# Reading Material for System Call Implementation in Linux Operating System [Including Kernel Recompilation]

**As Part of**

**Operating Systems [CS F372] Course**

**Semester I, 2019 - 2020**

**BITS Pilani**
K K Birla Goa Campus

**Prepared by**

**HARSHA V, VEERENDRA VIDYADHAR RISBUD, PALLAVI VARSHNEY AND KARUNA GREWAL**

**Edited by**

**BIJU K RAVEENDRAN, Dept. of CS & IS**

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

## INDEX

## Basic Preparation for Recompilation

Step #1. Download the kernel source 5.2.9 from [ftp://10.1.9.224/Linux/Kernels/](ftp://10.1.9.224/Linux/Kernels/) or from [https://www.kernel.org](https://www.kernel.org)

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

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

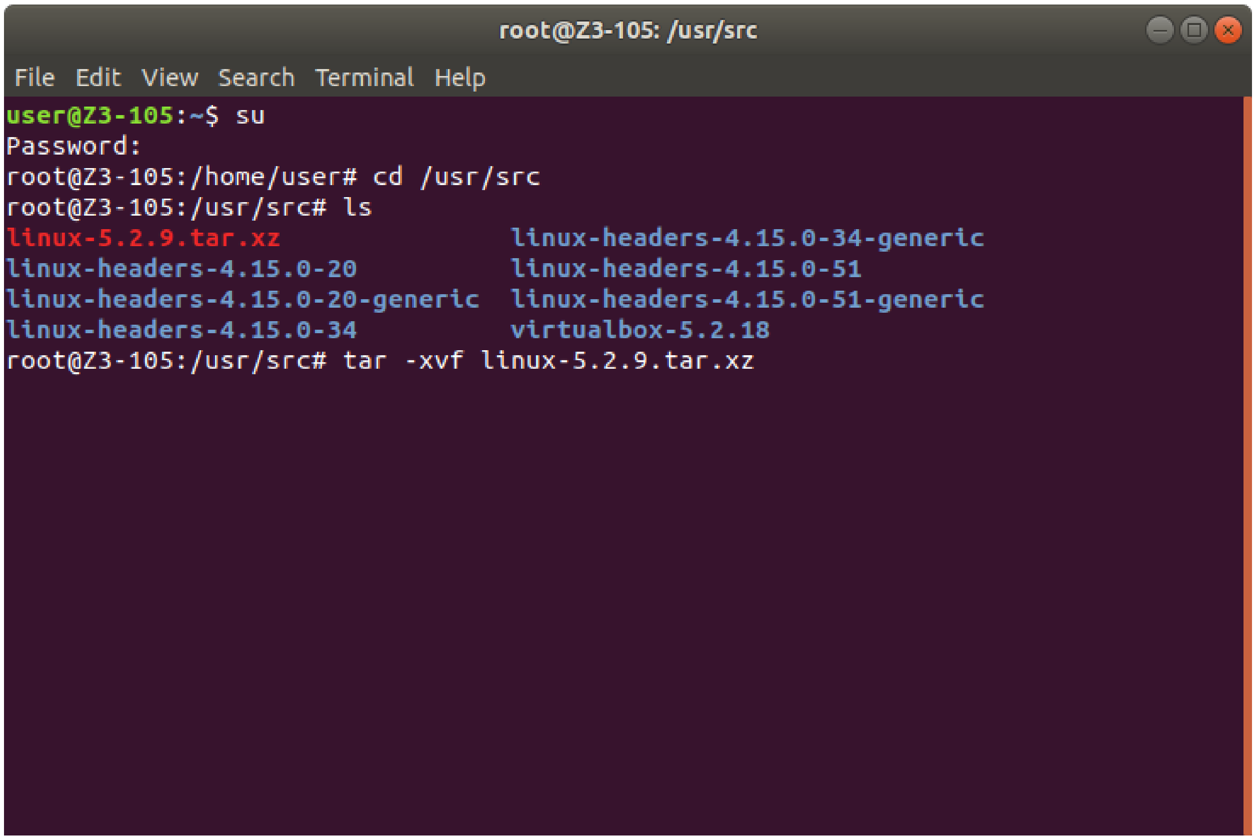
Step #3: Place the tar.xz file in /usr/src/ directory

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

$ ***cd /usr/src/***

Step #5: Untar the ***linux-5.2.9.tar.xz*** file by

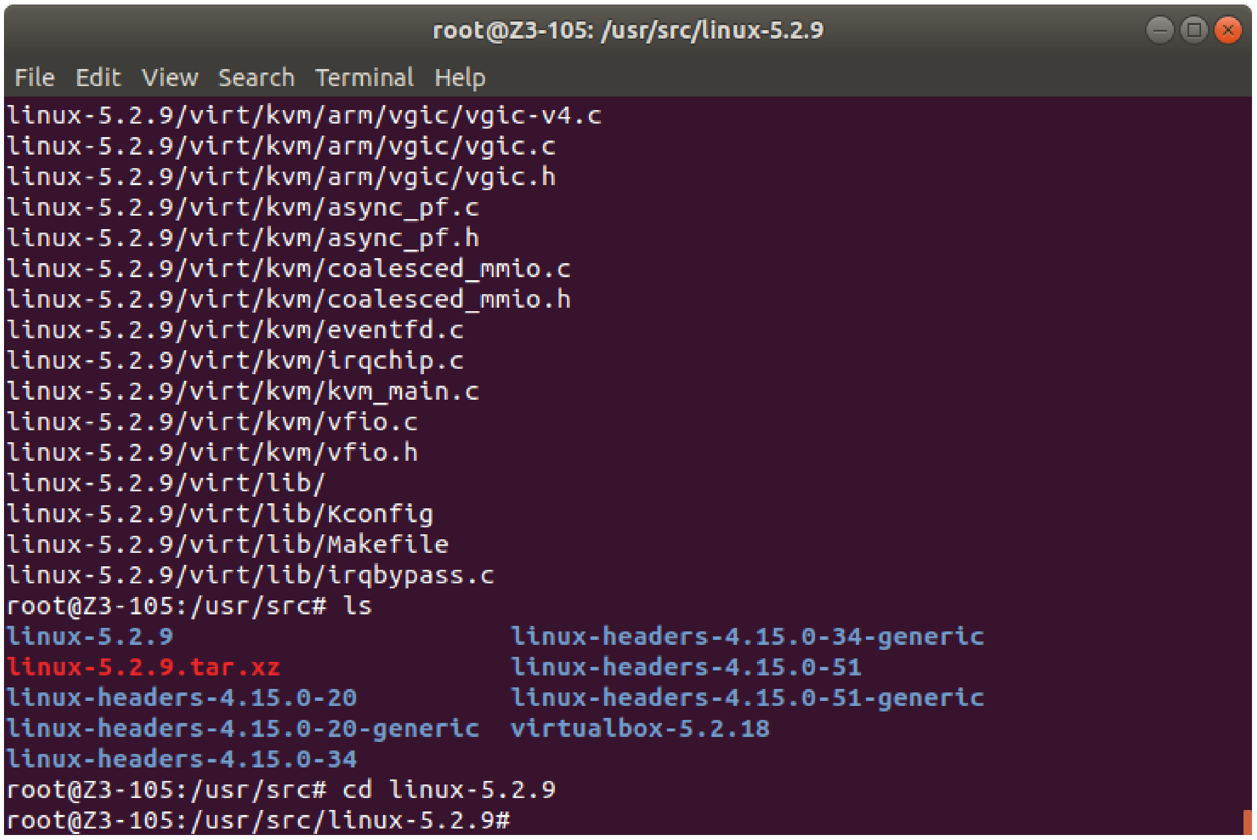
$ ***tar -xvf linux-5.2.9.tar.xz***



```
user@Z3-105:~$ su
Password:
root@Z3-105:/home/user# cd /usr/src
root@Z3-105:/usr/src# ls
linux-5.2.9.tar.xz               linux-headers-4.15.0-34-generic
linux-headers-4.15.0-20          linux-headers-4.15.0-51
linux-headers-4.15.0-20-generic  linux-headers-4.15.0-51-generic
linux-headers-4.15.0-34          virtualbox-5.2.18
root@Z3-105:/usr/src# tar -xvf linux-5.2.9.tar.xz
```

Step #6: Set the present working directory as linux-5.2.9

$ ***cd linux-5.2.9/***


root@Z3-105: /usr/src/linux-5.2.9
File Edit View Search Terminal Help

```
linux-5.2.9/virt/kvm/arm/vgic/vgic-v4.c
linux-5.2.9/virt/kvm/arm/vgic/vgic.c
linux-5.2.9/virt/kvm/arm/vgic/vgic.h
linux-5.2.9/virt/kvm/async_pf.c
linux-5.2.9/virt/kvm/async_pf.h
linux-5.2.9/virt/kvm/coalesced_mmio.c
linux-5.2.9/virt/kvm/coalesced_mmio.h
linux-5.2.9/virt/kvm/eventfd.c
linux-5.2.9/virt/kvm/irqchip.c
linux-5.2.9/virt/kvm/kvm_main.c
linux-5.2.9/virt/kvm/vfio.c
linux-5.2.9/virt/kvm/vfio.h
linux-5.2.9/virt/lib/
linux-5.2.9/virt/lib/Kconfig
linux-5.2.9/virt/lib/Makefile
linux-5.2.9/virt/lib/irqbypass.c
root@Z3-105:/usr/src# ls
linux-5.2.9                       linux-headers-4.15.0-34-generic
linux-5.2.9.tar.xz                linux-headers-4.15.0-51
linux-headers-4.15.0-20           linux-headers-4.15.0-51-generic
linux-headers-4.15.0-20-generic   virtualbox-5.2.18
linux-headers-4.15.0-34
root@Z3-105:/usr/src# cd linux-5.2.9
root@Z3-105:/usr/src/linux-5.2.9#
```

# 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**

**$apt install flex bison libssl-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**

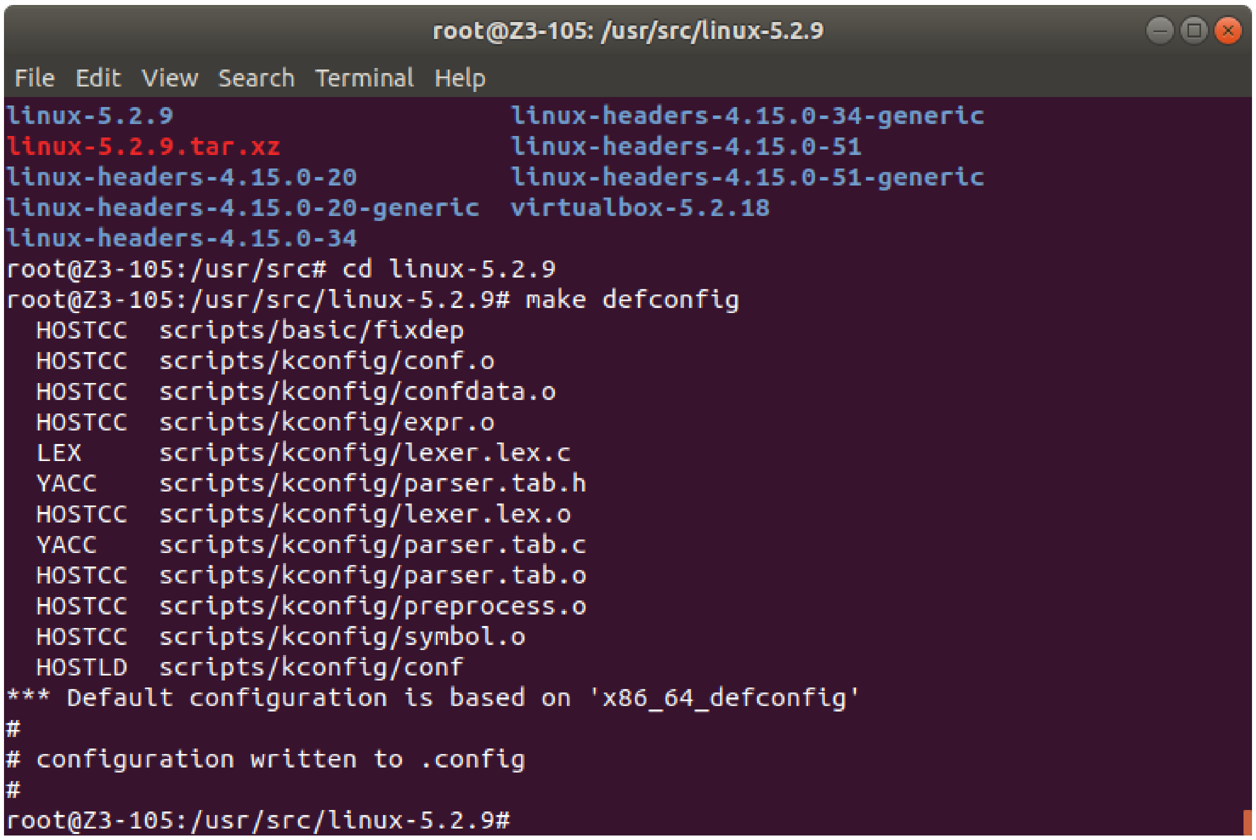
<X windows (Qt) based configuration tool, works best under KDE Desktop.>

**$ 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>

**$ 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>

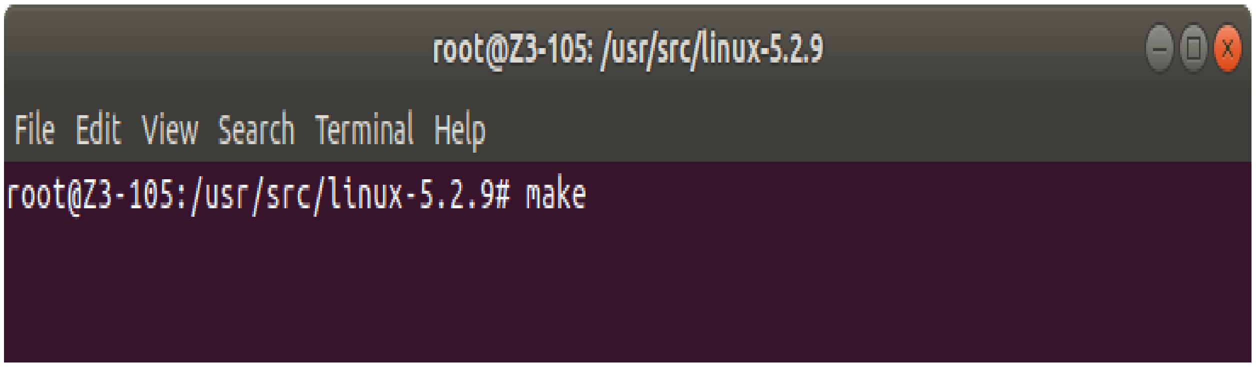

root@Z3-105: /usr/src/linux-5.2.9
File Edit View Search Terminal Help

```
linux-5.2.9                       linux-headers-4.15.0-34-generic
linux-5.2.9.tar.xz                linux-headers-4.15.0-51
linux-headers-4.15.0-20           linux-headers-4.15.0-51-generic
linux-headers-4.15.0-20-generic   virtualbox-5.2.18
linux-headers-4.15.0-34
root@Z3-105:/usr/src# cd linux-5.2.9
root@Z3-105:/usr/src/linux-5.2.9# make defconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.h
  HOSTCC  scripts/kconfig/lexer.lex.o
  YACC    scripts/kconfig/parser.tab.c
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTLD  scripts/kconfig/conf
*** Default configuration is based on 'x86_64_defconfig'
#
# configuration written to .config
#
root@Z3-105:/usr/src/linux-5.2.9#
```

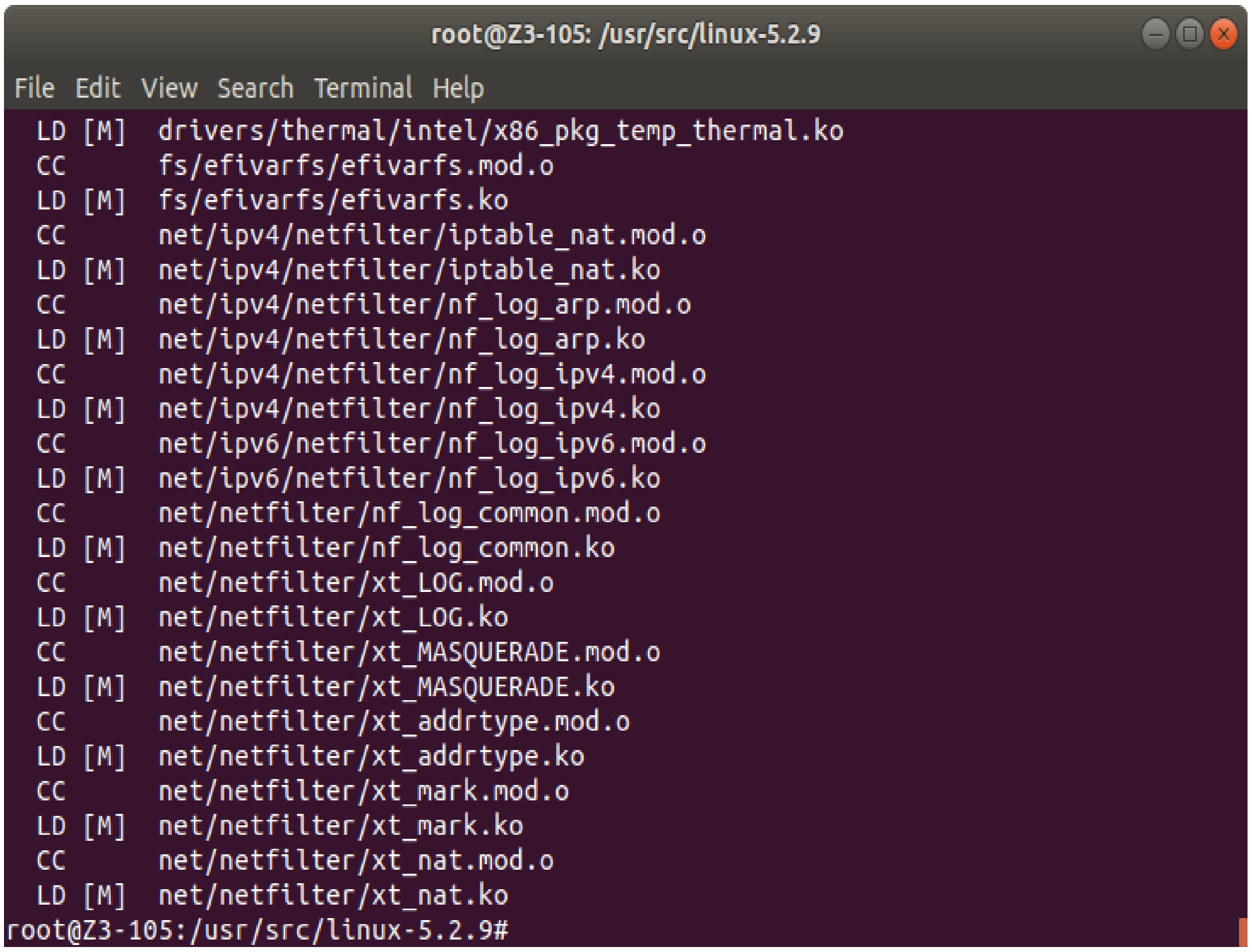
## Step #2: Preliminary Recompilation of the Kernel

Execute make to compile the kernel.

$ make


root@Z3-105: /usr/src/linux-5.2.9
File Edit View Search Terminal Help

```
root@Z3-105:/usr/src/linux-5.2.9# make
```


root@Z3-105: /usr/src/linux-5.2.9
File Edit View Search Terminal Help

```
  LD [M]  drivers/thermal/intel/x86_pkg_temp_thermal.ko
  CC      fs/efivarfs/efivarfs.mod.o
  LD [M]  fs/efivarfs/efivarfs.ko
  CC      net/ipv4/netfilter/iptable_nat.mod.o
  LD [M]  net/ipv4/netfilter/iptable_nat.ko
  CC      net/ipv4/netfilter/nf_log_arp.mod.o
  LD [M]  net/ipv4/netfilter/nf_log_arp.ko
  CC      net/ipv4/netfilter/nf_log_ipv4.mod.o
  LD [M]  net/ipv4/netfilter/nf_log_ipv4.ko
  CC      net/ipv6/netfilter/nf_log_ipv6.mod.o
  LD [M]  net/ipv6/netfilter/nf_log_ipv6.ko
  CC      net/netfilter/nf_log_common.mod.o
  LD [M]  net/netfilter/nf_log_common.ko
  CC      net/netfilter/xt_LOG.mod.o
  LD [M]  net/netfilter/xt_LOG.ko
  CC      net/netfilter/xt_MASQUERADE.mod.o
  LD [M]  net/netfilter/xt_MASQUERADE.ko
  CC      net/netfilter/xt_addrtype.mod.o
  LD [M]  net/netfilter/xt_addrtype.ko
  CC      net/netfilter/xt_mark.mod.o
  LD [M]  net/netfilter/xt_mark.ko
  CC      net/netfilter/xt_nat.mod.o
  LD [M]  net/netfilter/xt_nat.ko
root@Z3-105:/usr/src/linux-5.2.9#
```

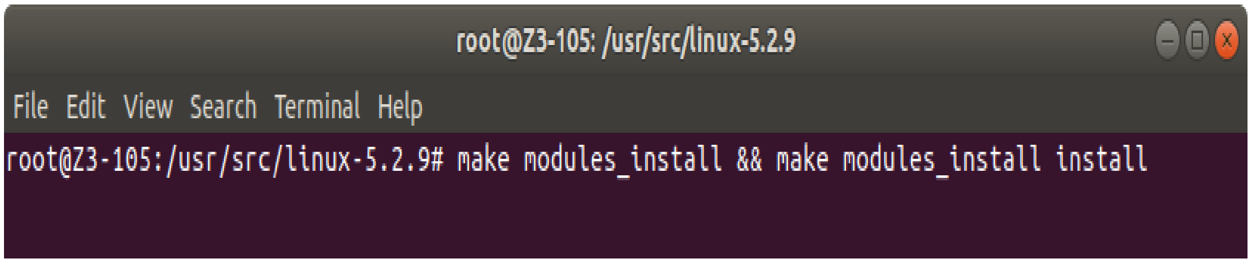
## Step #3: Recompilation of the Kernel module, update initramfs and grub

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

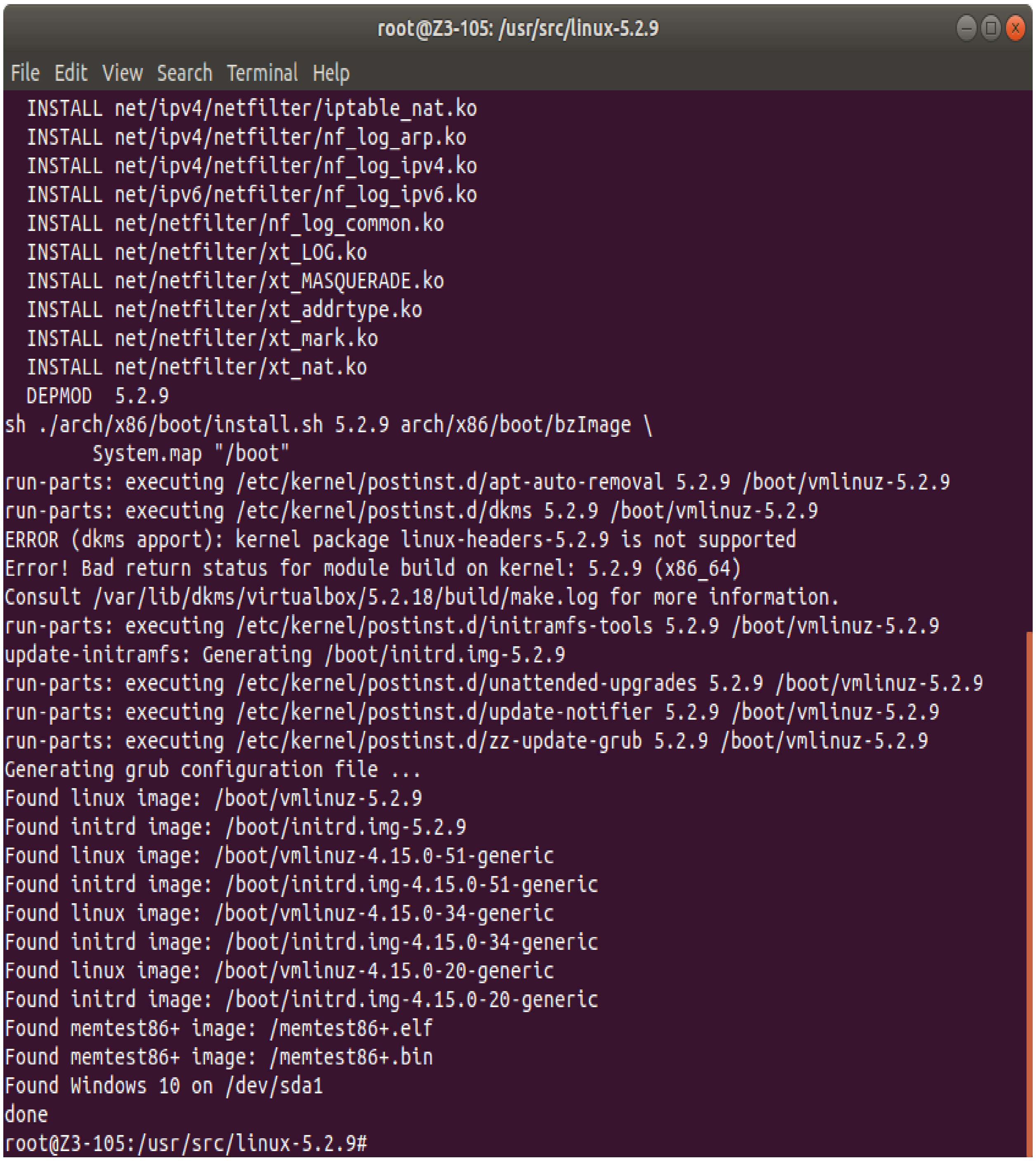
## $ make modules_install && make modules_install install

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

update-initramfs -c -k linux-5.2.9
update-grub


root@Z3-105: /usr/src/linux-5.2.9
File Edit View Search Terminal Help

```
root@Z3-105:/usr/src/linux-5.2.9# make modules_install && make modules_install install
```



```
  INSTALL net/ipv4/netfilter/iptable_nat.ko
  INSTALL net/ipv4/netfilter/nf_log_arp.ko
  INSTALL net/ipv4/netfilter/nf_log_ipv4.ko
  INSTALL net/ipv6/netfilter/nf_log_ipv6.ko
  INSTALL net/netfilter/nf_log_common.ko
  INSTALL net/netfilter/xt_LOG.ko
  INSTALL net/netfilter/xt_MASQUERADE.ko
  INSTALL net/netfilter/xt_addrtype.ko
  INSTALL net/netfilter/xt_mark.ko
  INSTALL net/netfilter/xt_nat.ko
  DEPMOD  5.2.9
sh ./arch/x86/boot/install.sh 5.2.9 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 5.2.9 /boot/vmlinuz-5.2.9
run-parts: executing /etc/kernel/postinst.d/dkms 5.2.9 /boot/vmlinuz-5.2.9
ERROR (dkms apport): kernel package linux-headers-5.2.9 is not supported
Error! Bad return status for module build on kernel: 5.2.9 (x86_64)
Consult /var/lib/dkms/virtualbox/5.2.18/build/make.log for more information.
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.2.9 /boot/vmlinuz-5.2.9
update-initramfs: Generating /boot/initrd.img-5.2.9
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.2.9 /boot/vmlinuz-5.2.9
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.2.9 /boot/vmlinuz-5.2.9
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.2.9 /boot/vmlinuz-5.2.9
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.2.9
Found initrd image: /boot/initrd.img-5.2.9
Found linux image: /boot/vmlinuz-4.15.0-51-generic
Found initrd image: /boot/initrd.img-4.15.0-51-generic
Found linux image: /boot/vmlinuz-4.15.0-34-generic
Found initrd image: /boot/initrd.img-4.15.0-34-generic
Found linux image: /boot/vmlinuz-4.15.0-20-generic
Found initrd image: /boot/initrd.img-4.15.0-20-generic
Found memtest86+ image: /memtest86+.elf
Found memtest86+ image: /memtest86+.bin
Found Windows 10 on /dev/sda1
done
root@Z3-105:/usr/src/linux-5.2.9#
```

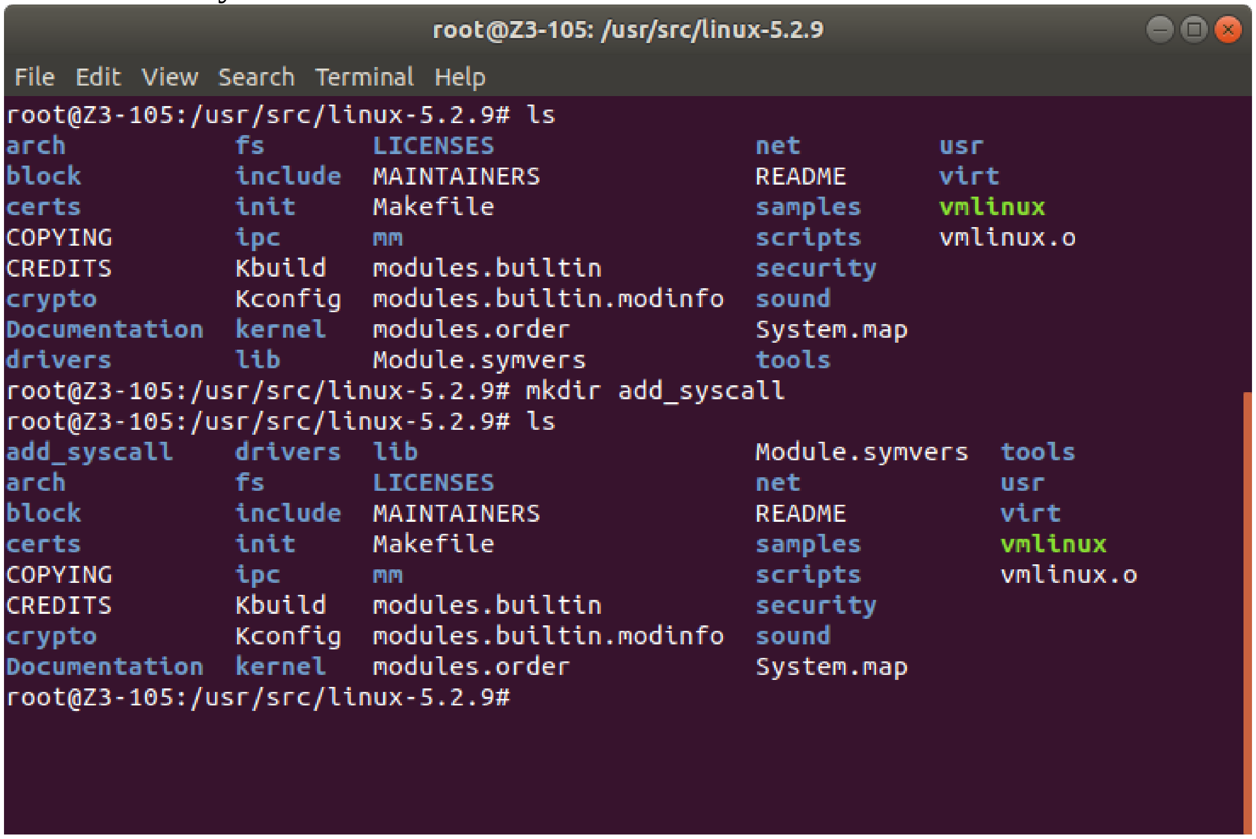
**Now that the kernel has been recompiled, reboot the system and boot into this kernel from the grub <Select advanced ubuntu tab followed by the New kernel>**

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

## Step #1: Create a directory under /usr/src/linux-5.2.9/

Create a directory named add_syscall under /usr/src/linux-5.2.9/

$ mkdir add_syscall


root@Z3-105: /usr/src/linux-5.2.9
File Edit View Search Terminal Help

```
root@Z3-105:/usr/src/linux-5.2.9# ls
arch           fs       LICENSES                 net        usr
block          include  MAINTAINERS              README     virt
certs          init     Makefile                 samples    vmlinux
COPYING        ipc      mm                       scripts    vmlinux.o
CREDITS        Kbuild   modules.builtin          security
crypto         Kconfig  modules.builtin.modinfo  sound
Documentation  kernel   modules.order            System.map
drivers        lib      Module.symvers           tools
root@Z3-105:/usr/src/linux-5.2.9# mkdir add_syscall
root@Z3-105:/usr/src/linux-5.2.9# ls
add_syscall    drivers  lib                      Module.symvers  tools
arch           fs       LICENSES                 net             usr
block          include  MAINTAINERS              README          virt
certs          init     Makefile                 samples         vmlinux
COPYING        ipc      mm                       scripts         vmlinux.o
CREDITS        Kbuild   modules.builtin          security
crypto         Kconfig  modules.builtin.modinfo  sound
Documentation  kernel   modules.order            System.map
root@Z3-105:/usr/src/linux-5.2.9#
```

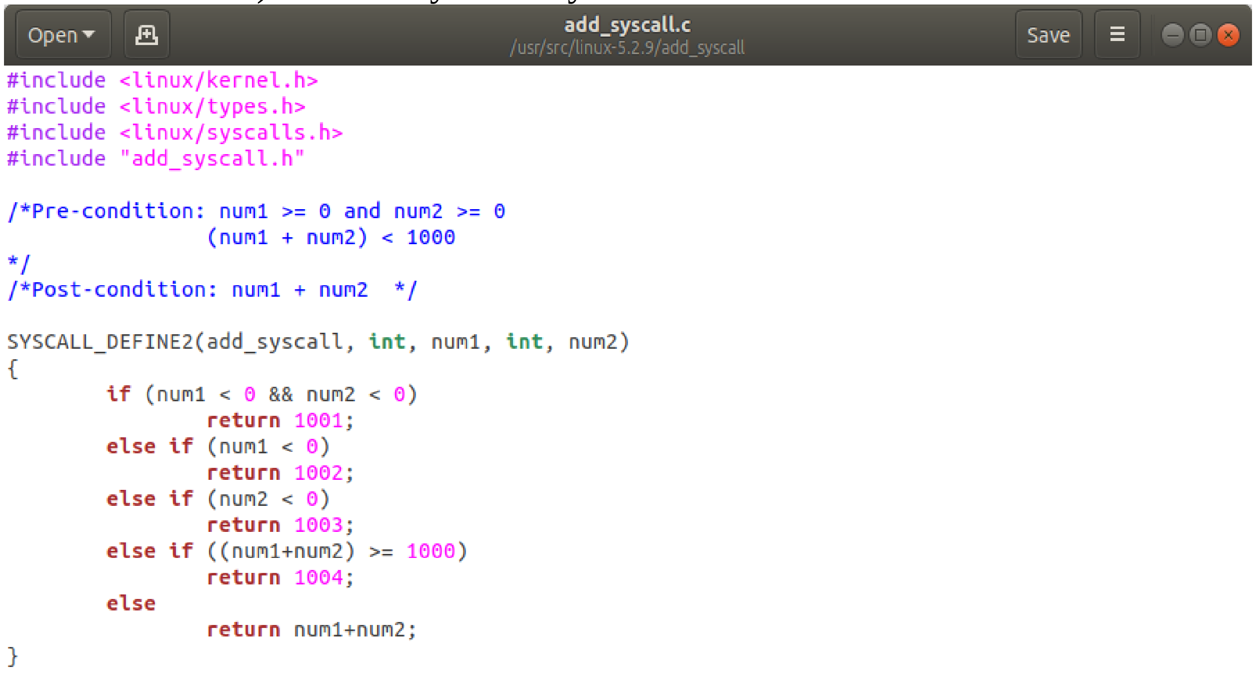
## Step #2: Create the following files under add_syscall directory

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

Contents of **add_syscall.c**

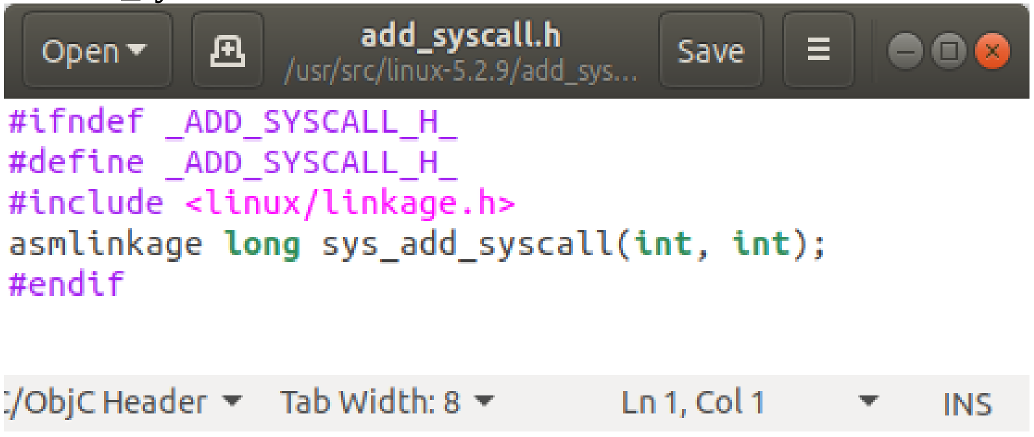
SYSCALL_DEFINE*n*() macros are the standard way for kernel code to define a system call, where the *n* suffix indicates the argument count. The first argument to the macro is the name of the system call (without sys_ prepended to it). The remaining arguments are pairs of type and name for the parameters.

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.



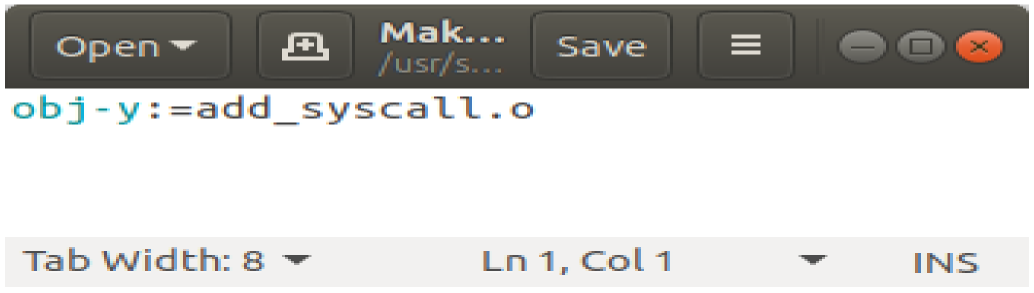
```
#include <linux/kernel.h>
#include <linux/types.h>
#include <linux/syscalls.h>
#include "add_syscall.h"

/*Pre-condition: num1 >= 0 and num2 >= 0
                (num1 + num2) < 1000
*/
/*Post-condition: num1 + num2  */

SYSCALL_DEFINE2(add_syscall, int, num1, int, num2)
{
        if (num1 < 0 && num2 < 0)
                return 1001;
        else if (num1 < 0)
                return 1002;
        else if (num2 < 0)
                return 1003;
        else if ((num1+num2) >= 1000)
                return 1004;
        else
                return num1+num2;
}
```

Content of **add_syscall.h**



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

Content of **Makefile**



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

## Step #3: Modify the following files

1. **/usr/src/linux-5.2.9/Makefile**
2. **/usr/src/linux-5.2.9/arch/x86/entry/syscalls/syscall_64.tbl**
3. **/usr/src/linux-5.2.9/include/asm-generic/syscalls.h**
4. **/usr/src/linux-5.2.9/include/linux/syscalls.h**

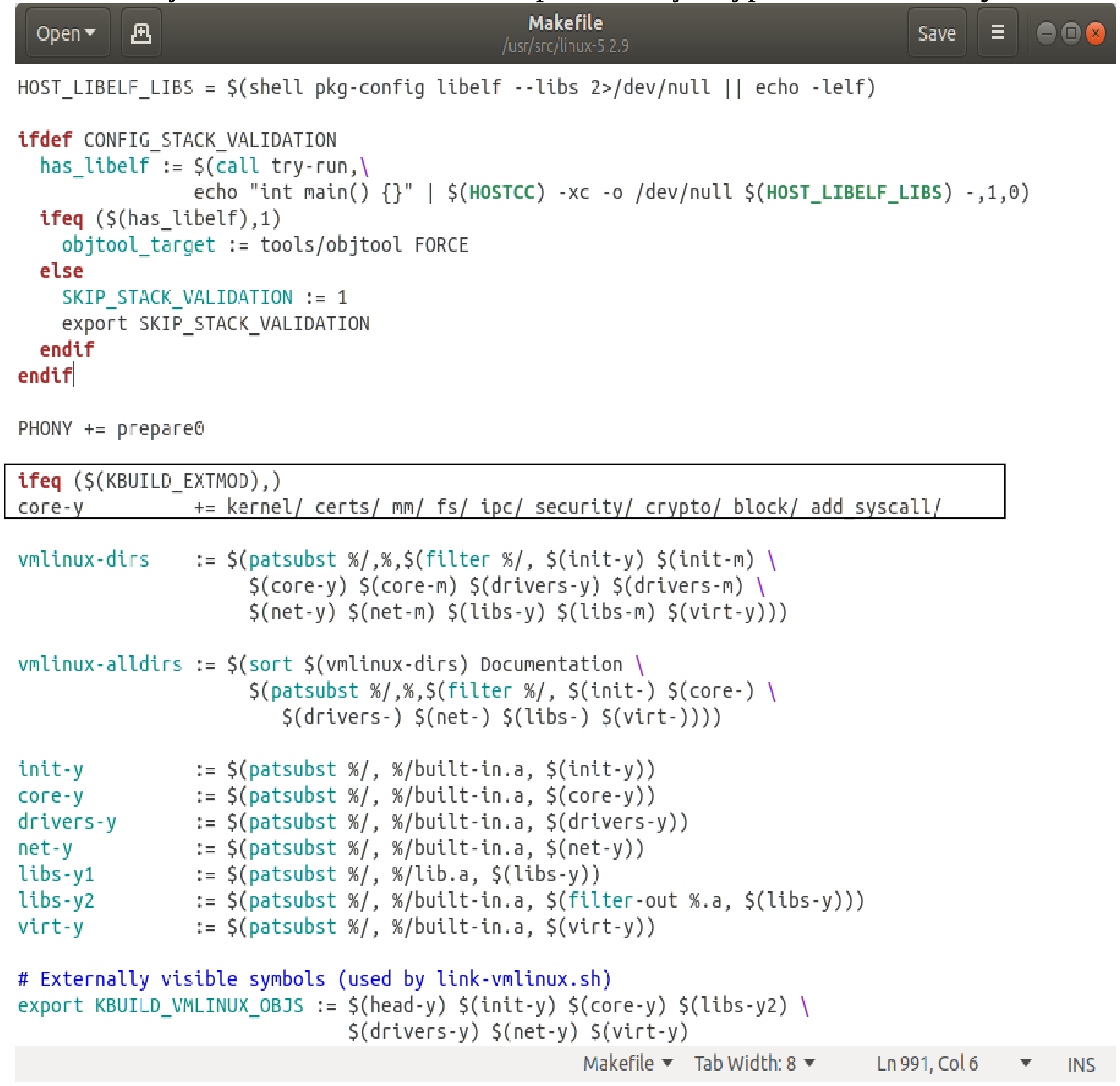
### 3.1: Modify /usr/src/linux-5.2.9/Makefile:

**<Update the following line in Makefile>**

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

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

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

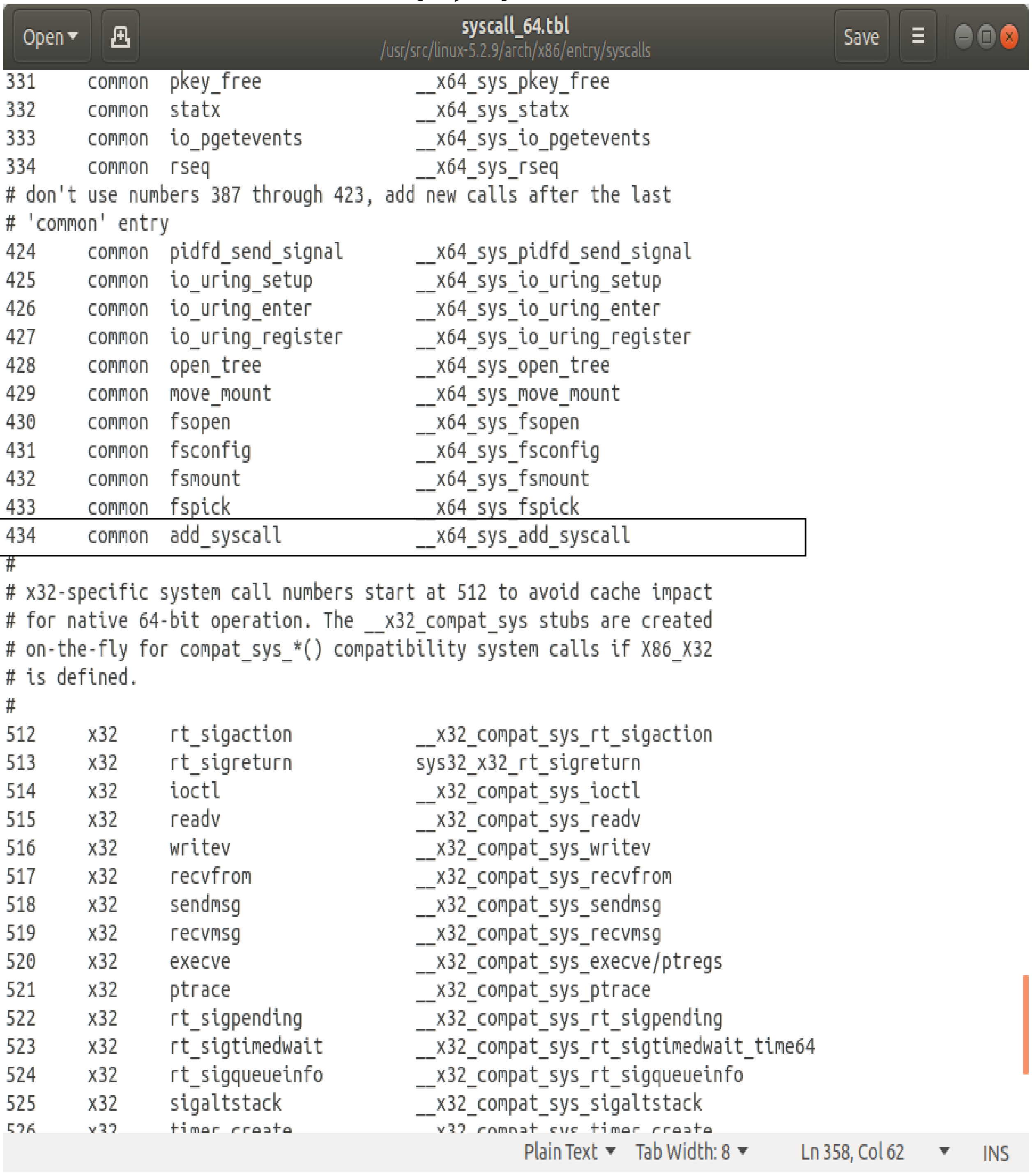


```
HOST_LIBELF_LIBS = $(shell pkg-config libelf --libs 2>/dev/null || echo -lelf)

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/ add_syscall/

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)))

vmlinux-alldirs := $(sort $(vmlinux-dirs) Documentation \
                     $(patsubst %/,%,$(filter %/, $(init-) $(core-) \
                        $(drivers-) $(net-) $(libs-) $(virt-))))

init-y          := $(patsubst %/, %/built-in.a, $(init-y))
core-y          := $(patsubst %/, %/built-in.a, $(core-y))
drivers-y       := $(patsubst %/, %/built-in.a, $(drivers-y))
net-y           := $(patsubst %/, %/built-in.a, $(net-y))
libs-y1         := $(patsubst %/, %/lib.a, $(libs-y))
libs-y2         := $(patsubst %/, %/built-in.a, $(filter-out %.a, $(libs-y)))
virt-y          := $(patsubst %/, %/built-in.a, $(virt-y))

# Externally visible symbols (used by link-vmlinux.sh)
export KBUILD_VMLINUX_OBJS := $(head-y) $(init-y) $(core-y) $(libs-y2) \
                              $(drivers-y) $(net-y) $(virt-y)
```

**3.2: Modify /usr/src/linux-5.2.9/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:

**434 common add_syscall __x64_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.**

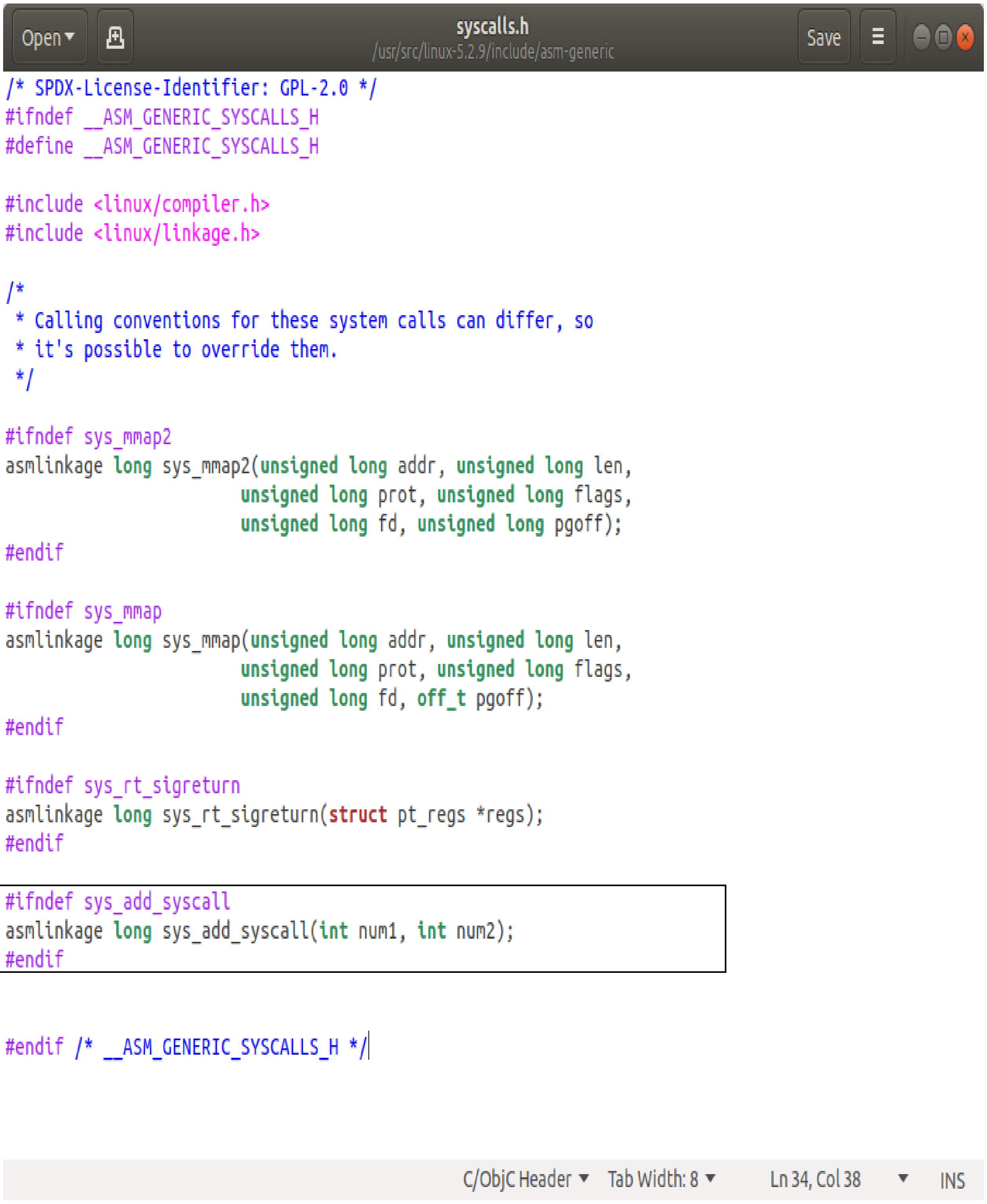

Open ▾ syscall_64.tbl /usr/src/linux-5.2.9/arch/x86/entry/syscalls Save

```
331     common  pkey_free               __x64_sys_pkey_free
332     common  statx                   __x64_sys_statx
333     common  io_pgetevents           __x64_sys_io_pgetevents
334     common  rseq                    __x64_sys_rseq
# don't use numbers 387 through 423, add new calls after the last
# 'common' entry
424     common  pidfd_send_signal       __x64_sys_pidfd_send_signal
425     common  io_uring_setup          __x64_sys_io_uring_setup
426     common  io_uring_enter          __x64_sys_io_uring_enter
427     common  io_uring_register       __x64_sys_io_uring_register
428     common  open_tree               __x64_sys_open_tree
429     common  move_mount              __x64_sys_move_mount
430     common  fsopen                  __x64_sys_fsopen
431     common  fsconfig                __x64_sys_fsconfig
432     common  fsmount                 __x64_sys_fsmount
433     common  fspick                  __x64_sys_fspick
434     common  add_syscall             __x64_sys_add_syscall
#
# 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
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_time64
524     x32     rt_sigqueueinfo         __x32_compat_sys_rt_sigqueueinfo
525     x32     sigaltstack             __x32_compat_sys_sigaltstack
```

Plain Text ▾ Tab Width: 8 ▾ Ln 358, Col 62 ▾ INS

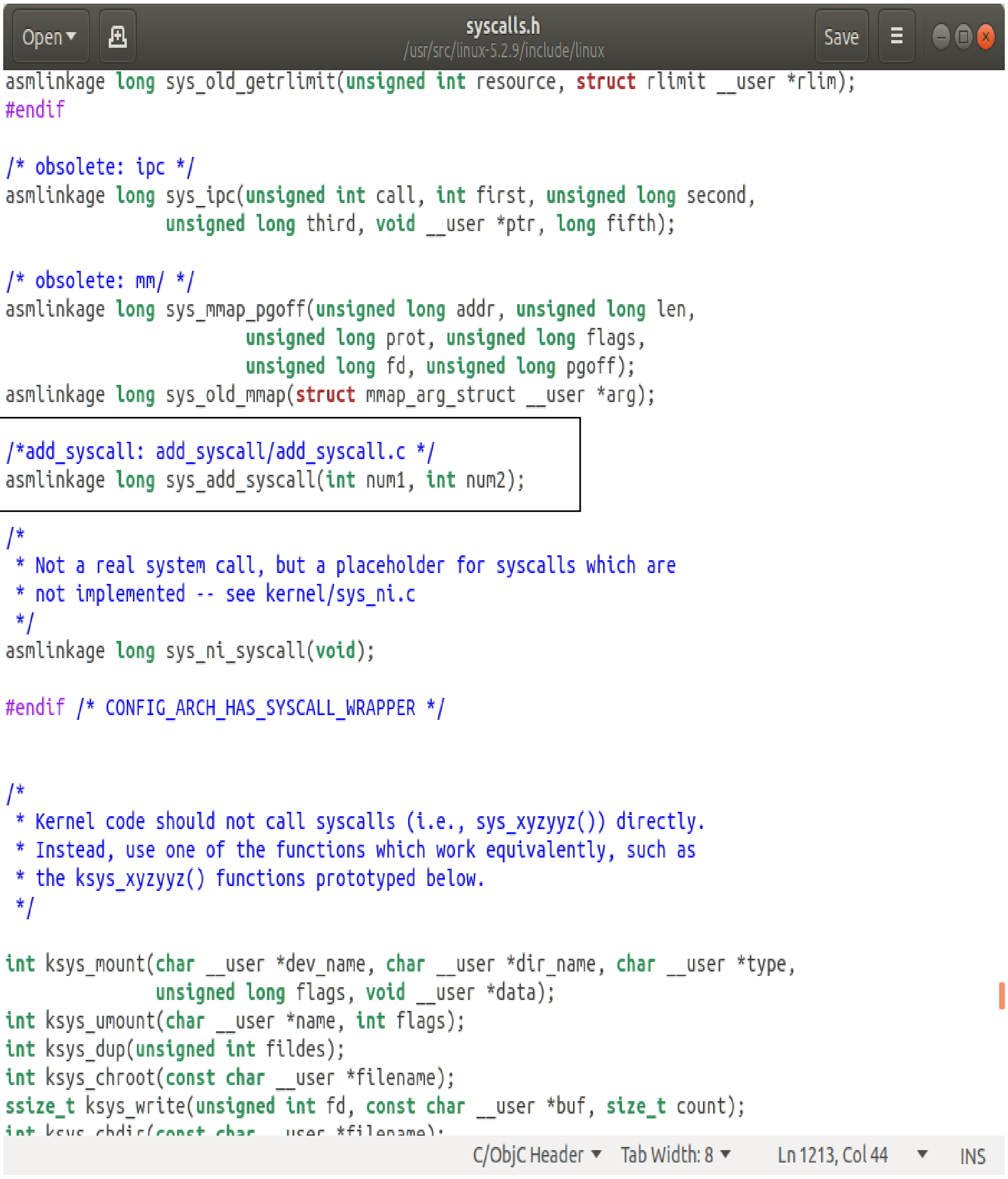
**This table is read by scripts and used to generate some of the boilerplate code**

## 3.3: Modify /usr/src/linux-5.2.9/include/asm-generic/syscalls.h:



```
/* SPDX-License-Identifier: GPL-2.0 */
#ifndef __ASM_GENERIC_SYSCALLS_H
#define __ASM_GENERIC_SYSCALLS_H

#include <linux/compiler.h>
#include <linux/linkage.h>

/*
 * Calling conventions for these system calls can differ, so
 * it's possible to override them.
 */

#ifndef sys_mmap2
asmlinkage long sys_mmap2(unsigned long addr, unsigned long len,
                        unsigned long prot, unsigned long flags,
                        unsigned long fd, unsigned long pgoff);
#endif

#ifndef sys_mmap
asmlinkage long sys_mmap(unsigned long addr, unsigned long len,
                        unsigned long prot, unsigned long flags,
                        unsigned long fd, off_t pgoff);
#endif

#ifndef sys_rt_sigreturn
asmlinkage long sys_rt_sigreturn(struct pt_regs *regs);
#endif

#ifndef sys_add_syscall
asmlinkage long sys_add_syscall(int num1, int num2);
#endif


#endif /* __ASM_GENERIC_SYSCALLS_H */
```

## 3.4: Modify /usr/src/linux-5.2.9/include/linux/syscalls.h:



```
asmlinkage long sys_old_getrlimit(unsigned int resource, struct rlimit __user *rlim);
#endif

/* obsolete: ipc */
asmlinkage long sys_ipc(unsigned int call, int first, unsigned long second,
                unsigned long third, void __user *ptr, long fifth);

/* obsolete: mm/ */
asmlinkage long sys_mmap_pgoff(unsigned long addr, unsigned long len,
                        unsigned long prot, unsigned long flags,
                        unsigned long fd, unsigned long pgoff);
asmlinkage long sys_old_mmap(struct mmap_arg_struct __user *arg);

/*add_syscall: add_syscall/add_syscall.c */
asmlinkage long sys_add_syscall(int num1, int num2);

/*
 * Not a real system call, but a placeholder for syscalls which are
 * not implemented -- see kernel/sys_ni.c
 */
asmlinkage long sys_ni_syscall(void);

#endif /* CONFIG_ARCH_HAS_SYSCALL_WRAPPER */


/*
 * Kernel code should not call syscalls (i.e., sys_xyzyyz()) directly.
 * Instead, use one of the functions which work equivalently, such as
 * the ksys_xyzyyz() functions prototyped below.
 */

int ksys_mount(char __user *dev_name, char __user *dir_name, char __user *type,
               unsigned long flags, void __user *data);
int ksys_umount(char __user *name, int flags);
int ksys_dup(unsigned int fildes);
int ksys_chroot(const char __user *filename);
ssize_t ksys_write(unsigned int fd, const char __user *buf, size_t count);
```

C/ObjC Header ▾   Tab Width: 8 ▾   Ln 1213, Col 44 ▾   INS

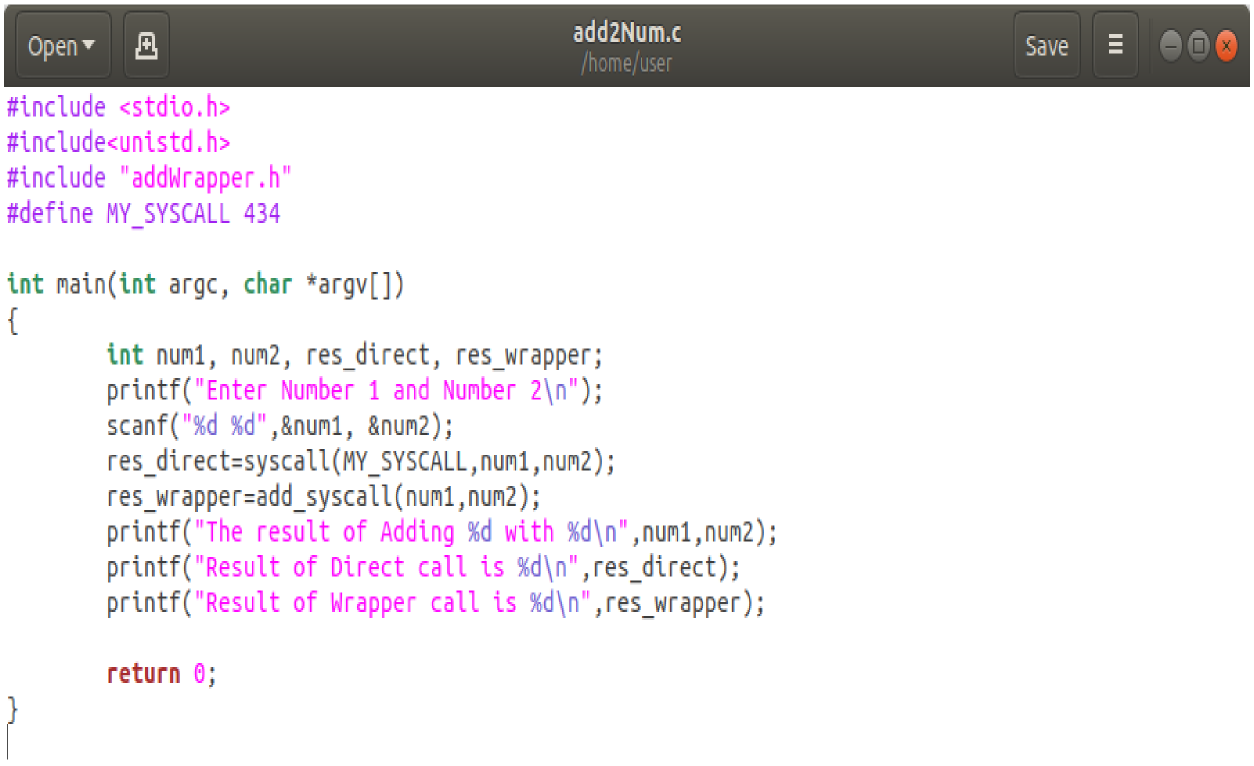
Recompile the Kernel [Follow section#2]to get all the changes reflected. Reboot the system and boot into this kernel from the grub <Select advanced ubuntu tab followed by the New kernel>

## Implementation of User Space Programs

1. **/home/user/add2Num.c**
2. **/home/user/addWrapper.h**

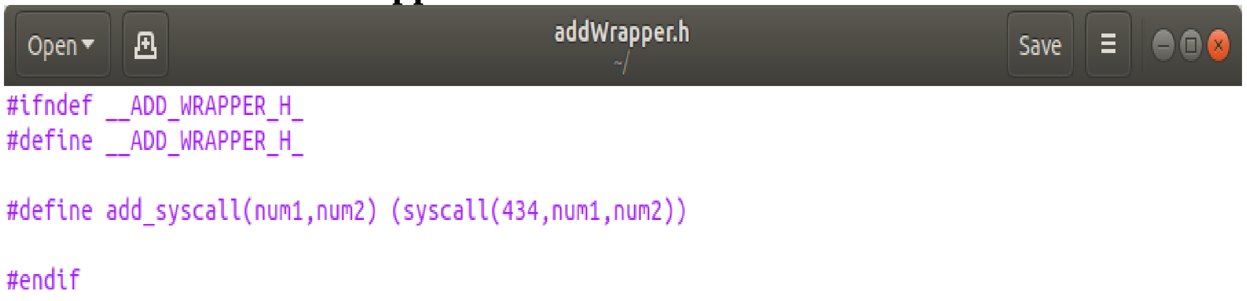
The C user library wraps most system calls for us. This avoids triggering interrupts directly. The user space .c file provides two mechanisms of calling a system call (A) directly using the ***syscall()*** function with the help of system call number [GNU C library provides this for us] and (B) with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

### 1.1: /home/user/add2Num.c:



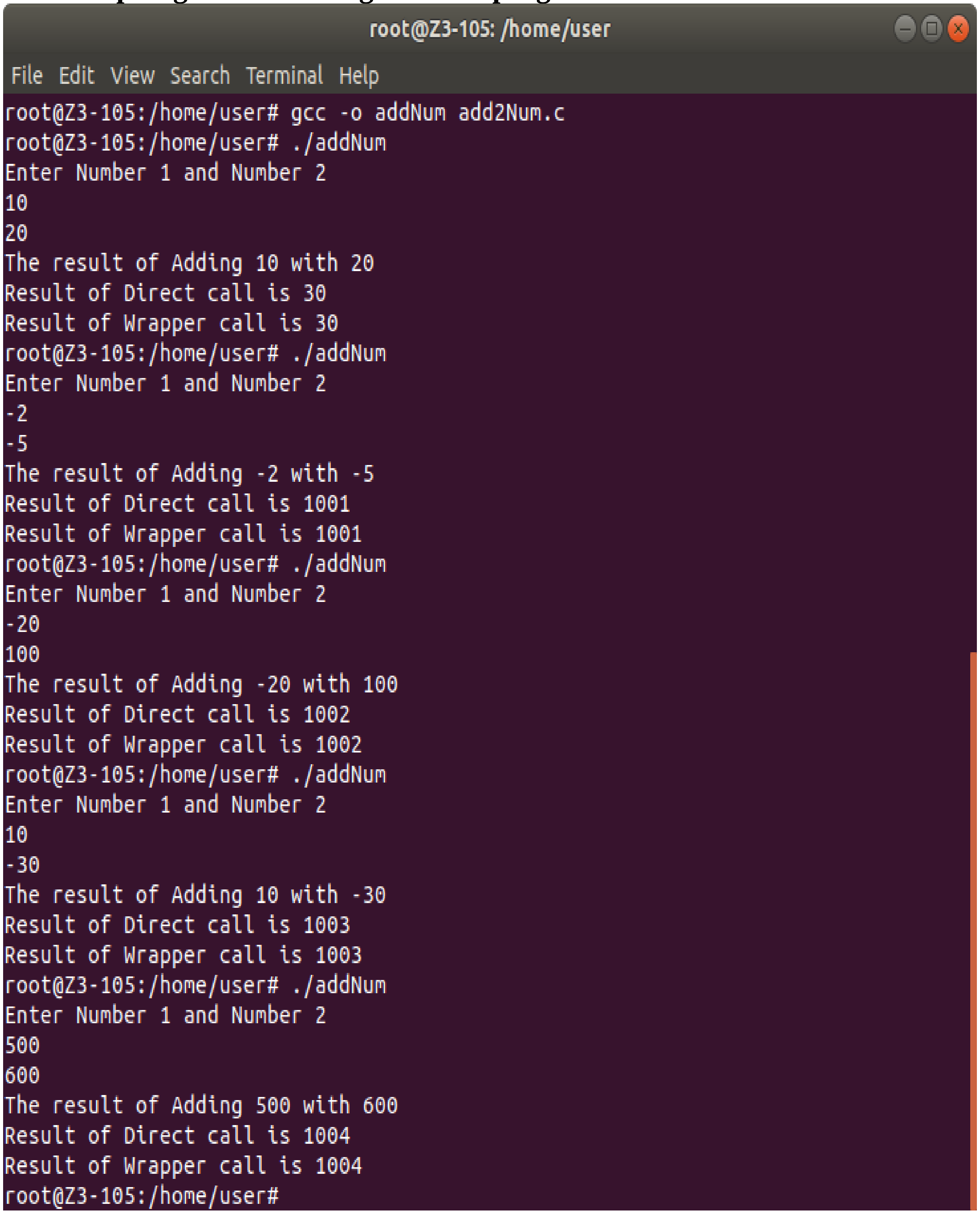
```c
#include <stdio.h>
#include<unistd.h>
#include "addWrapper.h"
#define MY_SYSCALL 434

int main(int argc, char *argv[])
{
        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: /home/user/addWrapper.h:



```c
#ifndef __ADD_WRAPPER_H_
#define __ADD_WRAPPER_H_

#define add_syscall(num1,num2) (syscall(434,num1,num2))

#endif
```

## 1.3: Compiling and Executing the User program:



```
root@Z3-105:/home/user# gcc -o addNum add2Num.c
root@Z3-105:/home/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
root@Z3-105:/home/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
root@Z3-105:/home/user# ./addNum
Enter Number 1 and Number 2
-20
100
The result of Adding -20 with 100
Result of Direct call is 1002
Result of Wrapper call is 1002
root@Z3-105:/home/user# ./addNum
Enter Number 1 and Number 2
10
-30
The result of Adding 10 with -30
Result of Direct call is 1003
Result of Wrapper call is 1003
root@Z3-105:/home/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
root@Z3-105:/home/user#
```