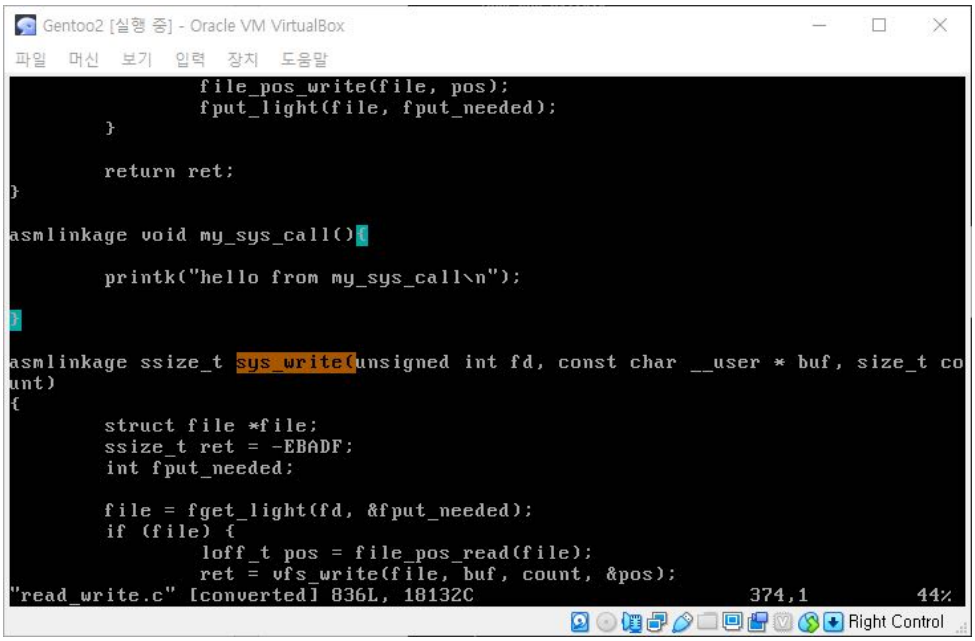
- insert the new system call name in arch/x86/kernel/syscall_table_32.S at index x
- define the function in appropriate file (such as "read_write.c")
- recompile and reboot

To use this system call in a user program

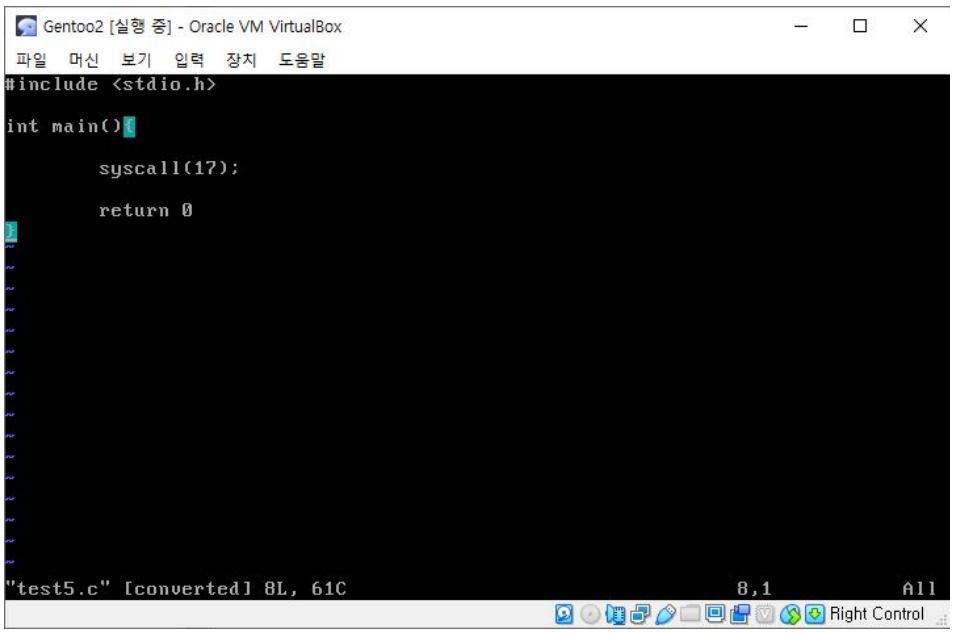
- void main(){
  syscall(x);
  }



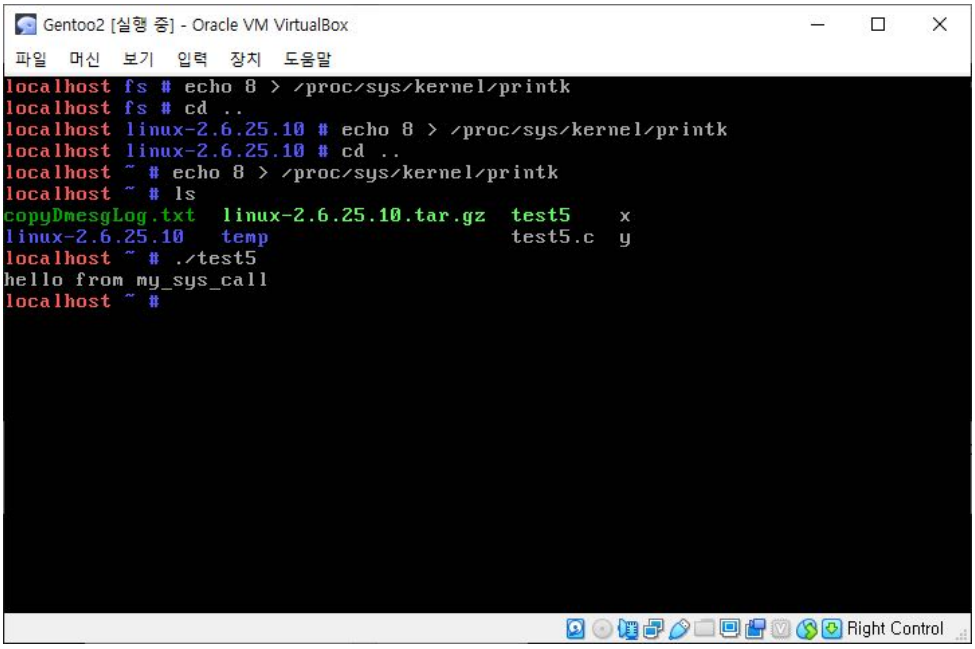
<system call table의 17번 위치에 'my_sys_call' 추가>



<read_write.c 내에 'my_sys_call()'함수 추가>

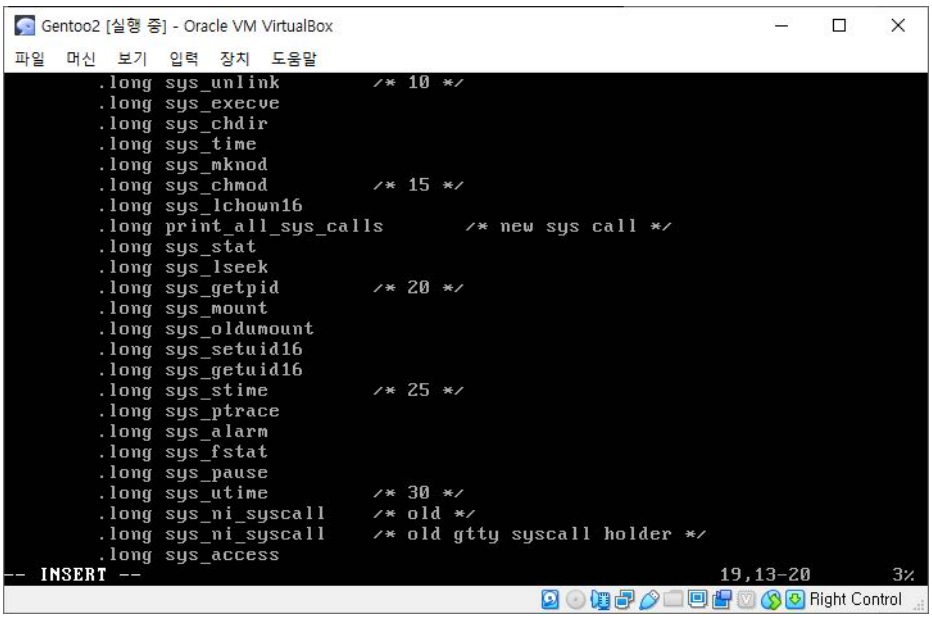


<test5.c 파일 생성>

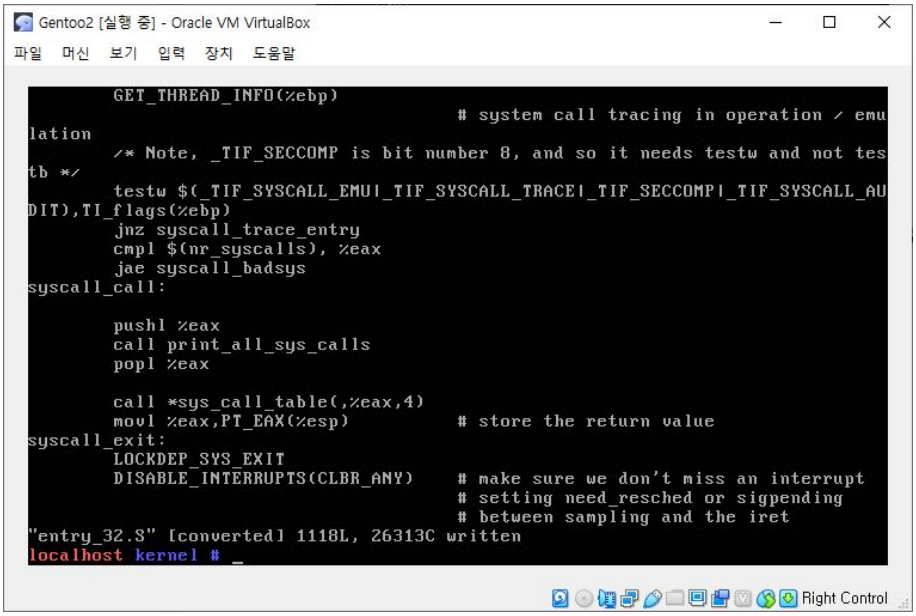


<test5.c 컴파일 한 결과>

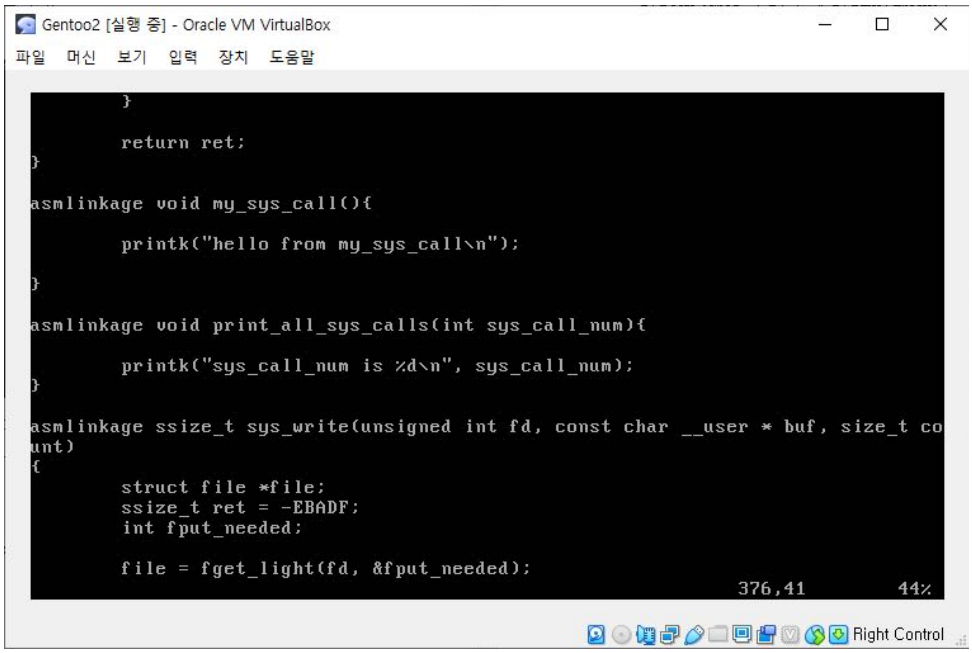
11) Modify the kernel such that it displays the system call number for all system calls. Run a simple program that displays "hello" in the screen and find out what system calls have been called.



<syscall_table의 17번에 'print_all_sys_calls'을 추가하였음>



<entry_32.S 파일 내 'syscall_call'부분을 수정함>



<read_write.c 파일 내 'print_all_sys_calls()'라는 system call 함수를 선언함
해당 함수는 특정 system call함수가 실행 될 때마다, 해당 system call number를
출력해주는 역할을 수행함>

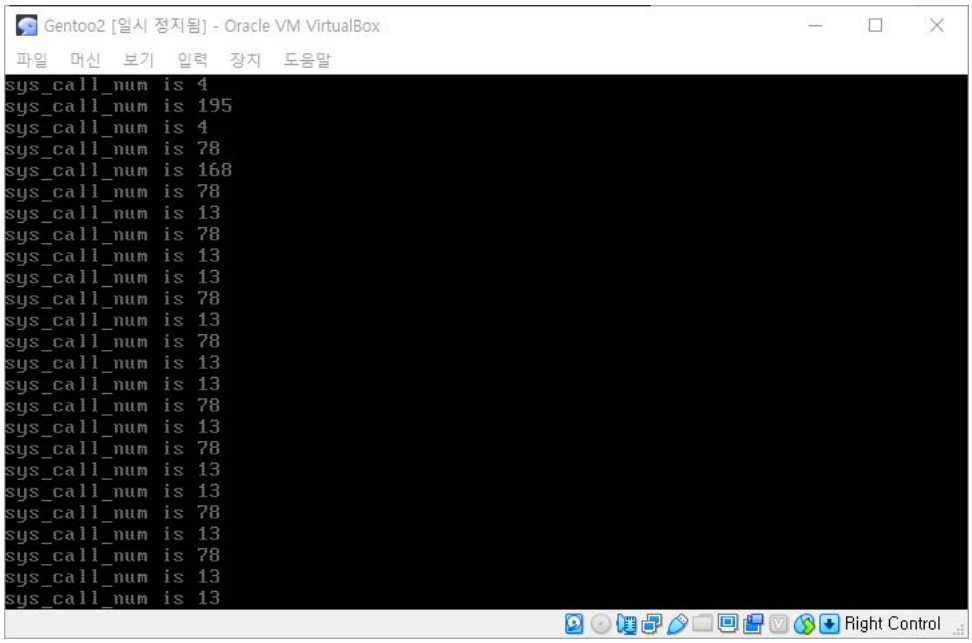
push, call, pop은 '어셈블리어'에 속하는 명령어들이다.

'push'는 stack 영역에 4byte 공간을 할당하고 해당 값을 할당된 공간에 저장한다. 보통, 뒤에 call되는 함수의 매개변수에 인자값을 전달하기 위해서 사용된다.

'call'은 접근 할 함수로 이동하여 실행한다.

'pop'은 이전 'push'로 저장되어진 4byte 공간을 제거한다.

이를 통해, system call이 발생할 때마다 'print_all_sys_calls()'의 system call함수가 호출되고, '%eax'가 해당 함수의 인자로 사용되어진다는 것을 알 수 있다.



<커널을 재컴파일하고 리부팅을 한 후, 호출되어지는 모든 system call 함수의 번호를 화면에 출력하기위해 'echo 8 > /proc/sys/kernel/printk'코드를 입력하였음, 이후 위 사진처럼 특정 system call 함수의 번호가 끊임없이 출력되는 오류가 발생함>

위 오류가 발생하였지만, 해당 오류를 통해 printf("hello")가 실행되었을 때 이를 통해 호출되어지는 모든 system call 함수들의 number가 화면에 출력될 것임을 예측할 수 있다.