

Reading Material for System Call Implementation in Linux OS [Including Kernel Recompile]

**As Part of
Operating Systems [CS F372] Course
Semester I, 2023 – 2024**



BITS Pilani
K K Birla Goa Campus

Please contact: [only if you fail miserably!!]

Saket B {f20200983@goa.bits-pilani.ac.in}

Rahul M Biju {f20200953@goa.bits-pilani.ac.in}

Ashmit Khandelwal {f20200980@goa.bits-pilani.ac.in}

S Ravi Sanker {f20200142@goa.bits-pilani.ac.in}

Harsh Gujarathi {f20201712@goa.bits-pilani.ac.in}

Akhil Bansal {f20200116@goa.bits-pilani.ac.in}

Manank Patel {f20201696@goa.bits-pilani.ac.in}

**BIRLA INSTITUTE OF TECHNOLOGY AND
SCIENCE, PILANI –K K BIRLA GOA CAMPUS**

INDEX

Sec. No.	Section Title	Page No.
1.	Basic Preparation for Recompilation	3
2.	Recompilation of Linux Kernel with / without Modification(s)	4
3.	Implementation of New System Call [New System Call to Add Two Positive Integers]	7
4.	Implementation of User Space Programs	13
5.	Practice Problem	14

Important Note: Please **do not try this directly on your laptop**, you may end up crashing your system/corrupting several files. We recommend installing Virtual Box and setting up a Linux VM on it. You can find instructions on how to do that here: <https://itsfoss.com/install-linux-in-virtualbox/>. Proceed with the next steps only after you a Ubuntu VM up and running in Virtual Box.

Basic Preparation for Recompilation

Step #1: Download the kernel source 5.19.8 from <https://www.kernel.org>

Step #2: Open a terminal and login to super-user by

```
$ sudo su
<Enter root password here>
```

Step #3: Place the tar.xz file in /usr/src/ directory

Step #4: Set the present working directory as /usr/src/ by

```
$ cd /usr/src/
```

Step #5: Untar the *linux-5.19.8.tar.xz* file by

```
$ tar -xvf linux-5.19.8.tar.xz
```

Step #6: Set the present working directory as linux-5.19.8

```
$ cd linux-5.19.8/
```

Recompilation of Linux Kernel with / without Modification(s)

Step #1: Reconfiguration of the Kernel

The Linux Kernel is extraordinarily configurable; you can enable and disable many of its features, as well as set build parameters. Some of the widely used options are: menuconfig, xconfig, gconfig, oldconfig, defconfig etc.

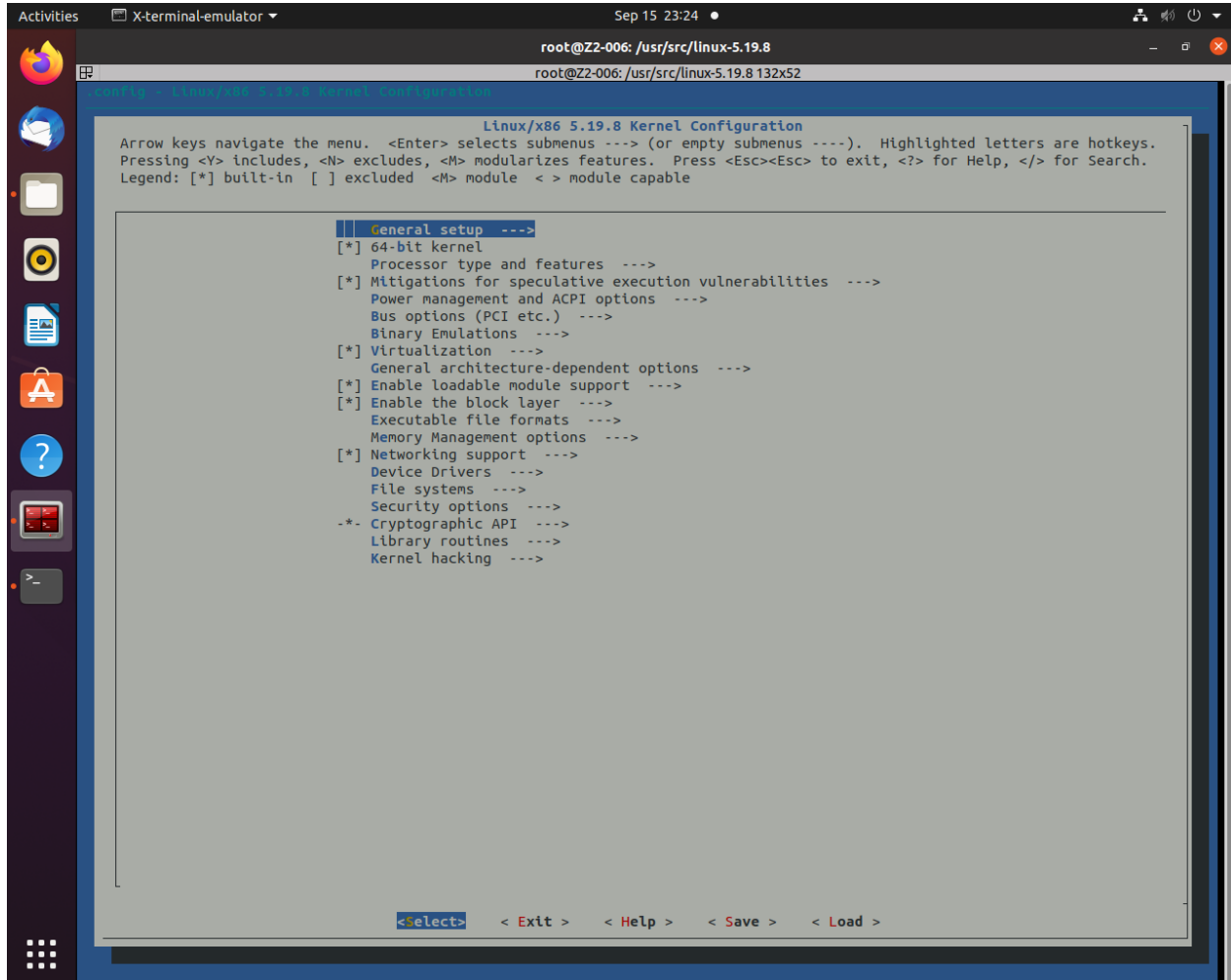
Dependencies you may require to install: flex, bison, libssl-dev, libelf-dev

\$apt install flex bison libssl-dev libelf-dev or

\$apt-get install <package name> E.g.: \$apt-get install flex

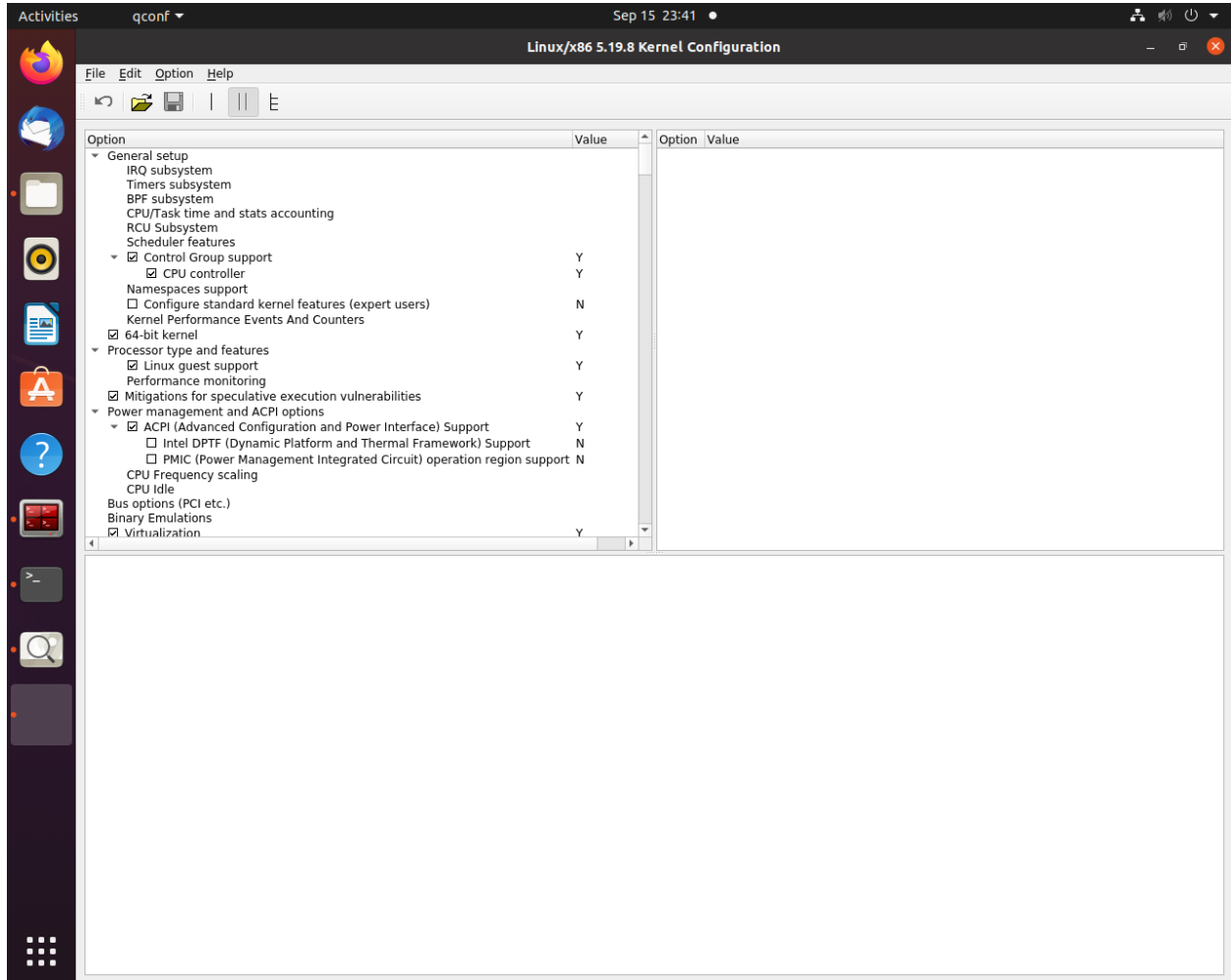
\$ make menuconfig

<Text based color menus, radio lists & dialogs. This option is also useful on remote server if you want to compile kernel remotely.>



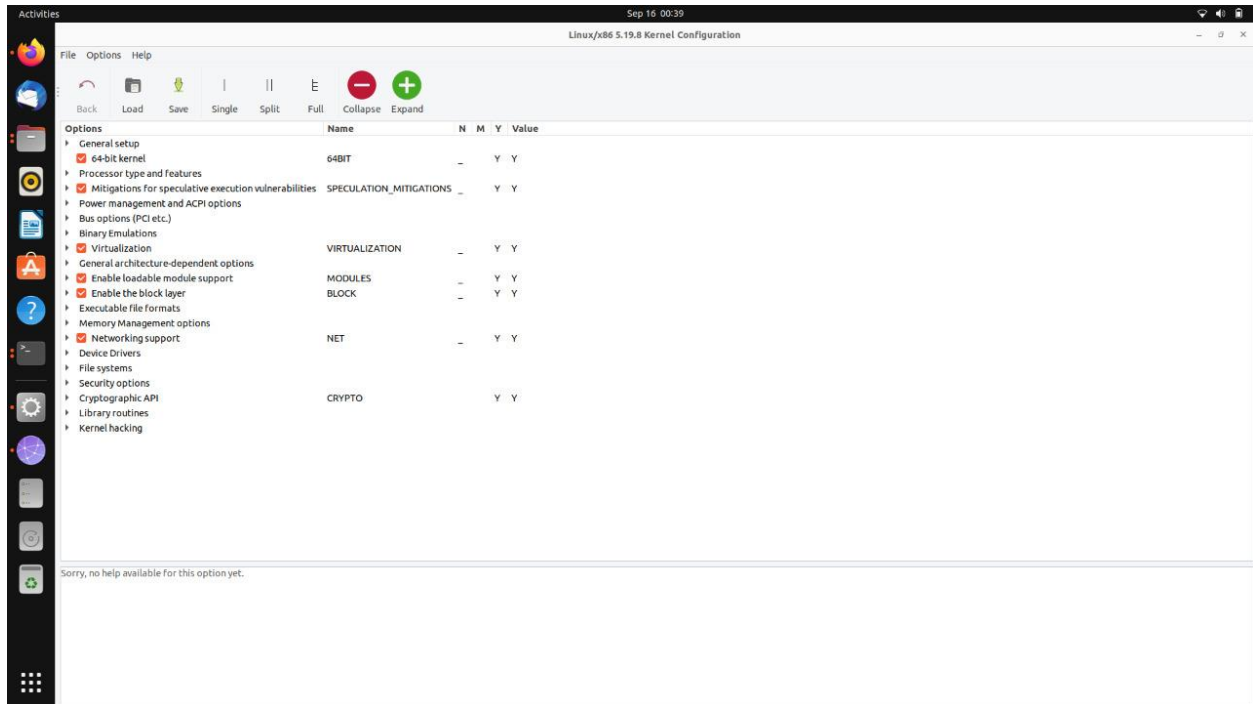
```
$ make xconfig
```

<X windows (Qt) based configuration tool, works best under KDE Desktop.>



\$ make gconfig

<X windows (Gtk) based configuration tool, works best under Gnome Desktop.>



\$ make oldconfig

<Reads the existing config file and prompts the user options in the current kernel source that are not found in the file>

```
root@Z2-006:/usr/src/linux-5.19.8# make oldconfig
HOSTCC  scripts/kconfig/conf.o
HOSTLD  scripts/kconfig/conf
#
# No change to .config
#
root@Z2-006:/usr/src/linux-5.19.8#
```

\$ make defconfig [Use this for reconfiguration option for this assignment]

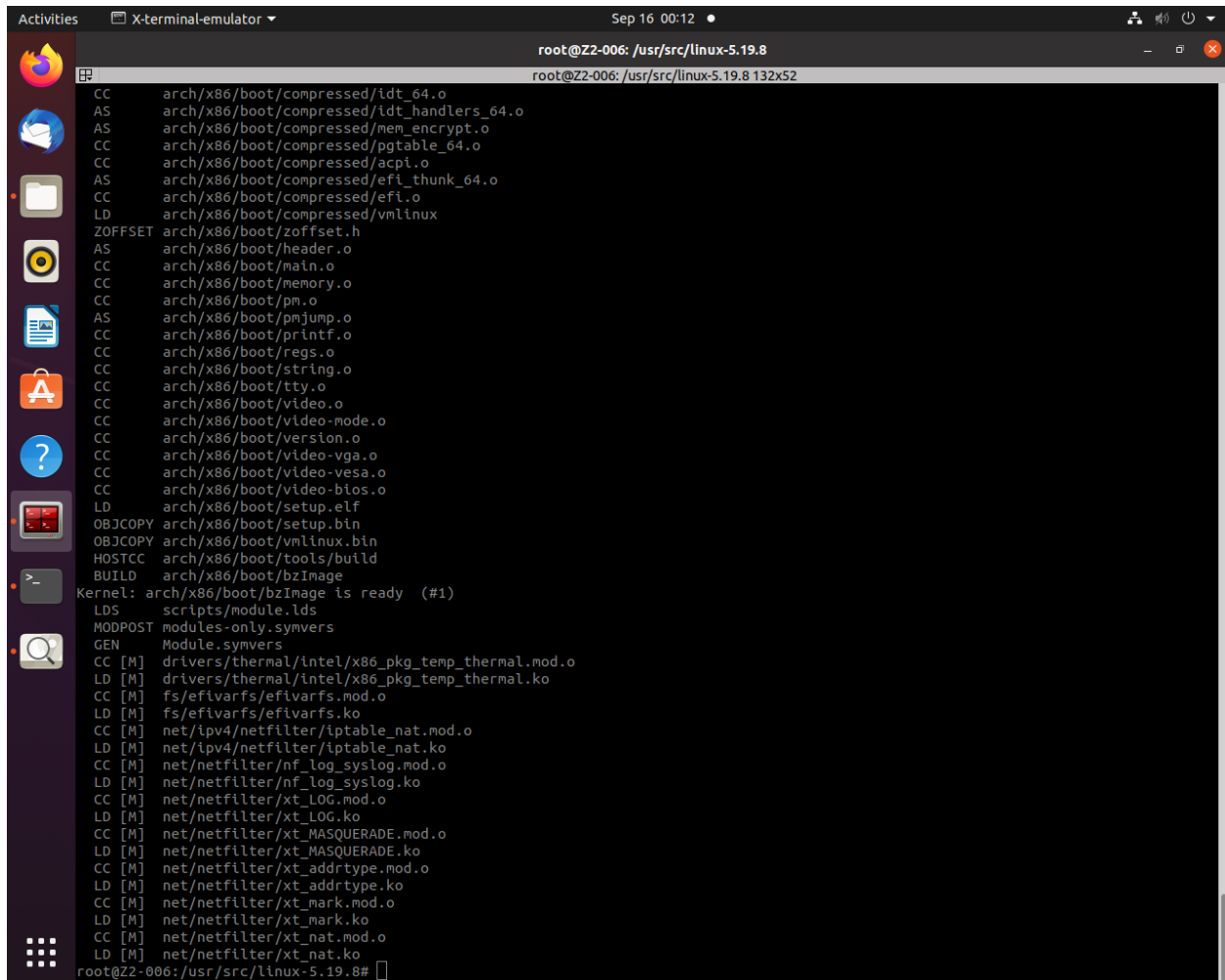
<Creates a default config file for the kernel delineating all the necessary modules to be installed into the kernel>

```
#
root@Z2-006:/usr/src/linux-5.19.8# make defconfig
*** Default configuration is based on 'x86_64_defconfig'
#
# configuration written to .config
#
root@Z2-006:/usr/src/linux-5.19.8#
```

Step #2: Preliminary Recompilation of the Kernel

Execute make to compile the kernel.

\$ make



```
root@Z2-006: /usr/src/linux-5.19.8
root@Z2-006: /usr/src/linux-5.19.8 132x52

CC      arch/x86/boot/compressed/ldt_64.o
AS      arch/x86/boot/compressed/ldt_handlers_64.o
AS      arch/x86/boot/compressed/mem_encrypt.o
CC      arch/x86/boot/compressed/pgtable_64.o
CC      arch/x86/boot/compressed/acpi.o
AS      arch/x86/boot/compressed/efi_thunk_64.o
CC      arch/x86/boot/compressed/efi.o
LD      arch/x86/boot/compressed/vmlinux
ZOFFSET arch/x86/boot/zoffset.h
AS      arch/x86/boot/header.o
CC      arch/x86/boot/main.o
CC      arch/x86/boot/memory.o
CC      arch/x86/boot/pm.o
AS      arch/x86/boot/pmjump.o
CC      arch/x86/boot/printf.o
CC      arch/x86/boot/regs.o
CC      arch/x86/boot/string.o
CC      arch/x86/boot/tty.o
CC      arch/x86/boot/video.o
CC      arch/x86/boot/video-mode.o
CC      arch/x86/boot/version.o
CC      arch/x86/boot/video-vga.o
CC      arch/x86/boot/video-vesa.o
CC      arch/x86/boot/video-bios.o
LD      arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
OBJCOPY arch/x86/boot/vmlinux.bin
HOSTCC  arch/x86/boot/tools/build
BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready (#1)
LDS     scripts/module.lds
MODPOST modules-only.symvers
GEN     Module.symvers
CC [M]  drivers/thermal/intel/x86_pkg_temp_thermal.mod.o
LD [M]  drivers/thermal/intel/x86_pkg_temp_thermal.ko
CC [M]  fs/efivarfs/efivarfs.mod.o
LD [M]  fs/efivarfs/efivarfs.ko
CC [M]  net/ipv4/netfilter/iptable_nat.mod.o
LD [M]  net/ipv4/netfilter/iptable_nat.ko
CC [M]  net/netfilter/nf_log_syslog.mod.o
LD [M]  net/netfilter/nf_log_syslog.ko
CC [M]  net/netfilter/xt_LOG.mod.o
LD [M]  net/netfilter/xt_LOG.ko
CC [M]  net/netfilter/xt_MASQUERADE.mod.o
LD [M]  net/netfilter/xt_MASQUERADE.ko
CC [M]  net/netfilter/xt_addrtype.mod.o
LD [M]  net/netfilter/xt_addrtype.ko
CC [M]  net/netfilter/xt_mark.mod.o
LD [M]  net/netfilter/xt_mark.ko
CC [M]  net/netfilter/xt_nat.mod.o
LD [M]  net/netfilter/xt_nat.ko
root@Z2-006: /usr/src/linux-5.19.8#
```

Step #3: Recompilation of the Kernel module, update initramfs and grub

Execute make modules_install & make modules_install install to compile the modules and update the initramfs and grup.

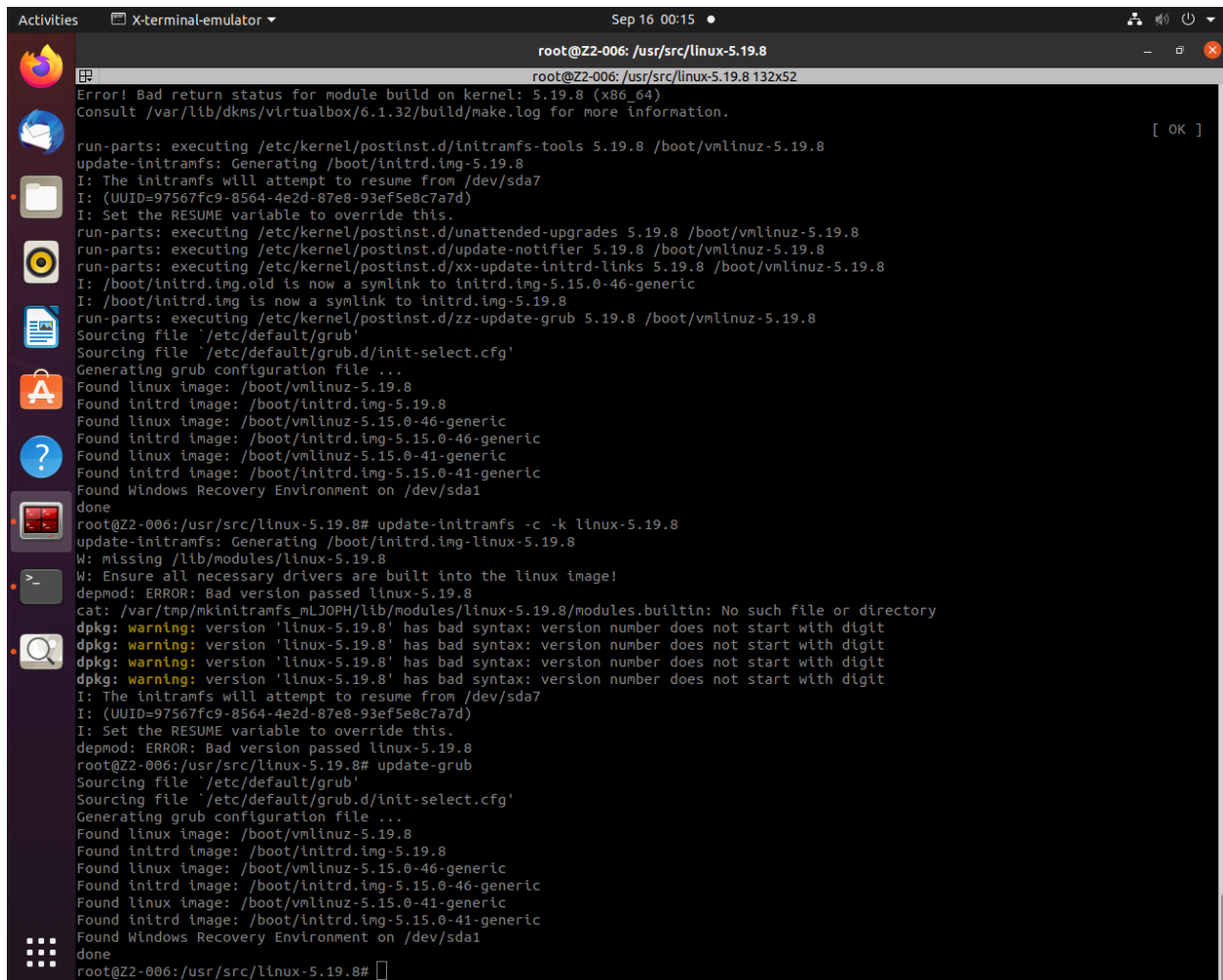
\$ make modules_install && make modules_install install

```
root@Z2-006:/usr/src/linux-5.19.8# make modules_install
INSTALL /lib/modules/5.19.8/kernel/drivers/thermal/intel/x86_pkg_temp_thermal.ko
INSTALL /lib/modules/5.19.8/kernel/fs/efivarfs/efivarfs.ko
INSTALL /lib/modules/5.19.8/kernel/net/ipv4/netfilter/iptable_nat.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/nf_log_syslog.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_LOG.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_MASQUERADE.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_addrtype.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_mark.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_nat.ko
DEPMOD /lib/modules/5.19.8
root@Z2-006:/usr/src/linux-5.19.8#
```

```
root@Z2-006:/usr/src/linux-5.19.8# make modules_install install
INSTALL /lib/modules/5.19.8/kernel/drivers/thermal/intel/x86_pkg_temp_thermal.ko
INSTALL /lib/modules/5.19.8/kernel/fs/efivarfs/efivarfs.ko
INSTALL /lib/modules/5.19.8/kernel/net/ipv4/netfilter/iptable_nat.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/nf_log_syslog.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_LOG.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_MASQUERADE.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_addrtype.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_mark.ko
INSTALL /lib/modules/5.19.8/kernel/net/netfilter/xt_nat.ko
DEPMOD /lib/modules/5.19.8
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/dkms 5.19.8 /boot/vmlinuz-5.19.8
* dkms: running auto installation service for kernel 5.19.8
Kernel preparation unnecessary for this kernel. Skipping...
Building module:
cleaning build area...
make -j8 KERNELRELEASE=5.19.8 -C /lib/modules/5.19.8/build M=/var/lib/dkms/virtualbox/6.1.32/build.....(bad exit status: 2)
ERROR (dkms apport): kernel package linux-headers-5.19.8 is not supported
Error! Bad return status for module build on kernel: 5.19.8 (x86_64)
Consult /var/lib/dkms/virtualbox/6.1.32/build/make.log for more information.
[ OK ]
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.19.8 /boot/vmlinuz-5.19.8
update-initramfs: Generating /boot/initrd.img-5.19.8
I: The initramfs will attempt to resume from /dev/sda7
I: (UUID=97567fc9-8564-4e2d-87e8-93ef5e8c7a7d)
I: Set the RESUME variable to override this.
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.19.8 /boot/vmlinuz-5.19.8
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.19.8 /boot/vmlinuz-5.19.8
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.19.8 /boot/vmlinuz-5.19.8
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-46-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.19.8
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.19.8 /boot/vmlinuz-5.19.8
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.19.8
Found initrd image: /boot/initrd.img-5.19.8
Found linux image: /boot/vmlinuz-5.15.0-46-generic
Found initrd image: /boot/initrd.img-5.15.0-46-generic
Found linux image: /boot/vmlinuz-5.15.0-41-generic
Found initrd image: /boot/initrd.img-5.15.0-41-generic
Found Windows Recovery Environment on /dev/sda1
done
root@Z2-006:/usr/src/linux-5.19.8#
```

In addition to installing the bzImage it even runs the following commands

```
update-initramfs -c -k linux-5.19.8
update-grub
```

```
root@Z2-006: /usr/src/linux-5.19.8
root@Z2-006: /usr/src/linux-5.19.8 132x52
Error! Bad return status for module build on kernel: 5.19.8 (x86_64)
Consult /var/lib/dkms/virtualbox/6.1.32/build/make.log for more information.
[ OK ]
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.19.8 /boot/vmlinuz-5.19.8
update-initramfs: Generating /boot/initrd.img-5.19.8
I: The initramfs will attempt to resume from /dev/sda7
I: (UUID=97567fc9-8564-4e2d-87e8-93ef5e8c7a7d)
I: Set the RESUME variable to override this.
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.19.8 /boot/vmlinuz-5.19.8
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.19.8 /boot/vmlinuz-5.19.8
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.19.8 /boot/vmlinuz-5.19.8
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-46-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.19.8
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.19.8 /boot/vmlinuz-5.19.8
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.19.8
Found initrd image: /boot/initrd.img-5.19.8
Found linux image: /boot/vmlinuz-5.15.0-46-generic
Found initrd image: /boot/initrd.img-5.15.0-46-generic
Found linux image: /boot/vmlinuz-5.15.0-41-generic
Found initrd image: /boot/initrd.img-5.15.0-41-generic
Found Windows Recovery Environment on /dev/sda1
done
root@Z2-006: /usr/src/linux-5.19.8# update-initramfs -c -k linux-5.19.8
update-initramfs: Generating /boot/initrd.img-linux-5.19.8
W: missing /lib/modules/linux-5.19.8
W: Ensure all necessary drivers are built into the linux image!
depmod: ERROR: Bad version passed linux-5.19.8
cat: /var/tmp/mkinitramfs_mLJOPH/lib/modules/linux-5.19.8/modules.builtin: No such file or directory
dpkg: warning: version 'linux-5.19.8' has bad syntax: version number does not start with digit
dpkg: warning: version 'linux-5.19.8' has bad syntax: version number does not start with digit
dpkg: warning: version 'linux-5.19.8' has bad syntax: version number does not start with digit
dpkg: warning: version 'linux-5.19.8' has bad syntax: version number does not start with digit
I: The initramfs will attempt to resume from /dev/sda7
I: (UUID=97567fc9-8564-4e2d-87e8-93ef5e8c7a7d)
I: Set the RESUME variable to override this.
depmod: ERROR: Bad version passed linux-5.19.8
root@Z2-006: /usr/src/linux-5.19.8# update-grub
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.19.8
Found initrd image: /boot/initrd.img-5.19.8
Found linux image: /boot/vmlinuz-5.15.0-46-generic
Found initrd image: /boot/initrd.img-5.15.0-46-generic
Found linux image: /boot/vmlinuz-5.15.0-41-generic
Found initrd image: /boot/initrd.img-5.15.0-41-generic
Found Windows Recovery Environment on /dev/sda1
done
root@Z2-006: /usr/src/linux-5.19.8#
```

Now that the kernel has been recompiled, reboot the system and boot into this kernel from the grub <Select advanced ubuntu tab followed by the New kernel>

Implementation of New System Call [New System Call to Add 2 Positive Integers]

Step #1: Create a directory under /usr/src/linux-5.19.8/

Create a directory named add_syscall under /usr/src/linux-5.19.8/

```
$ mkdir add_syscall
```

Step #2: Create the following files under add_syscall directory

1. add_syscall.c
2. add_syscall.h
3. Makefile

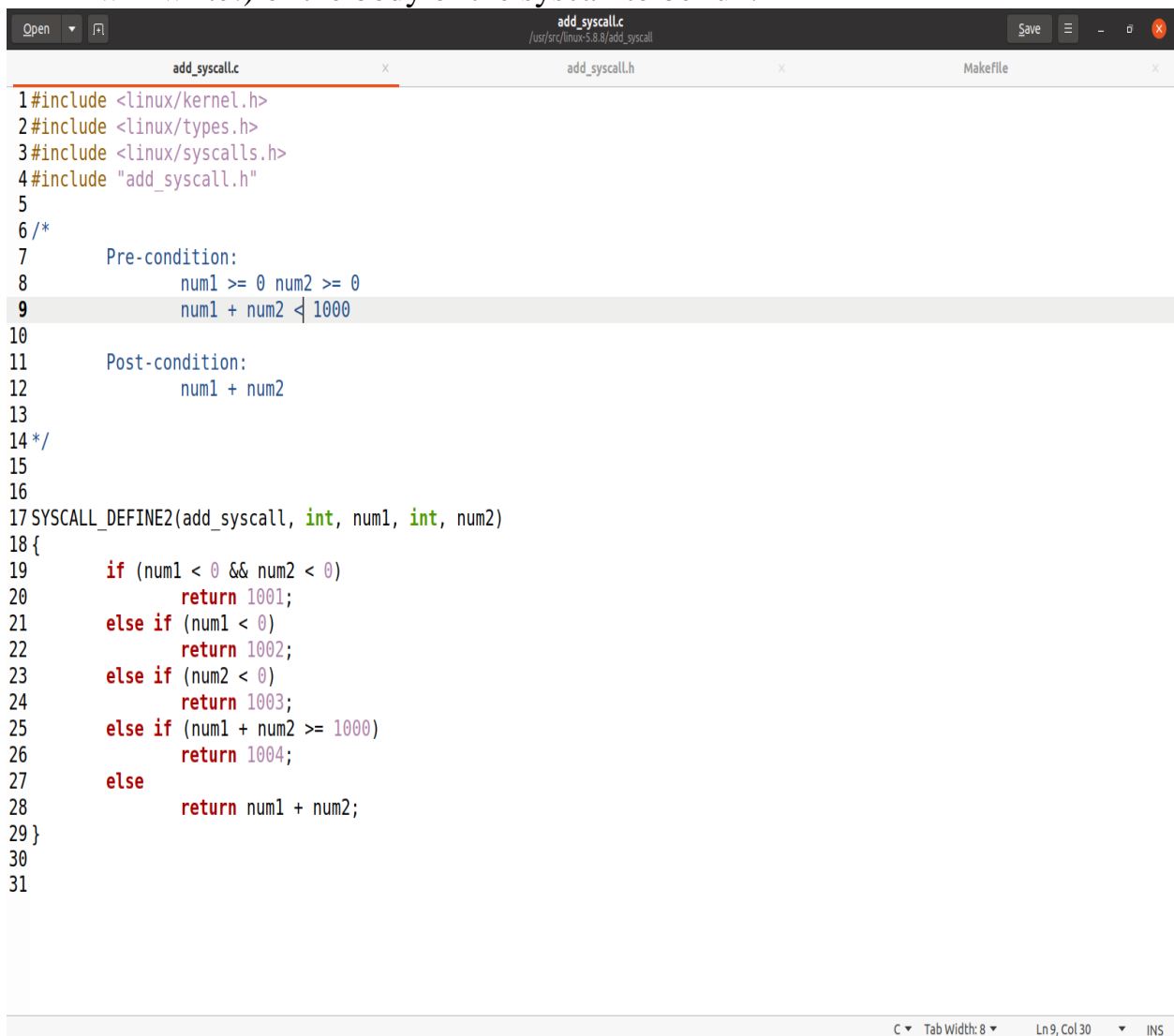
Contents of **add_syscall.c**

SYSCALL_DEFINE*n*() macros are the standard way for kernel code to define a system call, where the *n* suffix indicates the argument count.

The first argument to the macro is the name of the system call (without **sys_** prepended to it). The remaining arguments are pairs of type and name for the parameters.

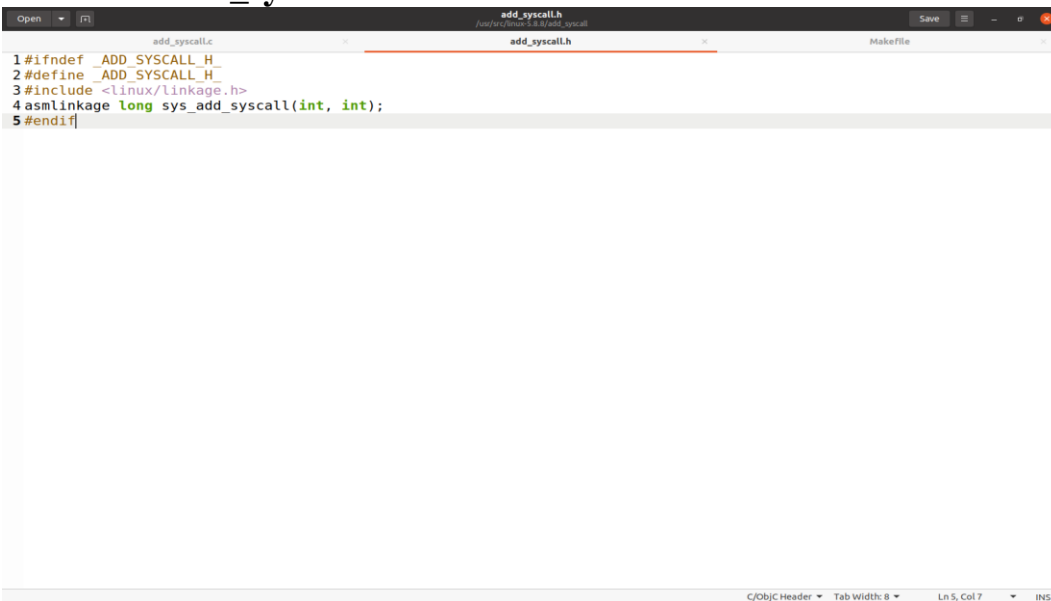
The definitions of these **SYSCALL_DEFINE**... macros are in **#include <linux/syscalls.h>**. Hence, the .c file in which you code the body of your syscall's service routine must **#include <linux/syscalls.h>**

It has within { ... } (after your **SYSCALL_DEFINE**...(...)) the code (you will write!) of the body of the syscall to be run.



```
1#include <linux/kernel.h>
2#include <linux/types.h>
3#include <linux/syscalls.h>
4#include "add_syscall.h"
5
6/*
7    Pre-condition:
8        num1 >= 0 num2 >= 0
9        num1 + num2 < 1000
10
11    Post-condition:
12        num1 + num2
13*/
14
15
16
17SYSCALL_DEFINE2(add_syscall, int, num1, int, num2)
18{
19    if (num1 < 0 && num2 < 0)
20        return 1001;
21    else if (num1 < 0)
22        return 1002;
23    else if (num2 < 0)
24        return 1003;
25    else if (num1 + num2 >= 1000)
26        return 1004;
27    else
28        return num1 + num2;
29}
30
31
```

Content of `add_syscall.h`

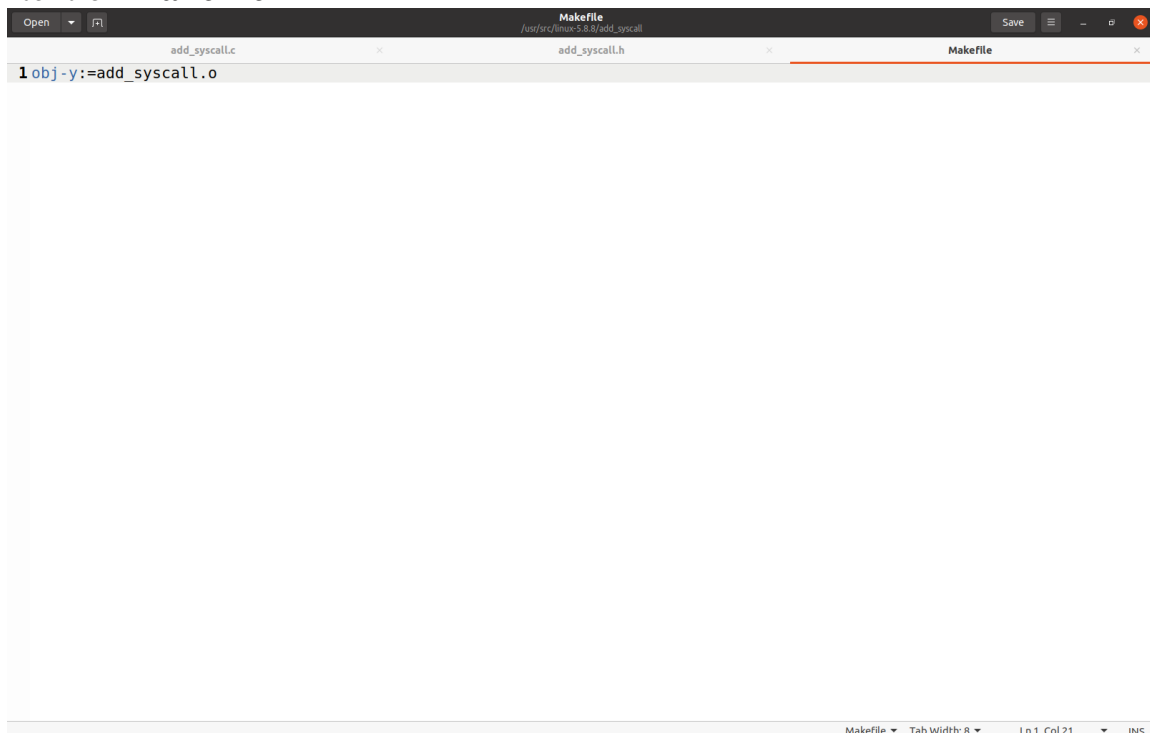


The screenshot shows a code editor with three tabs: `add_syscall.c`, `add_syscall.h` (active), and `Makefile`. The `add_syscall.h` file contains the following code:

```
1 #ifndef _ADD_SYSCALL_H_
2 #define _ADD_SYSCALL_H_
3 #include <linux/linkage.h>
4 asmlinkage long sys_add_syscall(int, int);
5 #endif
```

The status bar at the bottom indicates the current file is `add_syscall.h`, with the cursor at line 5, column 7.

Content of `Makefile`



The screenshot shows a code editor with three tabs: `add_syscall.c`, `add_syscall.h`, and `Makefile` (active). The `Makefile` file contains the following code:

```
1 obj-y:=add_syscall.o
```

The status bar at the bottom indicates the current file is `Makefile`, with the cursor at line 1, column 21.

Step #3: Modify the following files

1. `/usr/src/linux-5.19.8/Makefile`
2. `/usr/src/linux-5.19.8/arch/x86/entry/syscalls/syscall_64.tbl`
3. `/usr/src/linux-5.19.8/include/asm-generic/syscalls.h`
4. `/usr/src/linux-5.19.8/include/linux/syscalls.h`

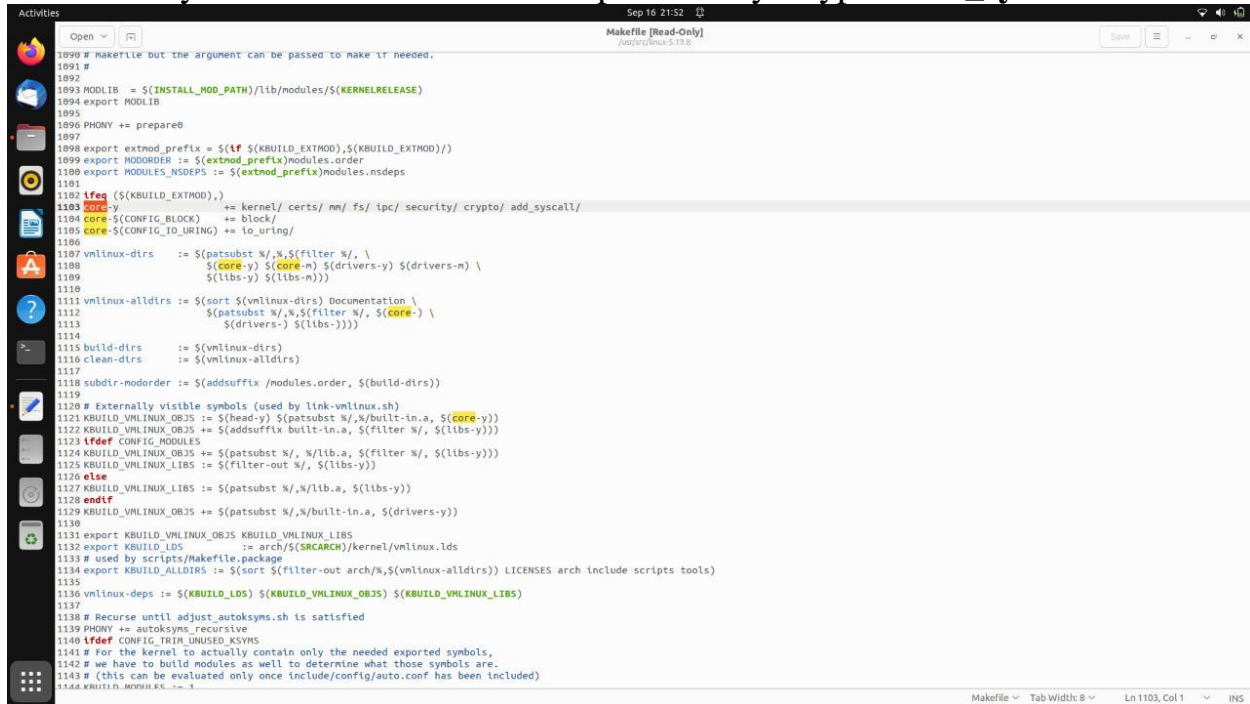
3.1: Modify /usr/src/linux-5.19.8/Makefile:

<Update the following line in Makefile>

core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/

<to the following by adding add_syscall/ in the end>

core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ **add_syscall/**



```
1090 # Makefile but the argument can be passed to make it needed.
1091 #
1092
1093 MODLIB := $(INSTALL_MOD_PATH)/lib/modules/$(KERNELRELEASE)
1094 export MODLIB
1095
1096 PHONY += prepare0
1097
1098 export extmod_prefix := $(if $(KBUILD_EXTMOD),$(KBUILD_EXTMOD)/)
1099 export MODORDER := $(extmod_prefix)modules.order
1100 export MODULES_NSDEPS := $(extmod_prefix)modules.nsdeps
1101
1102 lflags := $(KBUILD_EXTMOD),
1103 core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ add_syscall/
1104 core-$(CONFIG_BLOCK) += block/
1105 core-$(CONFIG_IO_URING) += io_uring/
1106
1107 vmlinux-dirs := $(patsubst %,%, $(filter %, \
1108     $(core-y) $(core-m) $(drivers-y) $(drivers-m) \
1109     $(libs-y) $(libs-m)))
1110
1111 vmlinux-all-dirs := $(sort $(vmlinux-dirs) Documentation \
1112     $(patsubst %,%, $(filter %, $(core-) \
1113     $(drivers-) $(libs-))))
1114
1115 build-dirs := $(vmlinux-dirs)
1116 clean-dirs := $(vmlinux-all-dirs)
1117
1118 subdir-morder := $(addsuffix /modules.order, $(build-dirs))
1119
1120 # Externally visible symbols (used by link-vmlinux.sh)
1121 KBUILD_VMLINUX_OBJS := $(head-y) $(patsubst %,%, $(built-in.a, $(core-y)))
1122 KBUILD_VMLINUX_OBJS += $(addsuffix built-in.a, $(filter %, $(libs-y)))
1123 ldef CONFIG_MODULES
1124 KBUILD_VMLINUX_OBJS += $(patsubst %, %, $(lib.a, $(filter %, $(libs-y)))
1125 KBUILD_VMLINUX_LIBS := $(filter-out %, $(libs-y))
1126 else
1127 KBUILD_VMLINUX_LIBS := $(patsubst %, %, $(lib.a, $(libs-y))
1128 endif
1129 KBUILD_VMLINUX_OBJS += $(patsubst %, %, $(built-in.a, $(drivers-y)))
1130
1131 export KBUILD_VMLINUX_OBJS KBUILD_VMLINUX_LIBS
1132 export KBUILD_LDS := arch/$(SRCARCH)/kernel/vmlinux.lds
1133 # used by scripts/Makefile.package
1134 export KBUILD_ALLDIRS := $(sort $(filter-out arch/%, $(vmlinux-all-dirs)) LICENSES arch include scripts tools)
1135
1136 vmlinux-deps := $(KBUILD_LDS) $(KBUILD_VMLINUX_OBJS) $(KBUILD_VMLINUX_LIBS)
1137
1138 # Recurse until adjust_autoksyms.sh is satisfied
1139 PHONY += autoksyms_recursive
1140 ldef CONFIG_TRIM_UNUSED_KSYMS
1141 # For the kernel to actually contain only the needed exported symbols,
1142 # we have to build modules as well to determine what those symbols are.
1143 # (this can be evaluated only once include/config/auto.conf has been included)
1144 KBUILD_N_MODULES_KC := 1
```

3.2: Modify /usr/src/linux-5.19.8/arch/x86/entry/syscalls/syscall_64.tbl:

Update the file: /arch/x86/entry/syscalls/syscall_64.tbl to add the new syscall at the next available system call number in the common list of syscalls like:

451 common add_syscall sys_add_syscall

Here sys_add_syscall is the entry point for the system call add_syscall and it will be common across the x86-{64, 32} bit architectures.

Sep 16 21:53

syscalls_64.tbl [Read-Only]
/usr/src/linux-5.19.8/arch/x86/entry/syscalls

syscalls_64.tbl

342	331	common	pkey_rtee	sys_pkey_rtee
343	332	common	statx	sys_statx
344	333	common	io_pgetevents	sys_io_pgetevents
345	334	common	rseq	sys_rseq
346 # don't use numbers 387 through 423, add new calls after the last				
347 # 'common' entry				
348	424	common	pidfd_send_signal	sys_pidfd_send_signal
349	425	common	io_uring_setup	sys_io_uring_setup
350	426	common	io_uring_enter	sys_io_uring_enter
351	427	common	io_uring_register	sys_io_uring_register
352	428	common	open_tree	sys_open_tree
353	429	common	move_mount	sys_move_mount
354	430	common	fsopen	sys_fsopen
355	431	common	fsconfig	sys_fsconfig
356	432	common	fsmount	sys_fsmount
357	433	common	fspick	sys_fspick
358	434	common	pidfd_open	sys_pidfd_open
359	435	common	clone3	sys_clone3
360	436	common	close_range	sys_close_range
361	437	common	openat2	sys_openat2
362	438	common	pidfd_getfd	sys_pidfd_getfd
363	439	common	faccessat2	sys_faccessat2
364	440	common	process_nadvice	sys_process_nadvice
365	441	common	epoll_pwait2	sys_epoll_pwait2
366	442	common	mount_setattr	sys_mount_setattr
367	443	common	quotactl_fd	sys_quotactl_fd
368	444	common	landlock_create_ruleset	sys_landlock_create_ruleset
369	445	common	landlock_add_rule	sys_landlock_add_rule
370	446	common	landlock_restrict_self	sys_landlock_restrict_self
371	447	common	memfd_secret	sys_memfd_secret
372	448	common	process_mrelease	sys_process_mrelease
373	449	common	futex_waitv	sys_futex_waitv
374	450	common	set_nepolicy_home_node	sys_set_nepolicy_home_node
375	451	common	add_syscall	sys_add_syscall
376 #				
377 # Due to a historical design error, certain syscalls are numbered differently				
378 # in x32 as compared to native x86_64. These syscalls have numbers 512-547.				
379 # Do not add new syscalls to this range. Numbers 548 and above are available				
380 # for non-x32 use.				
381 #				
382	512	x32	rt_sigaction	compat_sys_rt_sigaction
383	513	x32	rt_sigreturn	compat_sys_x32_rt_sigreturn
384	514	x32	ioctl	compat_sys_ioctl
385	515	x32	readv	sys_readv
386	516	x32	writew	sys_writew
387	517	x32	recvfrom	compat_sys_recvfrom
388	518	x32	sendmsg	compat_sys_sendmsg
389	519	x32	recvmsg	compat_sys_recvmsg
390	520	x32	execve	compat_sys_execve
391	521	x32	ptrace	compat_sys_ptrace
392	522	x32	rt_sigpending	compat_sys_rt_sigpending
393	523	x32	rt_sigtimedwait	compat_sys_rt_sigtimedwait_time64
394	524	x32	rt_clone3waitinfo	compat_sys_rt_clone3waitinfo

Plain Text Tab Width: 8 Ln 375, Col 56 INS

This table is read by scripts and used to generate some of the boilerplate code

3.3: Modify /usr/src/linux-5.19.8/include/asm-generic/syscalls.h:

Sep 16 21:56

*syscalls.h [Read-Only]
/usr/src/linux-5.19.8/include/asm-generic

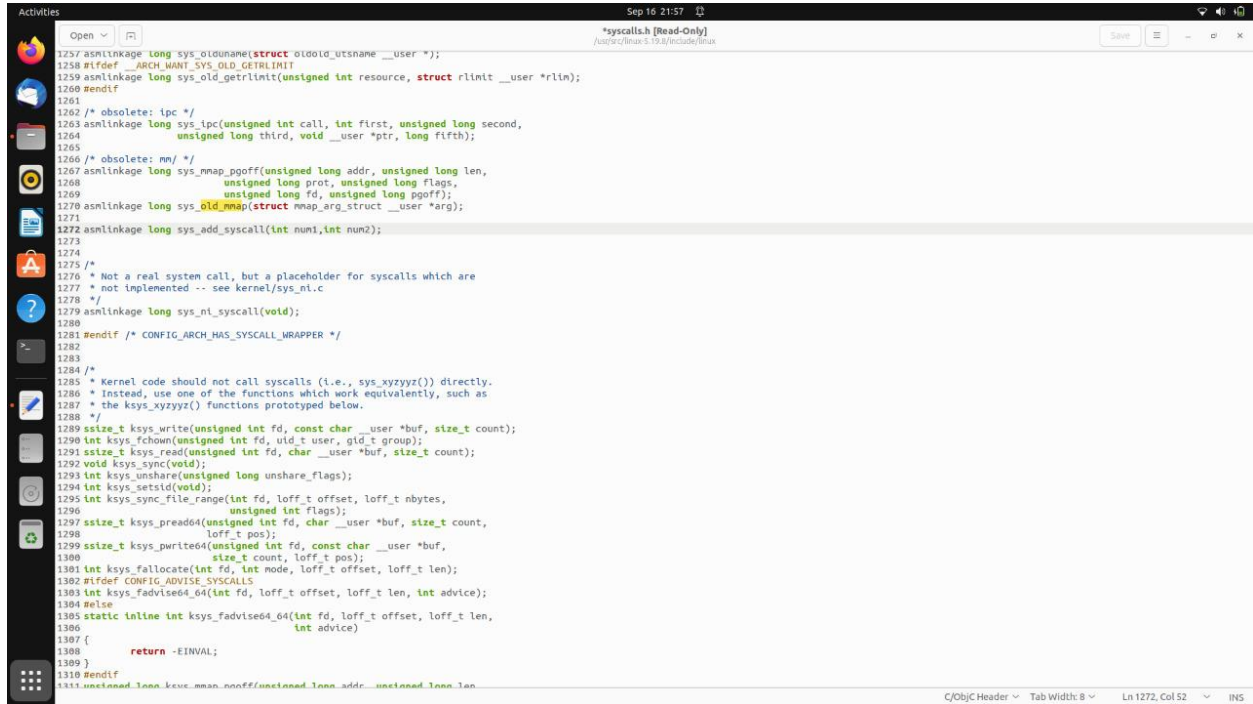
```

1 /* SPDX-License-Identifier: GPL-2.0 */
2 #ifndef __ASM_GENERIC_SYSCALLS_H
3 #define __ASM_GENERIC_SYSCALLS_H
4
5 #include <linux/compiler.h>
6 #include <linux/linkage.h>
7
8 /*
9  * Calling conventions for these system calls can differ, so
10  * it's possible to override them.
11  */
12
13 #ifndef sys_mmap2
14 asmlinkage long sys_mmap2(unsigned long addr, unsigned long len,
15                          unsigned long prot, unsigned long flags,
16                          unsigned long fd, unsigned long pgoff);
17 #endif
18
19 #ifndef sys_mmap
20 asmlinkage long sys_mmap(unsigned long addr, unsigned long len,
21                          unsigned long prot, unsigned long flags,
22                          unsigned long fd, off_t pgoff);
23 #endif
24
25 #ifndef sys_rt_sigreturn
26 asmlinkage long sys_rt_sigreturn(struct pt_regs *regs);
27 #endif
28
29 #ifndef sys_rt_sigreturn
30 asmlinkage long sys_add_syscall(int num1, int num2);
31 #endif
32
33 #endif /* __ASM_GENERIC_SYSCALLS_H */

```

C/ObjC Header Tab Width: 8 Ln 33, Col 38 INS

3.4: Modify /usr/src/linux-5.19.8/include/linux/syscalls.h:



```
1257 asmmlinkage long sys_olduname(struct oldold_utsname __user *);
1258 #ifdef __ARCH_WANT_SYS_OLD_GETRLIMIT
1259 asmmlinkage long sys_old_getrlimit(unsigned int resource, struct rlimit __user *rlim);
1260 #endif
1261
1262 /* obsolete: ipc */
1263 asmmlinkage long sys_ipc(unsigned int call, int first, unsigned long second,
1264                          unsigned long third, void __user *ptr, long fifth);
1265
1266 /* obsolete: mm */
1267 asmmlinkage long sys_mmap_pgoff(unsigned long addr, unsigned long len,
1268                                unsigned long prot, unsigned long flags,
1269                                unsigned long fd, unsigned long pgoff);
1270 asmmlinkage long sys_old_mmap(struct mmap_arg_struct __user *arg);
1271
1272 asmmlinkage long sys_add_syscall(int num1, int num2);
1273
1274
1275 /*
1276  * Not a real system call, but a placeholder for syscalls which are
1277  * not implemented -- see kernel/sys_ni.c
1278  */
1279 asmmlinkage long sys_ni_syscall(void);
1280
1281 #endif /* CONFIG_ARCH_HAS_SYSCALL_WRAPPER */
1282
1283
1284 /*
1285  * Kernel code should not call syscalls (i.e., sys_xyzzyz()) directly.
1286  * Instead, use one of the functions which work equivalently, such as
1287  * the ksys_xyzzyz() functions prototyped below.
1288  */
1289 ssize_t ksys_write(unsigned int fd, const char __user *buf, size_t count);
1290 int ksys_fchown(unsigned int fd, uid_t user, gid_t group);
1291 ssize_t ksys_read(unsigned int fd, char __user *buf, size_t count);
1292 void ksys_sync(void);
1293 int ksys_unshare(unsigned long unshare_flags);
1294 int ksys_setuid(void);
1295 int ksys_sync_file_range(int fd, loff_t offset, loff_t nbytes,
1296                          unsigned int flags);
1297 ssize_t ksys_pread4(unsigned int fd, char __user *buf, size_t count,
1298                    loff_t pos);
1299 ssize_t ksys_pwrite4(unsigned int fd, const char __user *buf,
1300                     size_t count, loff_t pos);
1301 int ksys_fallocate(int fd, int mode, loff_t offset, loff_t len);
1302 #ifdef CONFIG_ADVISE_SYSCALLS
1303 int ksys_fadvise64_04(int fd, loff_t offset, loff_t len, int advice);
1304 #else
1305 static inline int ksys_fadvise64_04(int fd, loff_t offset, loff_t len,
1306                                     int advice)
1307 {
1308     return -EINVAL;
1309 }
1310 #endif
1311 #endif
1312 #endif
```

Recompile the Kernel [Follow section#2]to get all the changes reflected. Reboot the system and boot into this kernel from the grub <Select advanced ubuntu tab followed by the New kernel>

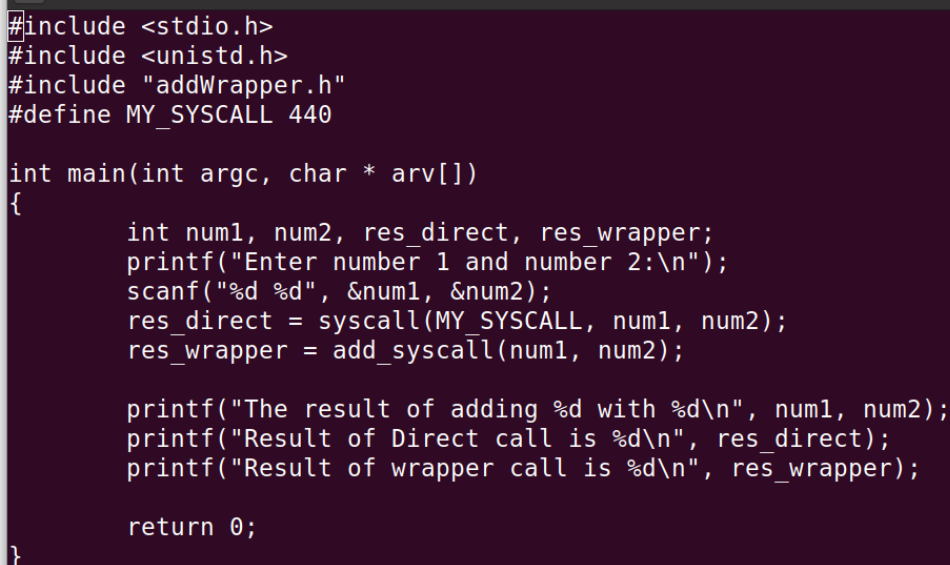
Implementation of User Space Programs

1. add2Num.c

2. addWrapper.h

The C user library wraps most system calls for us. This avoids triggering interrupts directly. The user space .c file provides two mechanisms of calling a system call (A) directly using the *syscall()* function with the help of system call number [GNU C library provides this for us] and (B) with the help of a Wrapper where the end user never need to remember the system call number.

1.1: add2Num.c:



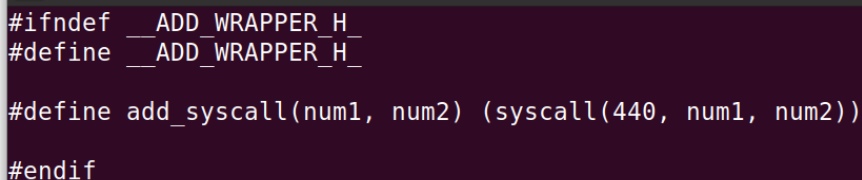
```
#include <stdio.h>
#include <unistd.h>
#include "addWrapper.h"
#define MY_SYSCALL 440

int main(int argc, char * arv[])
{
    int num1, num2, res_direct, res_wrapper;
    printf("Enter number 1 and number 2:\n");
    scanf("%d %d", &num1, &num2);
    res_direct = syscall(MY_SYSCALL, num1, num2);
    res_wrapper = add_syscall(num1, num2);

    printf("The result of adding %d with %d\n", num1, num2);
    printf("Result of Direct call is %d\n", res_direct);
    printf("Result of wrapper call is %d\n", res_wrapper);

    return 0;
}
```

1.2: addWrapper.h:

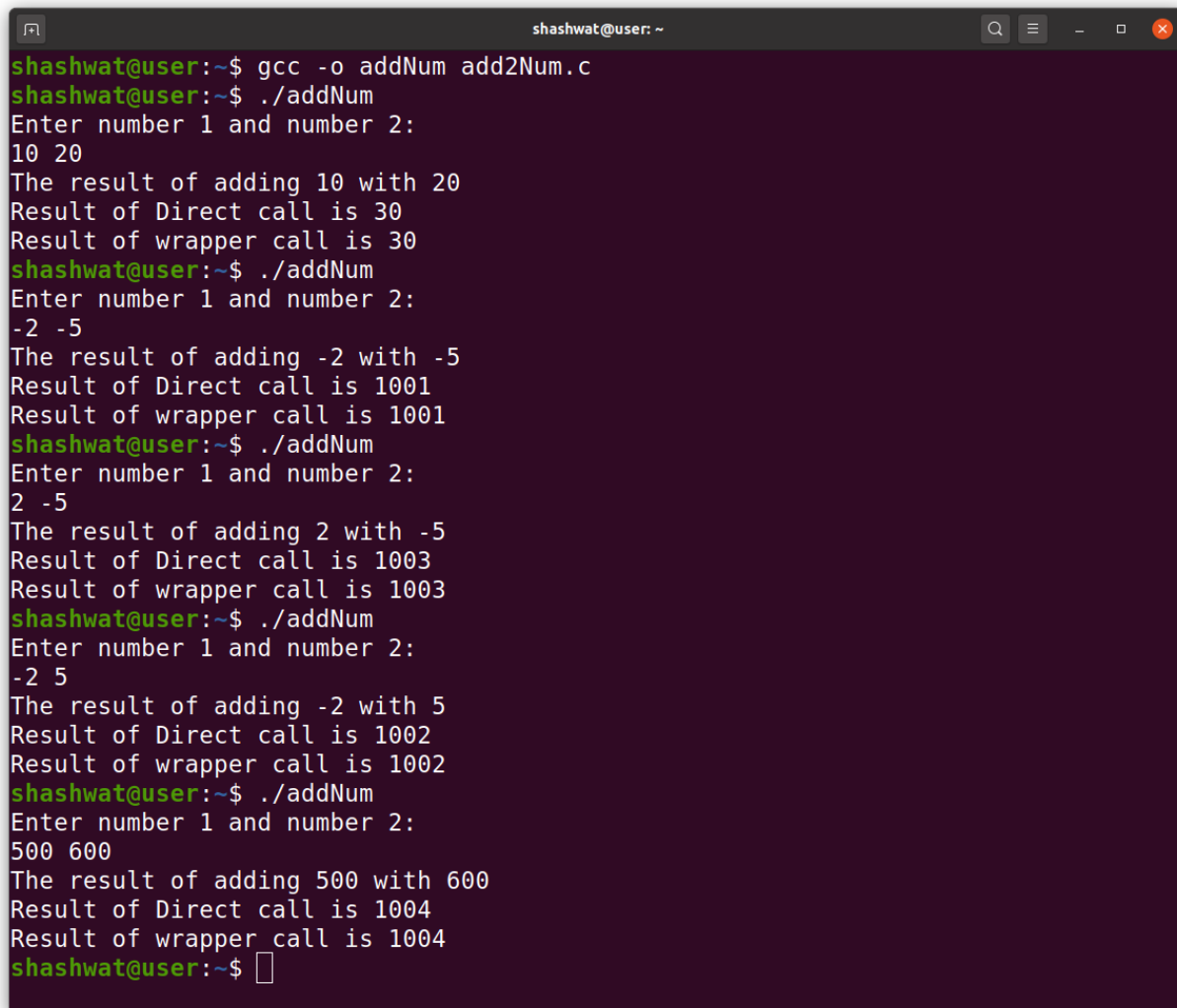


```
#ifndef __ADD_WRAPPER_H_
#define __ADD_WRAPPER_H_

#define add_syscall(num1, num2) (syscall(440, num1, num2))

#endif
```

1.3: Compiling and Executing the User program:



```
shashwat@user:~$ gcc -o addNum add2Num.c
shashwat@user:~$ ./addNum
Enter number 1 and number 2:
10 20
The result of adding 10 with 20
Result of Direct call is 30
Result of wrapper call is 30
shashwat@user:~$ ./addNum
Enter number 1 and number 2:
-2 -5
The result of adding -2 with -5
Result of Direct call is 1001
Result of wrapper call is 1001
shashwat@user:~$ ./addNum
Enter number 1 and number 2:
2 -5
The result of adding 2 with -5
Result of Direct call is 1003
Result of wrapper call is 1003
shashwat@user:~$ ./addNum
Enter number 1 and number 2:
-2 5
The result of adding -2 with 5
Result of Direct call is 1002
Result of wrapper call is 1002
shashwat@user:~$ ./addNum
Enter number 1 and number 2:
500 600
The result of adding 500 with 600
Result of Direct call is 1004
Result of wrapper call is 1004
shashwat@user:~$
```

Practice Problem:

Write a New system call in Kernel space which will add 2 floating point numbers and return the result to the user space. Make sure both the floating point numbers are Valid Positive Numbers. Make sure the result is a Valid Positive floating point number.