

Operating System

project 2

Department: 土木5B

ID: 0511330

Name: 劉紘華

Vedio Link: <https://youtu.be/xgIZucagqng>

Objective:

To help the student to get familiar with system calls, including the definition, usage and implementation of a system call.

[Section 1-2]

[Screenshot 1] Defining a new system call.

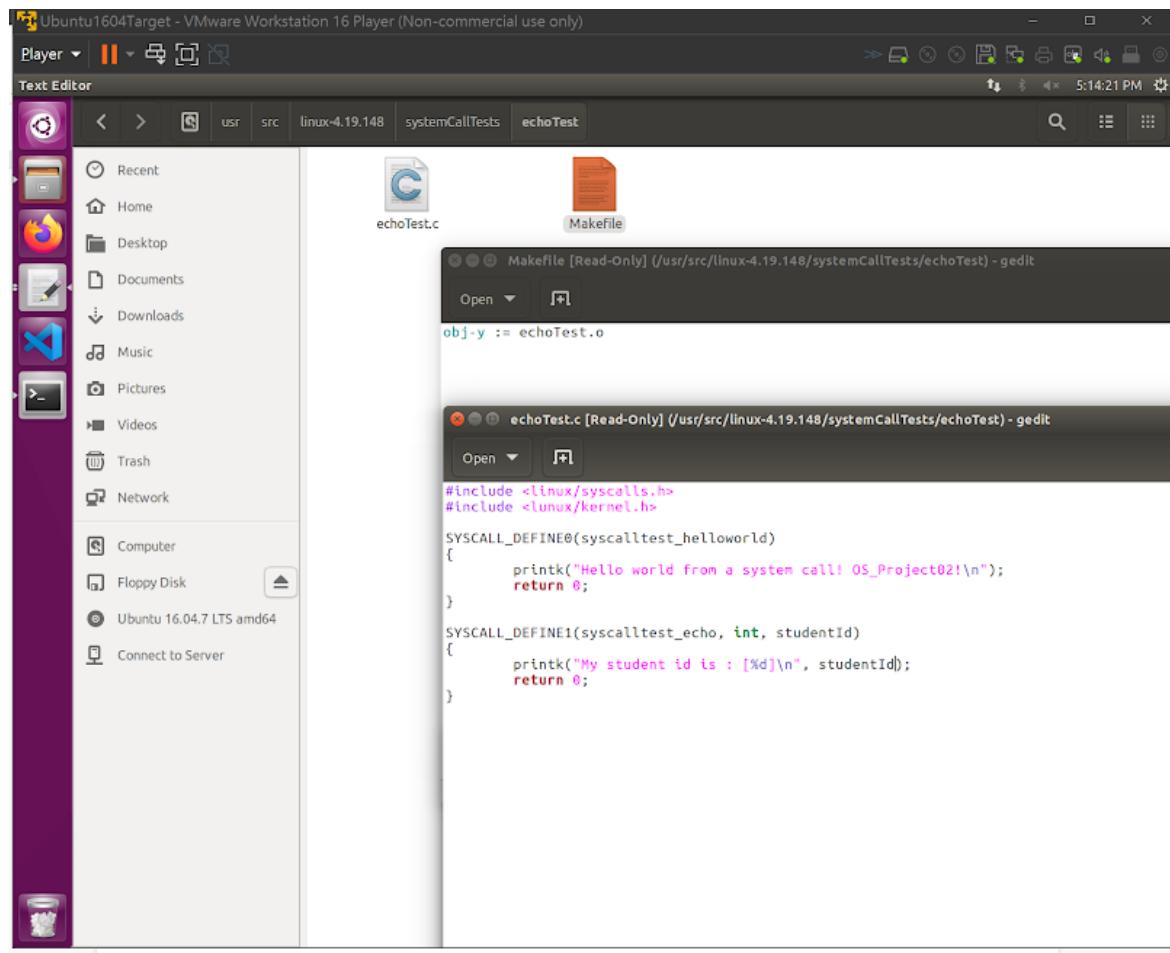
1. In order to place our custom system call code, create a new folder **systemCallTests** and a subfolder **echoTest**.

(in the kernel folder: **/usr/src/linux-4.19.148**)

[Command: **\$sudo mkdir systemCallTests**, and so does echoTest.]

2. Create a C file **echoTest.c** to implement our system calls, and a text file **Makefile** to ensure that our system calls will be compiled.

[Command: **\$sudo gedit echoTest.c**, and so does **Makefile**]



[Note]

We also need to notify the compiler where our c file is, i.e., which folder our files will be. So append the folder name **syscallTests/echoTest/** to the end of the line **core-y += kernel/ certs/ mm/...** in the Makefile, which should be in the kernel folder **/usr/src/linux-4.19.148**.

[Screenshot 2, 3] Modify syscall_64.tbl

Adding our system call into the system call table **syscall_64.tbl**, to let the user program can access it.

[Command: `$sudo gedit syscall_64.tbl`]

(under the path: /linux-4.19.148/arch/x86/entry/syscalls)

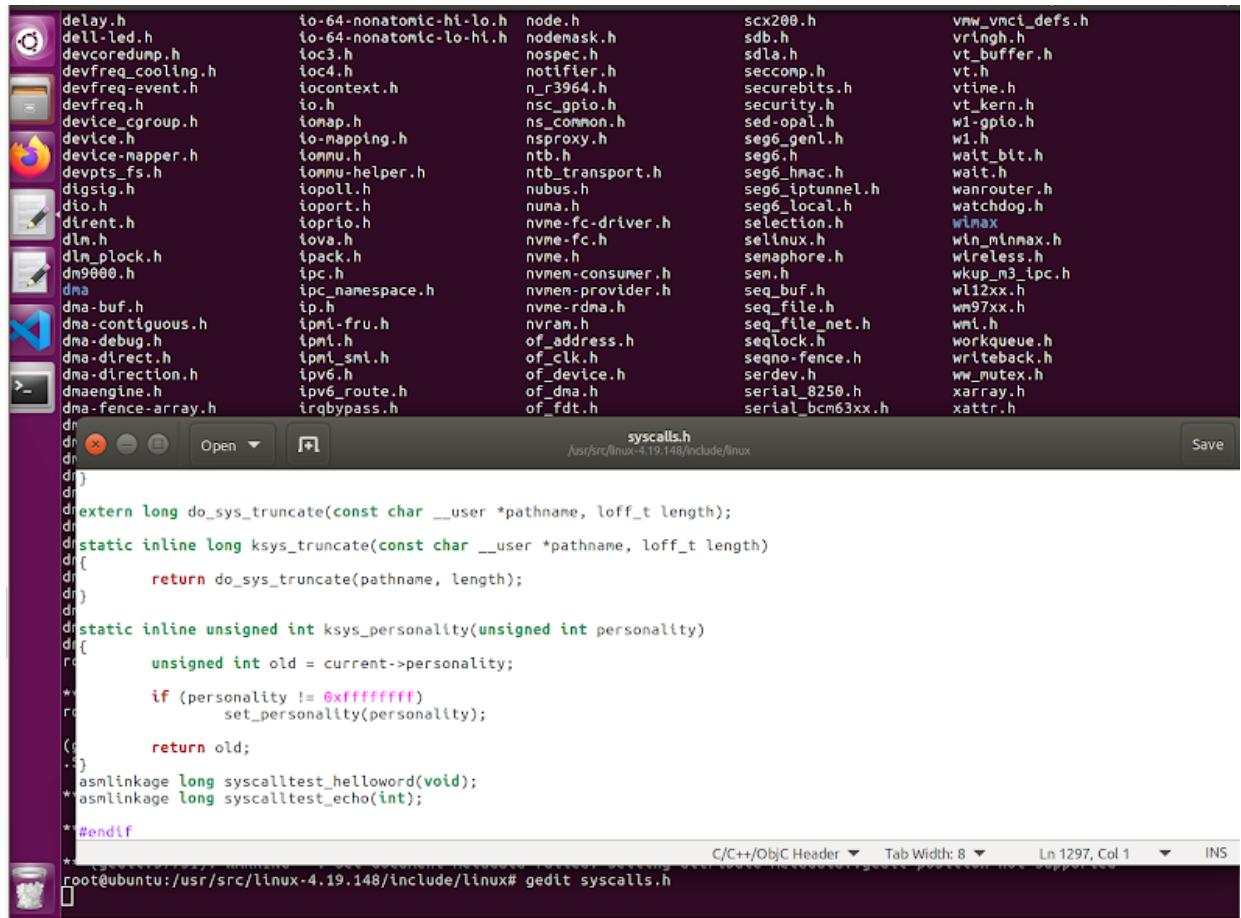
before

after

[Screenshot 4] syscalls.h

Define the prototype of the function in **syscalls.h**. “asmlinkage” is a keyword used to indicate that all parameters of the function would be available on the stack.

[command: **\$sudo gedit syscalls.h**, path: **/usr/src/include/linux**]



The screenshot shows a terminal window with a file browser sidebar on the left. The sidebar lists various header files such as delay.h, devfreq.h, device.h, devpts.h, digsig.h, dirent.h, dlm.h, dn9000.h, dna.h, dna-buf.h, dna-contiguous.h, dna-debug.h, dna-direct.h, dna-direction.h, dna-engine.h, dna-fence-array.h, io-64-nonatomic-hi-lo.h, ioc3.h, ioc4.h, iocontext.h, io.h, iomap.h, to-mapping.h, tommu.h, tommu-helper.h, iopoll.h, ioport.h, toprio.h, tova.h, tpack.h, ipc.h, ipc_namespace.h, ip.h, ipmi-fru.h, ipml.h, ipml_smt.h, ipv6.h, ipv6_route.h, irqbypass.h, node.h, nodemask.h, nospec.h, notifier.h, n_3964.h, nsc_gpio.h, ns_common.h, nsproxy.h, ntb.h, ntb_transport.h, nubus.h, numa.h, nvme-fc-driver.h, nvme-fc.h, nvme.h, nvmem-consumer.h, nvmem-provider.h, nvme-rdma.h, nvram.h, of_address.h, of_clk.h, of_device.h, of_dma.h, of_fdt.h, scx200.h, sdb.h, sdla.h, seccomp.h, securebits.h, security.h, sed-opal.h, seq6_genlh.h, seq6.h, seq6_hmac.h, seq6_ip tunnel.h, seq6_local.h, selection.h, selinux.h, semaphore.h, sem.h, seq_buf.h, seq_file.h, seq_file_net.h, seqlock.h, seqno-fence.h, serdev.h, serial_8250.h, serial_bcm63xx.h, vmw_vmc_defs.h, vrings.h, vt_buffer.h, vt.h, vtime.h, vt_kern.h, wi-gpio.h, wi.h, wait_bit.h, wait.h, wanrouter.h, watchdog.h, wimax, wln_minmax.h, wlreless.h, wkup_m3_ipc.h, wl12xx.h, wn97xx.h, wni.h, workqueue.h, writeback.h, ww_mutex.h, xarray.h, xattr.h.

```
syscalls.h
/usr/src/linux-4.19.148/include/linux

drwxr-xr-x  2 root root   4096 Jan  1  1970 .
drwxr-xr-x 12 root root   4096 Jan  1  1970 ..
drwxr-xr-x  2 root root   4096 Jan  1  1970 asm
drwxr-xr-x  2 root root   4096 Jan  1  1970 blkdev
drwxr-xr-x  2 root root   4096 Jan  1  1970 block
drwxr-xr-x  2 root root   4096 Jan  1  1970 driver
drwxr-xr-x  2 root root   4096 Jan  1  1970 fs
drwxr-xr-x  2 root root   4096 Jan  1  1970 include
drwxr-xr-x  2 root root   4096 Jan  1  1970 ipc
drwxr-xr-x  2 root root   4096 Jan  1  1970 lib
drwxr-xr-x  2 root root   4096 Jan  1  1970 mm
drwxr-xr-x  2 root root   4096 Jan  1  1970 net
drwxr-xr-x  2 root root   4096 Jan  1  1970 sound
drwxr-xr-x  2 root root   4096 Jan  1  1970 sysfs
drwxr-xr-x  2 root root   4096 Jan  1  1970 uapi
drwxr-xr-x  2 root root   4096 Jan  1  1970 udev
drwxr-xr-x  2 root root   4096 Jan  1  1970 uinput
drwxr-xr-x  2 root root   4096 Jan  1  1970 virtio
drwxr-xr-x  2 root root   4096 Jan  1  1970 xfs

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
root@ubuntu:/usr/src/linux-4.19.148/include/linux# gedit syscalls.h
```

[Screenshot 5] Test syscalls whether it works or not

After recompile and reboot the system, create a C file to test if the system call we made works.

The screenshot shows a Linux desktop environment with several windows open:

- Terminal Window:** Displays the output of a command-line session. It includes kernel logs (e.g., network interface status, Bluetooth initialization) and user input where the user runs the program `./syscallsHelloEco`. The output shows the program's execution and the printed messages "helloworld : 0" and "echo : 0".
- File Browser:** Shows the file tree starting from the root directory. It lists various system files and folders like `Documents`, `Downloads`, `Music`, `Pictures`, `Videos`, `Trash`, `Computer`, `Floppy Disk`, and `Connect to Server`.
- Code Editor:** An instance of Gedit is open, displaying the source code of the C program `syscallsHelloEco.c`. The code includes standard library includes, a main function that prints "helloworld" and "echo" using system calls, and a student ID assignment.

```
C syscallsHelloEco.c > ...
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/syscall.h>
4
5 int main() {
6     int studentId = 511330; //0511330
7
8     printf("helloworld : %ld\n", syscall(335));
9     printf("echo : %ld\n", syscall(336, studentId));
10
11     return 0;
12 }
13
```

```
[ 4364.874388] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[ 5136.980028] e1000: ens33 NIC Link is Down
[ 5141.012705] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[ 5143.027893] e1000: ens33 NIC Link is Down
[ 5147.061087] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[ 5235.762768] e1000: ens33 NIC Link is Down
[ 5239.798383] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
[ 8354.790853] perf: interrupt took too long (2548 > 2500), lowering kernel.perf_event_max_sample_rate to 7825
[ 8473.905561] Hello world from a system call! OS_Project02!
[ 8473.905614] My student id is : [511330]
user@test0511330@ubuntu:~/Desktop/SystemCallsRunTests/echo$ ./syscallsHelloEco
helloworld : 0
echo : 0
user@test0511330@ubuntu:~/Desktop/SystemCallsRunTests/echo$
```

```
[ 9.215363] Adding 998590K Swap on /dev/sda5.  Priority:-2 extents:1 across:998590
k FS
9.215363] Decoding supported only on Scalable MCA processors.
9.468695] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
9.468697] Bluetooth: BNEP filters: protocol multicast
9.468703] Bluetooth: BNEP socket layer initialized
9.797344] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
9.806108] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
9.812956] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None
9.814191] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
11.812400] Bluetooth: RFCOMM TTY layer initialized
11.812407] Bluetooth: RFCOMM socket layer initialized
11.812413] Bluetooth: RFCOMM ver 1.11
[ 174.289435] Hello world from a system call! OS_Project02!
[ 174.289507] My student id is : [511330]
user@test0511330@ubuntu:~/Desktop/SystemCallsRunTests/echo$
```

```
Documents
Downloads
Music
Pictures
Videos
Trash
Computer
Floppy Disk
Connect to Server
```

```
*syscallsHelloEco.c (~/Desktop/SystemCallsRunTests/echo) - gedit
Open ▾
```

```
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>

int main() {
    int studentId = 511330; //0511330

    printf("helloworld : %ld\n", syscall(335));
    printf("echo : %ld\n", syscall(336, studentId));

    return 0;
}
```

[Section 1-3]

[Screenshot 6] numericalTest.c (syscall)

The number in the function name **SYSCALL_DEFINE3** means this system call contains **3** parameters.

1. **syscalltest_returnIndividualValues**

It contains 3 integer parameters: **studentId**, **a** and **b**, and does the following things:

- (i) Print **studentId**, **a**, and **b** in the kernel ring buffer.
- (ii) Return value 0 to where it was called.

2. **syscalltest_addition**

It contains 3 integer parameters: **studentId**, **a** and **b**, and does the following things:

- (i) Print **studentId**, **a**, **b** and **a+b** in the kernel ring buffer.
- (ii) Return value **a+b** to where it was called.

3. **syscalltest_multiplication**

It contains 3 integer parameters: **studentId**, **a** and **b**, and does the following things:

- (i) Print **studentId**, **a**, **b** and **a*b** in the kernel ring buffer.
- (ii) Return value **a*b** to where it was called. (refer to [Note])

4. **syscalltest_dataTypes**

It contains 1 integer parameters: **studentId**, and does the following things:

- (i) Print the size of basic data type in the kernel ring buffer.
(by **sizeof** operator)
- (ii) Return value 0 to where it was called.

[Note]

If a system call **returns a negative value**, it's treated as an **error**. A special error handling code invoked in libc. So it returns **-1** instead of our expectation when it returns a negative value.

We need to handle this situation by passing a pointer containing our result to a kernel function **copy_to_user**, and it will copy the pointer in the kernel area to user space, then we can get the correct return value in user space which is negative.

```

//STUDENT ID: 0511330

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

SYSCALL_DEFINE3(syscalltest_returnIndividualValues, int, studentId, int, a, int, b)
{
    printk("[%d] syscalltest_returnIndividualValues : %d, %d\n", studentId, a, b);
    return 0;
}

SYSCALL_DEFINE3(syscalltest_addition, int, studentId, int, a, int, b)
{
    printk("[%d] syscalltest_addition : %d, %d, %d\n", studentId, a, b, a+b);
    return a+b;
}

SYSCALL_DEFINE3(syscalltest_multiplication, int, studentId, int, a, int, b)
{
    printk("[%d] syscalltest_multiplication : %d, %d, %d\n", studentId, a, b, a*b);
    return a*b;
}

SYSCALL_DEFINE1(syscalltest_dataTypes, int, studentId) {
    printk("[%d] Size of unsigned int : %d bytes.\n" , studentId, sizeof(unsigned int));
    printk("[%d] Size of signed int : %d bytes.\n" , studentId, sizeof(signed int));
    printk("[%d] Size of unsigned long : %d bytes.\n" , studentId, sizeof(unsigned long));
    printk("[%d] Size of signed long : %d bytes.\n" , studentId, sizeof(signed long));
    printk("[%d] Size of unsigned long long : %d bytes.\n" , studentId, sizeof(unsigned long long));
    printk("[%d] Size of signed long long : %d bytes.\n" , studentId, sizeof(signed long long));
    printk("[%d] Size of double : %d bytes.\n" , studentId, sizeof(double));
    printk("[%d] Size of char : %d bytes.\n" , studentId, sizeof(char));
    return 0;
}

```

[Screenshot 7] Create the Makefile for our syscalls.

Create a Makefile and fill it with line: **obj-y := numericalTest.o**

As mentioned, This is to make sure that our file (**numericalTest.c**) will be compiled.

[Command: **\$sudo gedit Makefile**]

[folder: **/linux-4.19.148/sysCallTests/numericalTest**]

```

Makefile
/usr/src/linux-4.19.148/systemCallTests/numericalTest

obj-y := numericalTest.o

user0511330@ubuntu: /usr/src/linux-4.19.148/systemCallTests/numericalTest
[sudo] password for user0511330:

(gedit:7330): IBUS-WARNING **: The owner of /home/user0511330/.config/ibus/bus is not root!

(gedit:7330): IBUS-WARNING **: Unable to connect to ibus: Unexpected lack of content trying to read a line

** (gedit:7330): WARNING **: Set document metadata failed: Setting attribute metadata ::gedit-position not supported
user0511330@ubuntu:/usr/src/linux-4.19.148$ cd systemCallTests
user0511330@ubuntu:/usr/src/linux-4.19.148/systemCallTests$ cd numericalTests
bash: cd: numericalTests: No such file or directory
user0511330@ubuntu:/usr/src/linux-4.19.148/systemCallTests$ cd numericalTest
user0511330@ubuntu:/usr/src/linux-4.19.148/systemCallTests/numericalTest$ sudo gedit Makefile
(gedit:7360): IBUS-WARNING **: The owner of /home/user0511330/.config/ibus/bus is

```

[Screenshot 8] Appending the directory to linux Makefile

We also need to tell the compiler where the syscalls we defined.

Append the directory **numericalTest/** to Makefile, which is lying in the kernel folder.

[Command: **\$sudo gedit Makefile]**

[folder: **/linux-4.19.148]**

```
builts-h.a Modules.builts numericalTest.c
Makefile modules.order numericalTest.o
usertest0511330@ubuntu:/usr/src/linux-4.19.148/systemCallTests/numericalTest$ cd ..
usertest0511330@ubuntu:/usr/src/linux-4.19.148/systemCallTests$ cd ..
usertest0511330@ubuntu:/usr/src/linux-4.19.148$ sudo gedit Makefile
[sudo] password for usertest0511330:
(gedit:7330): IBUS-WARNING **: The owner of /home/usertest0511330/.config/ibus/bus is
not root!
(gedit:7330): IBUS-WARNING **: Unable to connect to ibus: Unexpected lack of content
trying to read a line
[FRONT+- prepared
ifeq ($(KBUILD_EXTMOD),
core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ systemCallTests/echoTest/ systemCallTests/
numericalTest/
```

[Screenshot 9]

For calling system calls from a user program in userspace, label our function to the syscalls table **syscalls_64.tbl**.

[Command: **\$sudo gedit syscalls_64.tbl]**

[in folder: **/linux-4.19.148/arch/x86/entry/syscalls]**

```
333 common io_pgetevents      __x64_sys_io_pgetevents
334 common rseq               __x64_sys_rseq
335 common syscalltest_helloworld __x64_sys_syscalltest_helloworld
336 common syscalltest_echo    __x64_sys_syscalltest_echo
337 common syscalltest_returnIndividualValues __x64_sys_syscalltest_returnIndividualValues
338 common syscalltest_addition __x64_sys_syscalltest_addition
339 common syscalltest_multiplication __x64_sys_syscalltest_multiplication
340 common syscalltest_dataTypes   __x64_sys_syscalltest_dataTypes

#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation. The __x32_compat_sys stubs are created
# on-the-fly for compat_sys_*() compatibility system calls if X86_X32
# is defined.
#
512  x32   rt_sigaction      __x32_compat_sys_rt_sigaction
513  x32   rt_sigreturn     sys32_x32_rt_sigreturn
514  x32   ioctl              __x32_compat_sys_ioctl
515  x32   readv             __x32_compat_sys_readv
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

Plain Text ▾ Tab Width: 8 ▾ Ln 34
s3 usertest0511330@ubuntu:/usr/src/linux-4.19.148/systemCallTests/numericalTest$ cd ..
s3 usertest0511330@ubuntu:~$ cd .
s3 usertest0511330@ubuntu:~$ cd /usr/src/linux-4.19.148
usertest0511330@ubuntu:/usr/src/linux-4.19.148$ cd arch/x86/entry/syscalls/
ldusertest0511330@ubuntu:/usr/src/linux-4.19.148/arch/x86/entry/syscalls$ ls
Makefile syscall_32.tbl syscall_64.tbl syscallhdr.sh syscaltbl.sh
usertest0511330@ubuntu:/usr/src/linux-4.19.148/arch/x86/entry/syscalls$ sudo gedit sy
```

[Screenshot 10]

As the previous section mentioned, we also need to add the prototype of the function to **syscalls.h**. “asmlinkage” is a keyword used to indicate that all parameters of the function would be available on the stack.

[Command: **\$sudo gedit syscalls.h**]

[folder: **/linux-4.19.148/include/linux**]

```
sudo: gedit: command not found
[usertest0511330@ubuntu:/usr/src/linux-4.19.148/include/linux]$ sudo gedit syscalls.h

(gedit:7656): IBUS-WARNING **: The owner of /home/usertest0511330/.config/ibus/bus is
not root!

(gedit:7656): IBUS-WARNING **: Unable to connect to ibus: Unexpected lack of content
trying to read a line

(gedit:7656): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBu
s.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .se
rvice files

}

asmlinkage long syscalltest_helloworld(void);
asmlinkage long syscalltest_echo(int);
asmlinkage long syscalltest_returnIndividualValues(int studentId, int a, int b);
asmlinkage long syscalltest_addition(int studentId, int a, int b);
asmlinkage long syscalltest_multiplication(int studentId, int a, int b);
asmlinkage long syscalltest_datatypes(int studentId);

#endif
```

Finally, we can recompile the kernel. run the following commands:

\$sudo su

make menuconfig

make -j \$(nproc)

make modules_install

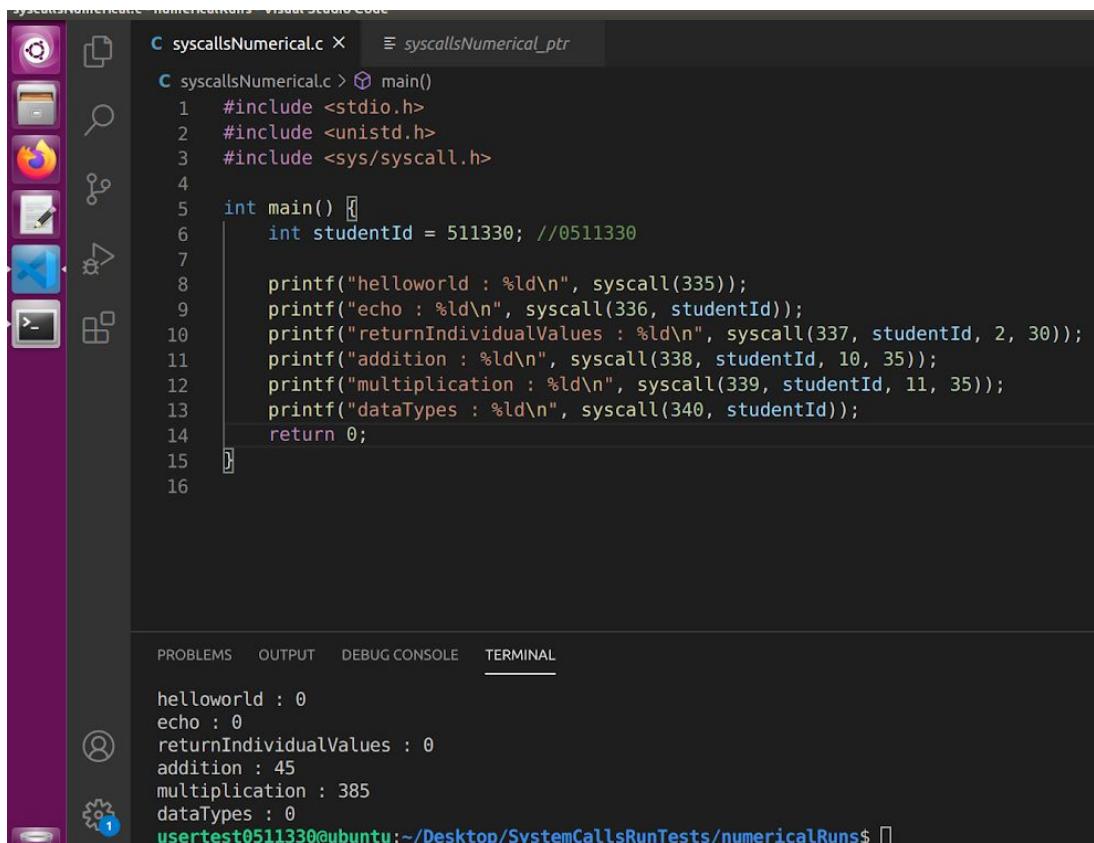
make install

[Screenshot 11] Source code of syscallsNumerical.c

The source code to test syscalls as shown.

[Remind]

When you expect to receive a **negative value return** from syscalls, It's treated as an error and always returns -1 instead of your expectation.

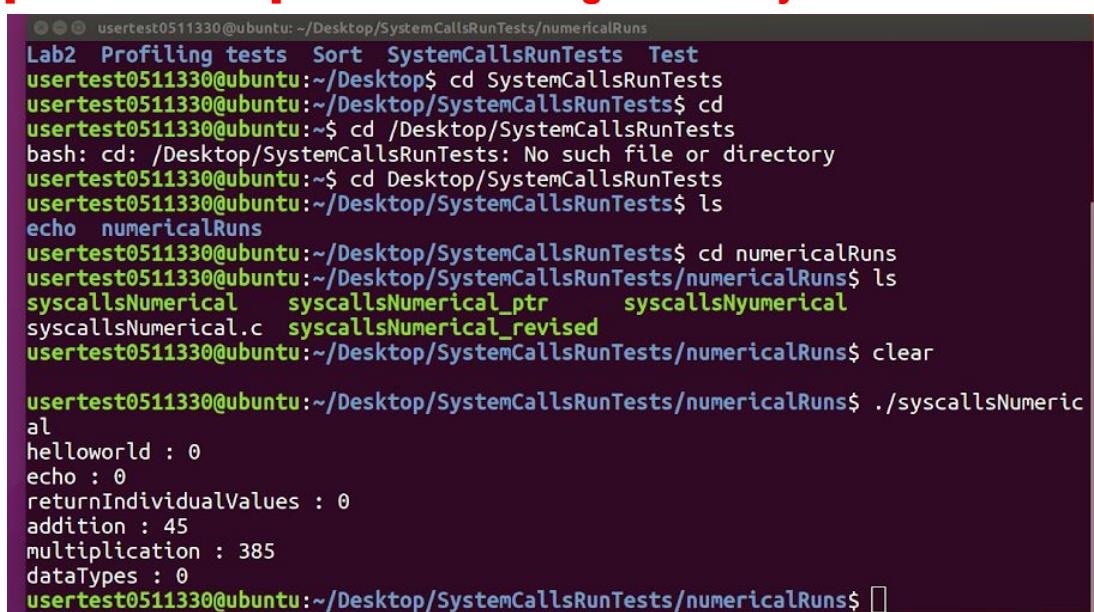


```
syscallsNumerical.c  syscallsNumerical_ptr

C syscallsNumerical.c > main()
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/syscall.h>
4
5 int main() {
6     int studentId = 511330; //0511330
7
8     printf("helloworld : %ld\n", syscall(335));
9     printf("echo : %ld\n", syscall(336, studentId));
10    printf("returnIndividualValues : %ld\n", syscall(337, studentId, 2, 30));
11    printf("addition : %ld\n", syscall(338, studentId, 10, 35));
12    printf("multiplication : %ld\n", syscall(339, studentId, 11, 35));
13    printf("dataTypes : %ld\n", syscall(340, studentId));
14
15 }
16

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
helloworld : 0
echo : 0
returnIndividualValues : 0
addition : 45
multiplication : 385
dataTypes : 0
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$
```

[Screenshot 12] Results of calling custom syscall



```
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns
Lab2 Profiling tests Sort SystemCallsRunTests Test
usertest0511330@ubuntu:~/Desktop$ cd SystemCallsRunTests
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests$ cd
usertest0511330@ubuntu:~$ cd /Desktop/SystemCallsRunTests
bash: cd: /Desktop/SystemCallsRunTests: No such file or directory
usertest0511330@ubuntu:~$ cd Desktop/SystemCallsRunTests
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests$ ls
echo numericalRuns
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests$ cd numericalRuns
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$ ls
syscallsNumerical  syscallsNumerical_ptr  syscallsNyumerical
syscallsNumerical.c  syscallsNumerical_revised
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$ clear

usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$ ./syscallsNumerical
helloworld : 0
echo : 0
returnIndividualValues : 0
addition : 45
multiplication : 385
dataTypes : 0
usertest0511330@ubuntu:~/Desktop/SystemCallsRunTests/numericalRuns$
```

[Screenshot 13] Messages printed on the kernel ring buffer

[Command: **dmesg**] to see the messages printed in the ring buffer.

```
user@user:~/Desktop/SystemCallsRunTests/numericalRuns$ [27338.982462] [511330] syscalltest_multiplication : 11, 35, 385  
[27338.982463] [511330] Size of unsinged int : 4 bytes.  
[27338.982464] [511330] Size of singed int : 4 bytes.  
[27338.982464] [511330] Size of unsigned long : 8 bytes.  
[27338.982464] [511330] Size of signed long : 8 bytes.  
[27338.982464] [511330] Size of unsigned long long : 8 bytes.  
[27338.982465] [511330] Size of signed long long : 8 bytes.  
[27338.982465] [511330] Size of double : 8 bytes.  
[27338.982465] [511330] Size of char : 1 bytes.  
[27360.676225] Hello world from a system call! OS_Project02!  
[27360.676271] My student id is : [511330]  
[27360.676273] [511330] syscalltest_returnIndividualValues : 2, 30  
[27360.676274] [511330] syscalltest_addition : 10, 35, 45  
[27360.676275] [511330] syscalltest_multiplication : 11, 35, 385  
[27360.676276] [511330] Size of unsinged int : 4 bytes.  
[27360.676276] [511330] Size of singed int : 4 bytes.  
[27360.676276] [511330] Size of unsigned long : 8 bytes.  
[27360.676276] [511330] Size of signed long : 8 bytes.  
[27360.676277] [511330] Size of unsigned long long : 8 bytes.  
[27360.676277] [511330] Size of signed long long : 8 bytes.  
[27360.676277] [511330] Size of double : 8 bytes.  
[27360.676277] [511330] Size of char : 1 bytes.  
user@user:~/Desktop/SystemCallsRunTests/numericalRuns$
```

[Questions]

1. What is Kernel space and User space? What's the difference between them?

(i) **Kernel Space:** Kernel space is strictly reserved for running a privileged instruction of the system kernel, kernel extensions, and device drivers.

(ii) **User Space:** User space is the memory area where application software and some drivers execute.

The separation serves to provide memory protection mechanism of OS such as memory protection and hardware protection from malicious or errant software behaviour. The user processes can only access a small part of the kernel via interface - **the system calls**

2. What are protection rings and how many? What are Ring 0 and Ring 1?

Protection ring is the mechanism to protect data and functionality from faults or malicious behavior. It's mainly to implement the idea of hierarchical protection domains. Typically in x86 architecture, there are 4 rings of protection.

- Ring 0 is kernel mode, the level with the most privileges and interacts most directly with hardware. (The most privileged one)
- Ring 1 may be used by virtual machine hypervisors or drivers.

3. What is a system call? How many are there? What are the differences?

The interface between kernel space and user space. When a user process needs to access the kernel, a system call will be called.

There are 547 system calls in linux version 4.19.148. Some are used to handle process interrupts, some serve user process the I/O requests, or deal with messages passing between processes.

4. For the custom kernel built, where is the list of system calls.

File name: syscall_64.tbl

Directory: /usr/src/linux-4.19.148/arch/x86/entry/syscalls/syscall_64.tbl

5. What is the system call ID?

System call ID is the unique number to each system call. All system calls will be placed in a system call table. When invoking a system call, the OS will find the system call in the table by system call ID, treated as **array index** to speed up.

6. (i) What do the reserved words “asmlinkage” and “printk” mean?

asmlinkage is a keyword to indicate all parameters of function would be available on the stack instead of passed by register.

(ii) What does the command **update-initramfs** do?

It will generate an initramfs image, managing your initramfs image on your local box

7. How do you use printk and read the messages printed by it?

Simply include the <linux/kernel.h> header to use it. We can use the command “dmesg” to see the logs which were printed in the kernel ring buffer.

8. What is the kernel ring buffer? How do you read its contents?

The kernel ring buffer is a data structure that records messages related to the operation of the kernel.

dmesg will output all kernel messages currently in the ring buffer to a file called "kernel_msgs.txt".

9. What is a function signature?

A function signature defines input and output of functions, including: parameters & their datatype, return value & their datatype, and other advanced information (e.g. **static**).

10. What does SYSCALL_DEFINE[n] mean? What is n?

It means that there will be **n** parameters in self-defined system call.

11. For a system call wrapper, how does its function signature look like when it has 0 inputs as parameters? 1, 2 or 3 integer numbers as input?

0 input : **SYSCALL_DEFINE0**(*syscall name*)

1 input : **SYSCALL_DEFINE1**(*syscall name, parameter 1*)

n inputs: **SYSCALL_DEFINEn**(*syscall name, parameter 1, ..., parameter n*)

(n<6)

12. Does the function signature of a SYSCALL_DEFINE change depending on the type of element returned?

No, the function signature of **SYSCALL_DEFINE** is independent of their return type.

13. What is #include<linux/kernel.h> and #include<linux/syscalls.h>?

<**linux/kernel.h**> is a header which gets used for module builds. It's part of the kernel source and intended for user processes.

<**linux/syscalls.h**> includes the system calls which are exposed to userspace.