Report

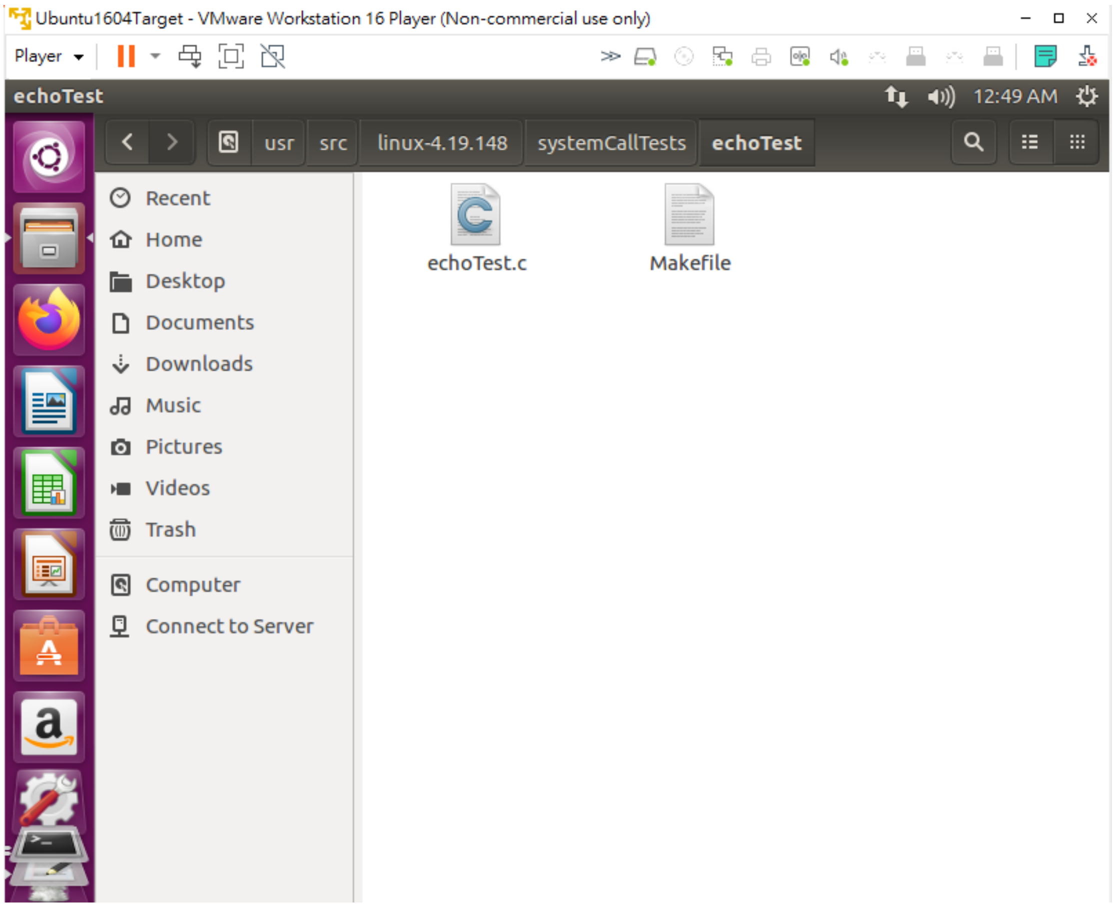
Video link: https://youtu.be/EWh_wRiewiA

Question:

1. kernel space: the location where the kernel is stored and execute
   user space: the location where normal processes(things other than kernel) run
   kernel space runs the privileged instruction to protect the system, user space runs all the other instruction
2. hierarchical protection domains, are the mechanism to protect data and function
   4 rings
   Ring 0: the most privileged and interacted most directly with hardware
   Ring 1: the one that is reserved for hardware device
3. syscall provide an interface for user to access the service of OS
   5 types: process control, file manipulation, device manipulation, information maintenance, and communication.
   process control: control(stop) the process
   file manipulation: every file relating execution, the OS often provide API to make these syscall
   device manipulation: provide resource device for program execution
   information maintenance: transfer information between user, kernel mode
   communication: let two or more processes communicate with each other
4. in the linux-4.19.148/fs folder, namei.c file
5. system call ID is the ID number one called to use the specific module, like mkdir has the syscall Id of 83, and rmdir has the ID of 84
6. asmlinkage: as when system call handler call the system call routine, the value will be pushed into the stack, therefore we have to use asmlinkage to look on the stack, instead of register, to read the parameter passed by system call handler
   printk: a C function from the Linux kernel interface that prints message to the kernel log
   update-initramfs: manage the initramfs images on the local box, keep track of the existing initramfs archives in /boot folder
7. We use printk in the kernel to print out message as we can't use printf in there as there is no c library in kernel
   we have to use "dmesg" function to read message printed by printk
8. it's the ring buffer in the kernel that stores messages related to the operation of the kernel
   we can read its content by calling "dmesg"

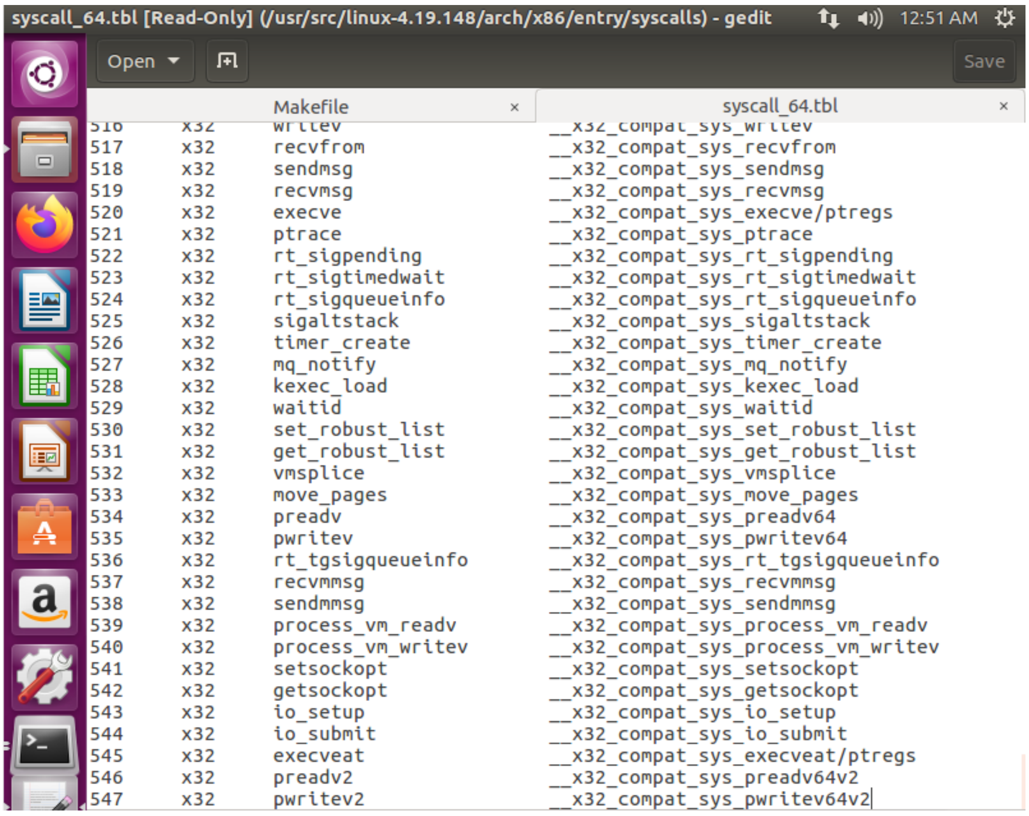
9. Function signature define the input and output of functions or modules
10. It's the start of the module definition, and the [n] means the variable that will be input to the system call
11. 0 input: asmlinkage long xxx(syscall name)(void)
    1 int input: asmlinkage long xxx(syscall name)(int)
    2 int input: asmlinkage long xxx(syscall name)(int, int)
    3 int input: asmlinkage long xxx(syscall name)(int, int, int)
12. No, one that change depending on type of element return is function signature, as mentioned in Q.11
13. Include the library kernel.h: contain some often-used function prototypes, ex printk
    Include the library syscall.h: contain all of the syscalls

Screenshot 1:
Define the kernel modules that we want to implement, and create Makefile
By the command "obj-y := echoTest.o" in the Makefile, we let the kbuild know that the echoTest.o should be made by echoTest.c



Screenshot 2:

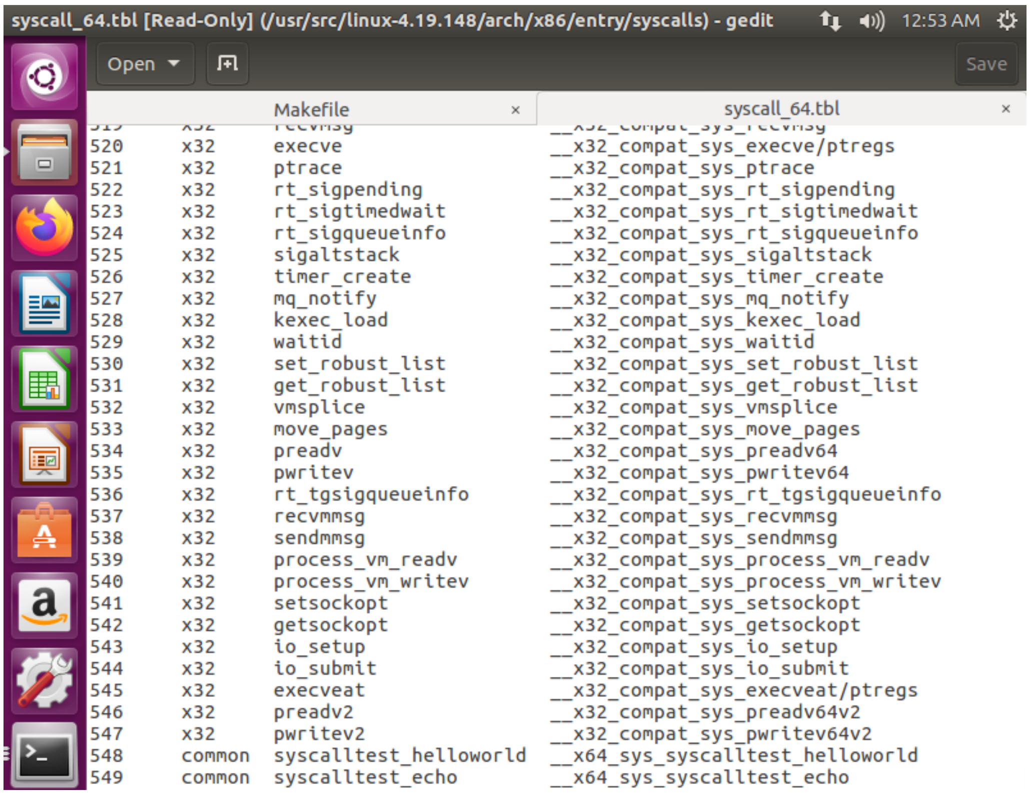
Find the syscall file that contain all of the syscall that is defined


syscall_64.tbl [Read-Only] (/usr/src/linux-4.19.148/arch/x86/entry/syscalls) - gedit
12:51 AM
Open
Save
Makefile
syscall_64.tbl

```
516     x32     writev                  __x32_compat_sys_writev
517     x32     recvfrom                __x32_compat_sys_recvfrom
518     x32     sendmsg                 __x32_compat_sys_sendmsg
519     x32     recvmsg                 __x32_compat_sys_recvmsg
520     x32     execve                  __x32_compat_sys_execve/ptregs
521     x32     ptrace                  __x32_compat_sys_ptrace
522     x32     rt_sigpending           __x32_compat_sys_rt_sigpending
523     x32     rt_sigtimedwait         __x32_compat_sys_rt_sigtimedwait
524     x32     rt_sigqueueinfo         __x32_compat_sys_rt_sigqueueinfo
525     x32     sigaltstack             __x32_compat_sys_sigaltstack
526     x32     timer_create            __x32_compat_sys_timer_create
527     x32     mq_notify               __x32_compat_sys_mq_notify
528     x32     kexec_load              __x32_compat_sys_kexec_load
529     x32     waitid                  __x32_compat_sys_waitid
530     x32     set_robust_list         __x32_compat_sys_set_robust_list
531     x32     get_robust_list         __x32_compat_sys_get_robust_list
532     x32     vmsplice                __x32_compat_sys_vmsplice
533     x32     move_pages              __x32_compat_sys_move_pages
534     x32     preadv                  __x32_compat_sys_preadv64
535     x32     pwritev                 __x32_compat_sys_pwritev64
536     x32     rt_tgsigqueueinfo       __x32_compat_sys_rt_tgsigqueueinfo
537     x32     recvmmsg                __x32_compat_sys_recvmmsg
538     x32     sendmmsg                __x32_compat_sys_sendmmsg
539     x32     process_vm_readv        __x32_compat_sys_process_vm_readv
540     x32     process_vm_writev       __x32_compat_sys_process_vm_writev
541     x32     setsockopt              __x32_compat_sys_setsockopt
542     x32     getsockopt              __x32_compat_sys_getsockopt
543     x32     io_setup                __x32_compat_sys_io_setup
544     x32     io_submit               __x32_compat_sys_io_submit
545     x32     execveat                __x32_compat_sys_execveat/ptregs
546     x32     preadv2                 __x32_compat_sys_preadv64v2
547     x32     pwritev2                __x32_compat_sys_pwritev64v2
```

Screenshot 3:

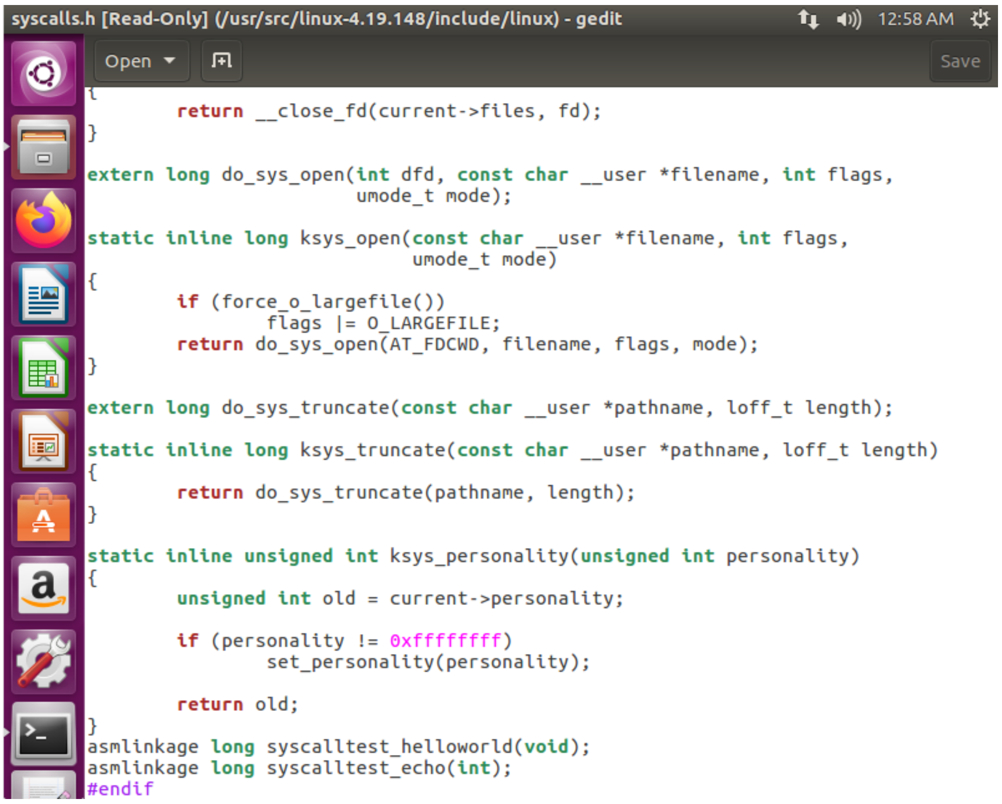
Add the syscall that we want to implement into the syscall file


syscall_64.tbl [Read-Only] (/usr/src/linux-4.19.148/arch/x86/entry/syscalls) - gedit
12:53 AM
Open
Save
Makefile
syscall_64.tbl

```
519     x32     recvmsg                 __x32_compat_sys_recvmsg
520     x32     execve                  __x32_compat_sys_execve/ptregs
521     x32     ptrace                  __x32_compat_sys_ptrace
522     x32     rt_sigpending           __x32_compat_sys_rt_sigpending
523     x32     rt_sigtimedwait         __x32_compat_sys_rt_sigtimedwait
524     x32     rt_sigqueueinfo         __x32_compat_sys_rt_sigqueueinfo
525     x32     sigaltstack             __x32_compat_sys_sigaltstack
526     x32     timer_create            __x32_compat_sys_timer_create
527     x32     mq_notify               __x32_compat_sys_mq_notify
528     x32     kexec_load              __x32_compat_sys_kexec_load
529     x32     waitid                  __x32_compat_sys_waitid
530     x32     set_robust_list         __x32_compat_sys_set_robust_list
531     x32     get_robust_list         __x32_compat_sys_get_robust_list
532     x32     vmsplice                __x32_compat_sys_vmsplice
533     x32     move_pages              __x32_compat_sys_move_pages
534     x32     preadv                  __x32_compat_sys_preadv64
535     x32     pwritev                 __x32_compat_sys_pwritev64
536     x32     rt_tgsigqueueinfo       __x32_compat_sys_rt_tgsigqueueinfo
537     x32     recvmmsg                __x32_compat_sys_recvmmsg
538     x32     sendmmsg                __x32_compat_sys_sendmmsg
539     x32     process_vm_readv        __x32_compat_sys_process_vm_readv
540     x32     process_vm_writev       __x32_compat_sys_process_vm_writev
541     x32     setsockopt              __x32_compat_sys_setsockopt
542     x32     getsockopt              __x32_compat_sys_getsockopt
543     x32     io_setup                __x32_compat_sys_io_setup
544     x32     io_submit               __x32_compat_sys_io_submit
545     x32     execveat                __x32_compat_sys_execveat/ptregs
546     x32     preadv2                 __x32_compat_sys_preadv64v2
547     x32     pwritev2                __x32_compat_sys_pwritev64v2
548     common  syscalltest_helloworld  __x64_sys_syscalltest_helloworld
549     common  syscalltest_echo        __x64_sys_syscalltest_echo
```

Screenshot 4:

Add our function signature to the syscall file in /linux folder, define the input data type

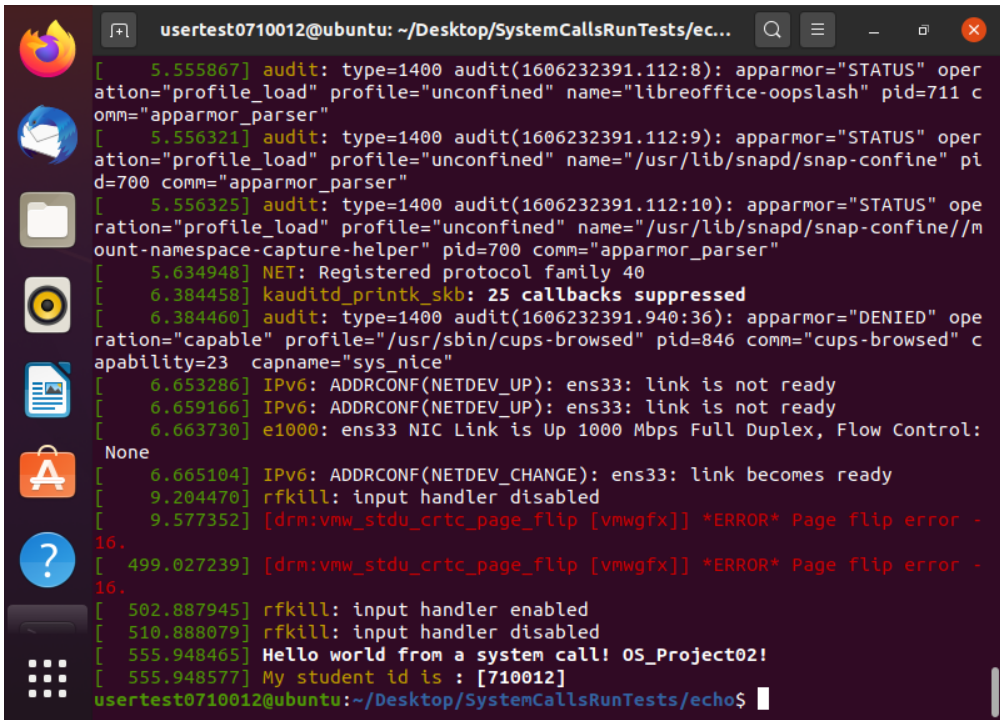


syscalls.h [Read-Only] (/usr/src/linux-4.19.148/include/linux) - gedit 12:58 AM

```
{
        return __close_fd(current->files, fd);
}

extern long do_sys_open(int dfd, const char __user *filename, int flags,
                        umode_t mode);

static inline long ksys_open(const char __user *filename, int flags,
                             umode_t mode)
{
        if (force_o_largefile())
                flags |= O_LARGEFILE;
        return do_sys_open(AT_FDCWD, filename, flags, mode);
}

extern long do_sys_truncate(const char __user *pathname, loff_t length);

static inline long ksys_truncate(const char __user *pathname, loff_t length)
{
        return do_sys_truncate(pathname, length);
}

static inline unsigned int ksys_personality(unsigned int personality)
{
        unsigned int old = current->personality;

        if (personality != 0xffffffff)
                set_personality(personality);

        return old;
}
asmlinkage long syscalltest_helloworld(void);
asmlinkage long syscalltest_echo(int);
#endif
```

Screenshot 5:

As we use printk in out syscall, the message typed in the module will be print out in kernel ring buffer

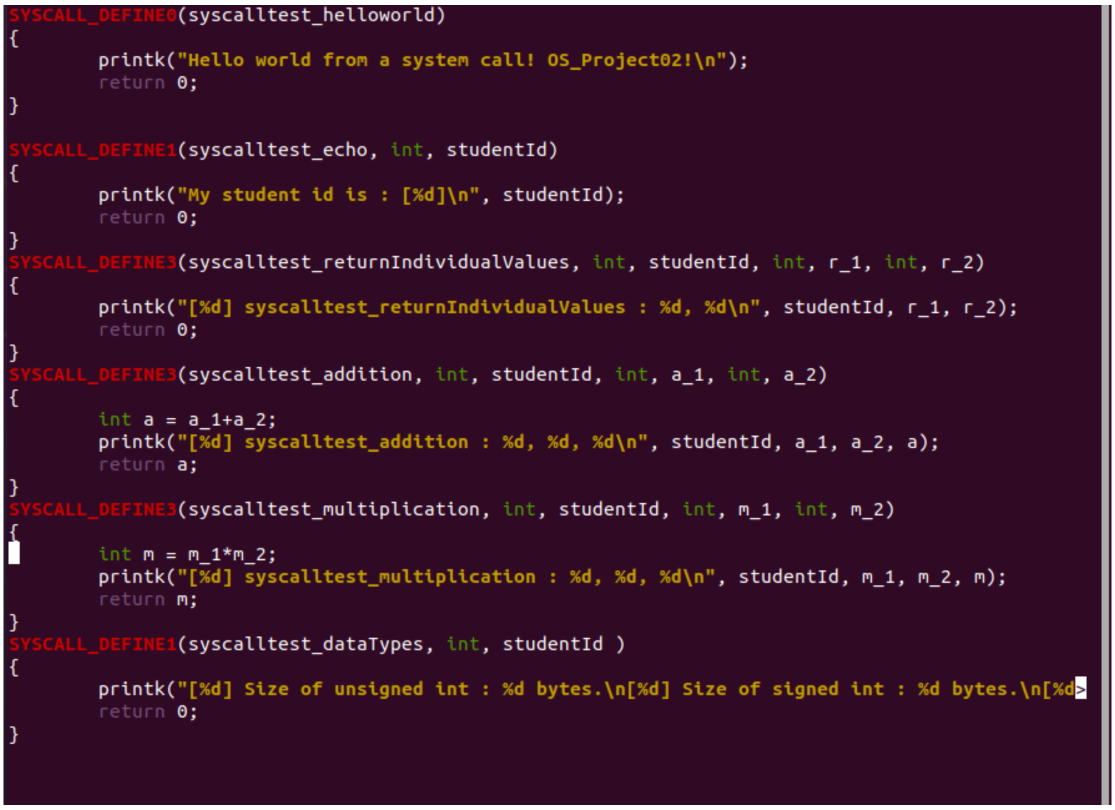


usertest0710012@ubuntu: ~/Desktop/SystemCallsRunTests/ec...

```
[    5.555867] audit: type=1400 audit(1606232391.112:8): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="libreoffice-oopslash" pid=711 c
omm="apparmor_parser"
[    5.556321] audit: type=1400 audit(1606232391.112:9): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="/usr/lib/snapd/snap-confine" pi
d=700 comm="apparmor_parser"
[    5.556325] audit: type=1400 audit(1606232391.112:10): apparmor="STATUS" ope
ration="profile_load" profile="unconfined" name="/usr/lib/snapd/snap-confine//m
ount-namespace-capture-helper" pid=700 comm="apparmor_parser"
[    5.634948] NET: Registered protocol family 40
[    6.384458] kauditd_printk_skb: 25 callbacks suppressed
[    6.384460] audit: type=1400 audit(1606232391.940:36): apparmor="DENIED" ope
ration="capable" profile="/usr/sbin/cups-browsed" pid=846 comm="cups-browsed" c
apability=23  capname="sys_nice"
[    6.653286] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[    6.659166] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[    6.663730] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
 None
[    6.665104] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[    9.204470] rfkill: input handler disabled
[    9.577352] [drm:vmw_stdu_crtc_page_flip [vmwgfx]] *ERROR* Page flip error -
16.
[  499.027239] [drm:vmw_stdu_crtc_page_flip [vmwgfx]] *ERROR* Page flip error -
16.
[  502.887945] rfkill: input handler enabled
[  510.888079] rfkill: input handler disabled
[  555.948465] Hello world from a system call! OS_Project02!
[  555.948577] My student id is : [710012]
usertest0710012@ubuntu:~/Desktop/SystemCallsRunTests/echo$
```

Screenshot 6:

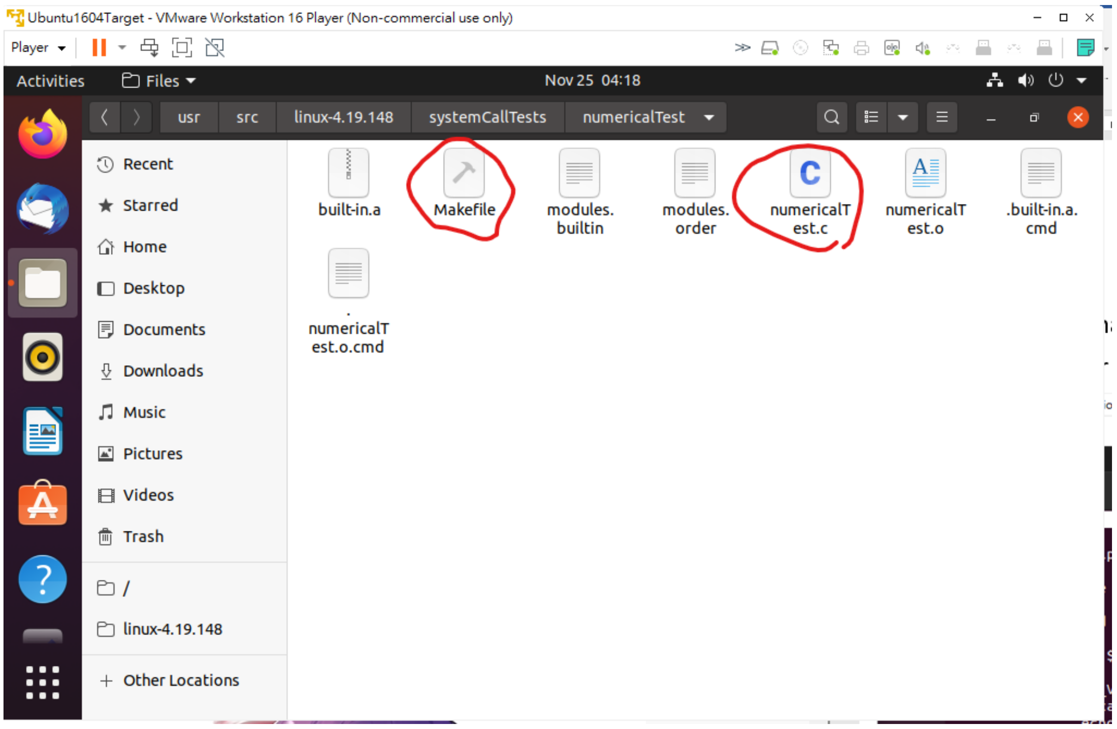
Add the rest of the syscalls we want to implement in the numericalTest.c



```
SYSCALL_DEFINE0(syscalltest_helloworld)
{
        printk("Hello world from a system call! OS_Project02!\n");
        return 0;
}

SYSCALL_DEFINE1(syscalltest_echo, int, studentId)
{
        printk("My student id is : [%d]\n", studentId);
        return 0;
}
SYSCALL_DEFINE3(syscalltest_returnIndividualValues, int, studentId, int, r_1, int, r_2)
{
        printk("[%d] syscalltest_returnIndividualValues : %d, %d\n", studentId, r_1, r_2);
        return 0;
}
SYSCALL_DEFINE3(syscalltest_addition, int, studentId, int, a_1, int, a_2)
{
        int a = a_1+a_2;
        printk("[%d] syscalltest_addition : %d, %d, %d\n", studentId, a_1, a_2, a);
        return a;
}
SYSCALL_DEFINE3(syscalltest_multiplication, int, studentId, int, m_1, int, m_2)
{
        int m = m_1*m_2;
        printk("[%d] syscalltest_multiplication : %d, %d, %d\n", studentId, m_1, m_2, m);
        return m;
}
SYSCALL_DEFINE1(syscalltest_dataTypes, int, studentId )
{
        printk("[%d] Size of unsigned int : %d bytes.\n[%d] Size of signed int : %d bytes.\n[%d>
        return 0;
}
```

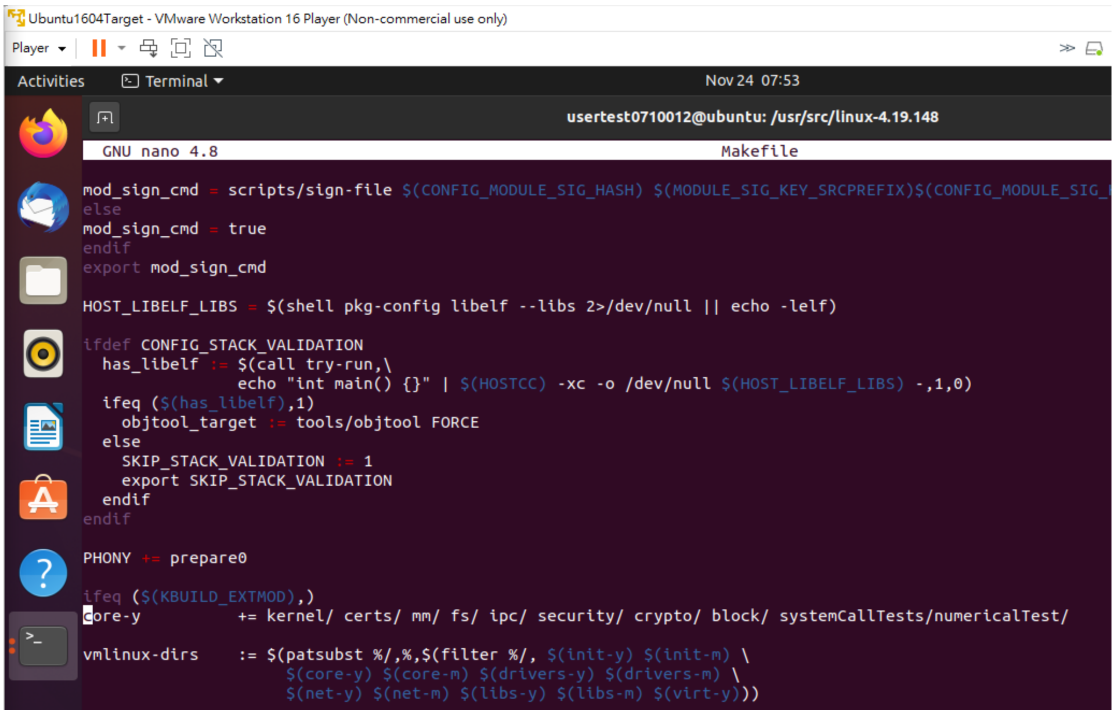
Screenshot 7:

Create a Makefile with a command “obj-y := numericalTest.o” in the numericalTest folder with the numericalTest.c alike



Screenshot 8:

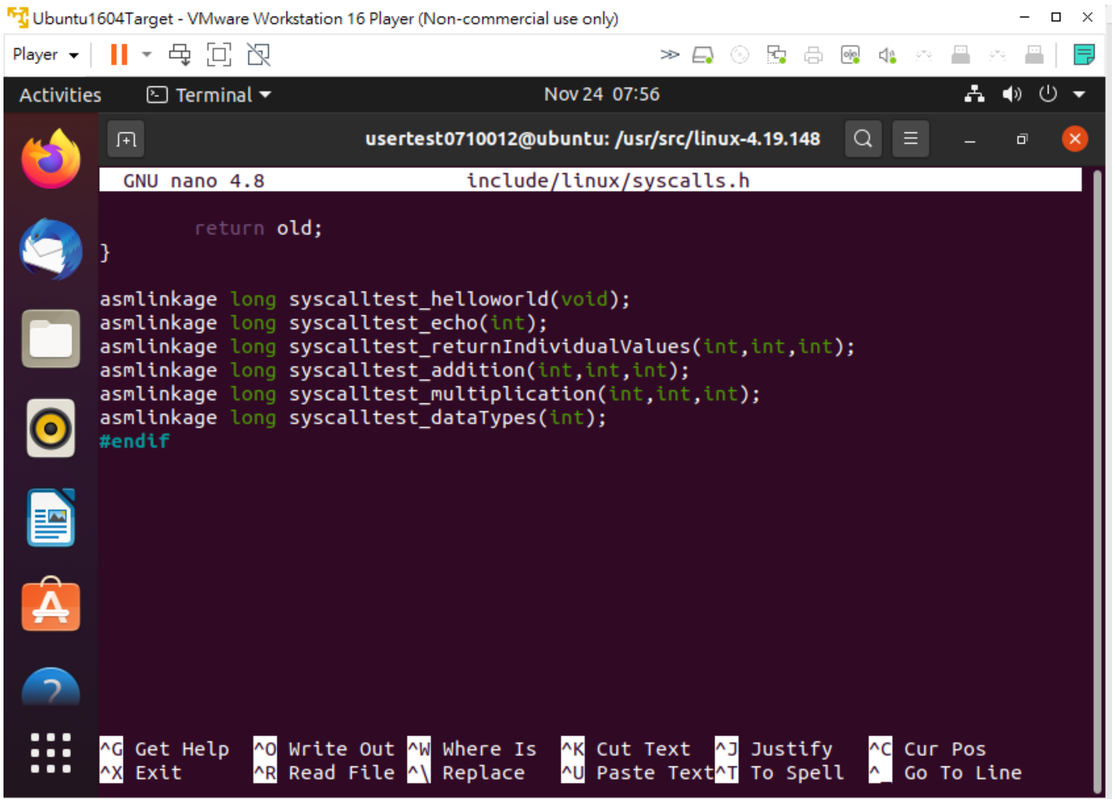
Change the folder that we previous implement the syscalls in echoTest folder to numericalTest folder by modify the command in Makefile



```
mod_sign_cmd = scripts/sign-file $(CONFIG_MODULE_SIG_HASH) $(MODULE_SIG_KEY_SRCPREFIX)$(CONFIG_MODULE_SIG_
else
mod_sign_cmd = true
endif
export mod_sign_cmd

HOST_LIBELF_LIBS = $(shell pkg-config libelf --libs 2>/dev/null || echo -lelf)

ifdef CONFIG_STACK_VALIDATION
  has_libelf := $(call try-run,\
                echo "int main() {}" | $(HOSTCC) -xc -o /dev/null $(HOST_LIBELF_LIBS) -,1,0)
  ifeq ($(has_libelf),1)
    objtool_target := tools/objtool FORCE
  else
    SKIP_STACK_VALIDATION := 1
    export SKIP_STACK_VALIDATION
  endif
endif

PHONY += prepare0

ifeq ($(KBUILD_EXTMOD),)
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ systemCallTests/numericalTest/

vmlinux-dirs    := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \
                     $(core-y) $(core-m) $(drivers-y) $(drivers-m) \
                     $(net-y) $(net-m) $(libs-y) $(libs-m) $(virt-y)))
```

Screenshot 9:

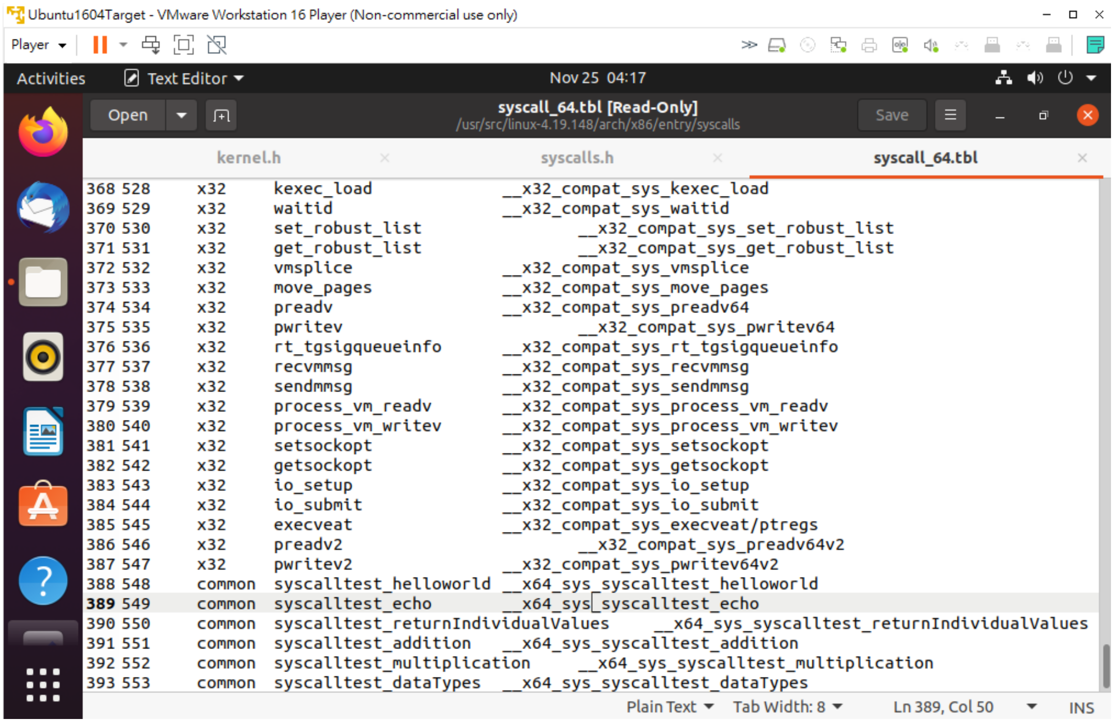
Add the rest of the syscall into the syscall.h file in /linux, and define the number of input parameters and their type



```
        return old;
}

asmlinkage long syscalltest_helloworld(void);
asmlinkage long syscalltest_echo(int);
asmlinkage long syscalltest_returnIndividualValues(int,int,int);
asmlinkage long syscalltest_addition(int,int,int);
asmlinkage long syscalltest_multiplication(int,int,int);
asmlinkage long syscalltest_dataTypes(int);
#endif
```

Screenshot 10:

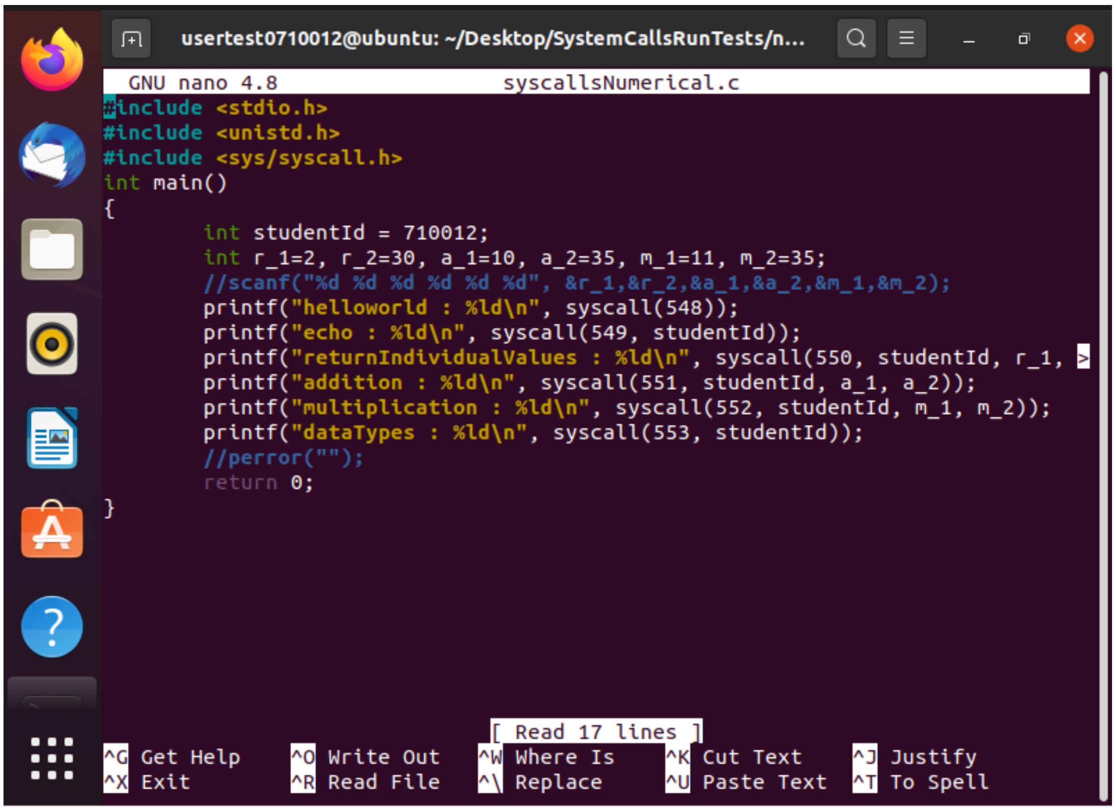
Add the rest of the syscalls we want to implement in the syscall file


Ubuntu1604Target - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Text Editor
Nov 25 04:17
Open
Save
syscall_64.tbl [Read-Only]
/usr/src/linux-4.19.148/arch/x86/entry/syscalls
kernel.h
syscalls.h
syscall_64.tbl
Plain Text
Tab Width: 8
Ln 389, Col 50
INS

```
368 528   x32      kexec_load             __x32_compat_sys_kexec_load
369 529   x32      waitid                 __x32_compat_sys_waitid
370 530   x32      set_robust_list                __x32_compat_sys_set_robust_list
371 531   x32      get_robust_list                __x32_compat_sys_get_robust_list
372 532   x32      vmsplice               __x32_compat_sys_vmsplice
373 533   x32      move_pages             __x32_compat_sys_move_pages
374 534   x32      preadv                 __x32_compat_sys_preadv64
375 535   x32      pwritev                        __x32_compat_sys_pwritev64
376 536   x32      rt_tgsigqueueinfo      __x32_compat_sys_rt_tgsigqueueinfo
377 537   x32      recvmmsg               __x32_compat_sys_recvmmsg
378 538   x32      sendmmsg               __x32_compat_sys_sendmmsg
379 539   x32      process_vm_readv       __x32_compat_sys_process_vm_readv
380 540   x32      process_vm_writev      __x32_compat_sys_process_vm_writev
381 541   x32      setsockopt             __x32_compat_sys_setsockopt
382 542   x32      getsockopt             __x32_compat_sys_getsockopt
383 543   x32      io_setup               __x32_compat_sys_io_setup
384 544   x32      io_submit              __x32_compat_sys_io_submit
385 545   x32      execveat               __x32_compat_sys_execveat/ptregs
386 546   x32      preadv2                        __x32_compat_sys_preadv64v2
387 547   x32      pwritev2               __x32_compat_sys_pwritev64v2
388 548   common   syscalltest_helloworld __x64_sys_syscalltest_helloworld
389 549   common   syscalltest_echo       __x64_sys_syscalltest_echo
390 550   common   syscalltest_returnIndividualValues      __x64_sys_syscalltest_returnIndividualValues
391 551   common   syscalltest_addition   __x64_sys_syscalltest_addition
392 552   common   syscalltest_multiplication     __x64_sys_syscalltest_multiplication
393 553   common   syscalltest_dataTypes  __x64_sys_syscalltest_dataTypes
```

Screenshot 11:

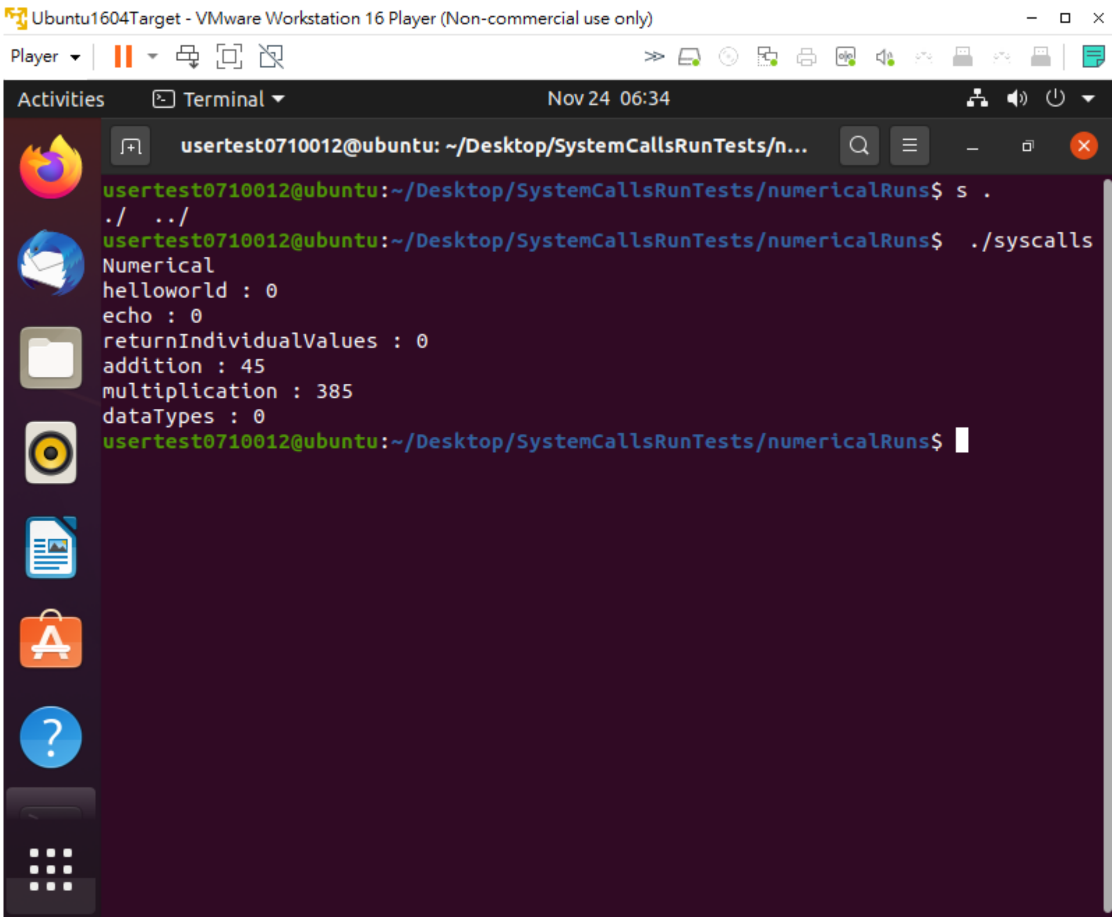
Call the syscalls we previously defined and give them parameters that the modules needed



```c
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
int main()
{
        int studentId = 710012;
        int r_1=2, r_2=30, a_1=10, a_2=35, m_1=11, m_2=35;
        //scanf("%d %d %d %d %d %d", &r_1,&r_2,&a_1,&a_2,&m_1,&m_2);
        printf("helloworld : %ld\n", syscall(548));
        printf("echo : %ld\n", syscall(549, studentId));
        printf("returnIndividualValues : %ld\n", syscall(550, studentId, r_1, >
        printf("addition : %ld\n", syscall(551, studentId, a_1, a_2));
        printf("multiplication : %ld\n", syscall(552, studentId, m_1, m_2));
        printf("dataTypes : %ld\n", syscall(553, studentId));
        //perror("");
        return 0;
}
```

Screenshot 12:

The program successfully print out the message and the return values of the syscalls

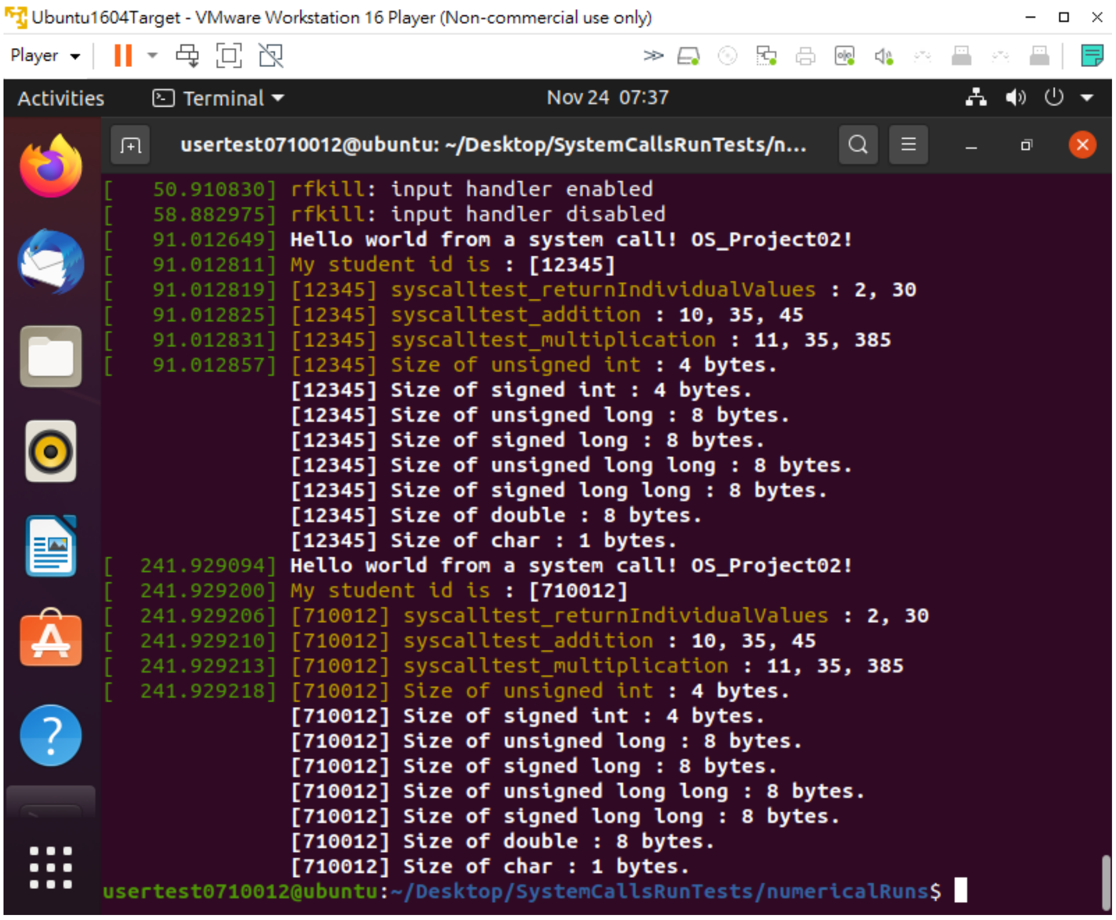


Ubuntu1604Target - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Nov 24 06:34
usertest0710012@ubuntu: ~/Desktop/SystemCallsRunTests/n...

```
usertest0710012@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$ s .
./  ../
usertest0710012@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$  ./syscalls
Numerical
helloworld : 0
echo : 0
returnIndividualValues : 0
addition : 45
multiplication : 385
dataTypes : 0
usertest0710012@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$ 
```

Screenshot 13:

We call the dmesg to see the kernel ring buffer and see that our message have been print out successfully; however, the form is a bit differnet, as the example given have the green part(timer) in the front of every datatypes output, I thought that the timer will be printed out when the syscall is called, but in the user mode(c program in Desktop), the example shown that the syscall has only been called once, moreover, the output of the syscall won't be different if I only input my studentId, so I just outputall of the required data at once, hope that the output meets the requirement.


Ubuntu1604Target - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Nov 24 07:37
usertest0710012@ubuntu: ~/Desktop/SystemCallsRunTests/n...

```
[   50.910830] rfkill: input handler enabled
[   58.882975] rfkill: input handler disabled
[   91.012649] Hello world from a system call! OS_Project02!
[   91.012811] My student id is : [12345]
[   91.012819] [12345] syscalltest_returnIndividualValues : 2, 30
[   91.012825] [12345] syscalltest_addition : 10, 35, 45
[   91.012831] [12345] syscalltest_multiplication : 11, 35, 385
[   91.012857] [12345] Size of unsigned int : 4 bytes.
               [12345] Size of signed int : 4 bytes.
               [12345] Size of unsigned long : 8 bytes.
               [12345] Size of signed long : 8 bytes.
               [12345] Size of unsigned long long : 8 bytes.
               [12345] Size of signed long long : 8 bytes.
               [12345] Size of double : 8 bytes.
               [12345] Size of char : 1 bytes.
[  241.929094] Hello world from a system call! OS_Project02!
[  241.929200] My student id is : [710012]
[  241.929206] [710012] syscalltest_returnIndividualValues : 2, 30
[  241.929210] [710012] syscalltest_addition : 10, 35, 45
[  241.929213] [710012] syscalltest_multiplication : 11, 35, 385
[  241.929218] [710012] Size of unsigned int : 4 bytes.
               [710012] Size of signed int : 4 bytes.
               [710012] Size of unsigned long : 8 bytes.
               [710012] Size of signed long : 8 bytes.
               [710012] Size of unsigned long long : 8 bytes.
               [710012] Size of signed long long : 8 bytes.
               [710012] Size of double : 8 bytes.
               [710012] Size of char : 1 bytes.
usertest0710012@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$
```