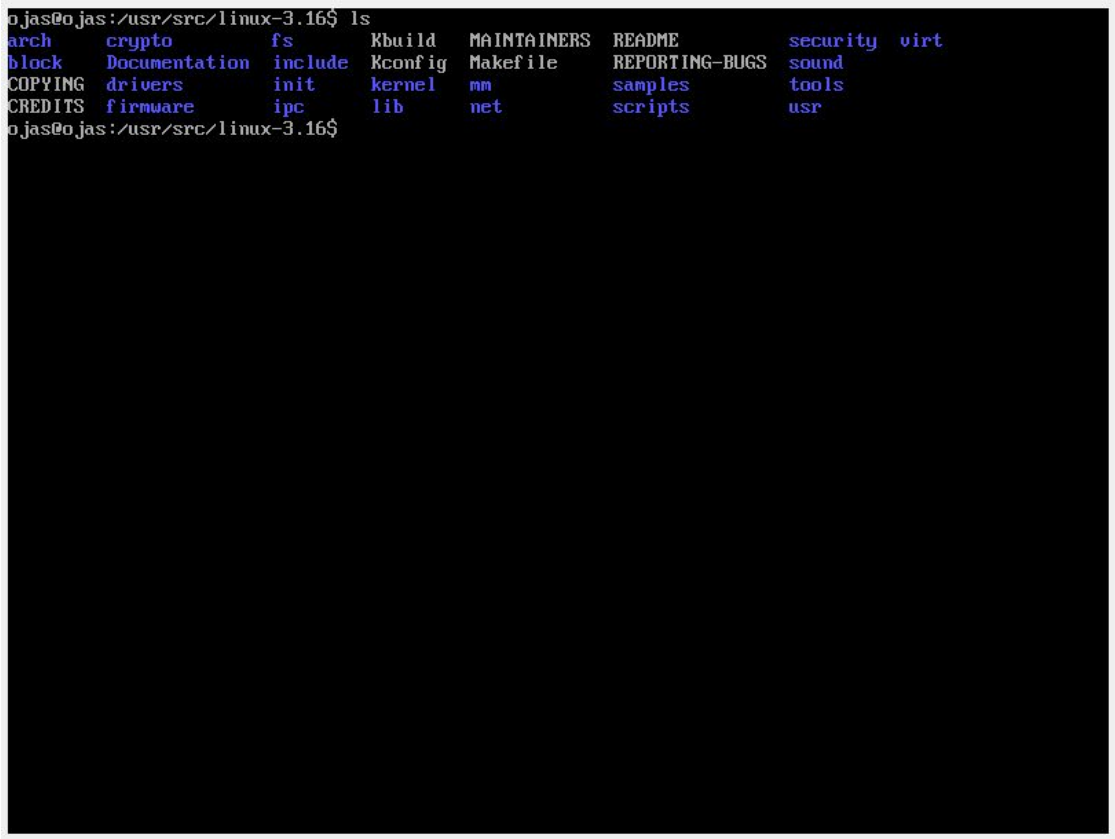
Assignment 1

Write-Up

Ojasva Saxena : 2018352

-> Download the linux-3.16 :
"wget https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.16.tar.xz"
and extract this downloaded file to /usr/src/ :
"sudo tar -xvf linux-3.16.tar.xz -c /usr/src/"



-> To make a new system call, inside this kernel version, add a new directory taskinfo/. In that, add a sh_task_info.c (the custom system call) and a Makefile.

```
root@zenbook:/usr/src/linux-3.16# ls
arch            firmware  Kconfig          modules.order   security          usr
block           fs        kernel           Module.symvers  signing_key.priv  virt
COPYING         hello     lib              net             signing_key.x509  vmlinux
CREDITS         include   MAINTAINERS      README          sound             vmlinux.o
crypto          init      Makefile         REPORTING-BUGS  System.map        x509.genkey
Documentation   ipc       mm               samples         taskinfo
drivers         Kbuild    modules.builtin  scripts         tools
root@zenbook:/usr/src/linux-3.16# cd taskinfo/
root@zenbook:/usr/src/linux-3.16/taskinfo# ls
Makefile  sh_task_info.c
root@zenbook:/usr/src/linux-3.16/taskinfo# _
```

```
 GNU nano 2.2.6                File: sh_task_info.c

                return -EINVAL;

        struct task_struct *task;
        struct file *file;
        loff_t pos = 0;
        int fileOpen , xVariable = 0;

        mm_segment_t old_fs = get_fs();
        set_fs(KERNEL_DS);

        fileOpen = sys_open(filename , O_WRONLY|O_CREAT , 0644);

        char data[500] , temp[500];
        int lines = 15;

        for_each_process(task)
        {
                if((int)task->pid == pid)
                {
                        xVariable = 3;

                        printk("Process Name: %s\n" , task->comm);
                        strcpy(data , "Process Name: ");
                        strcat(data , task->comm);
                        strcat(data , "\n");_

                        lines = lines + 1;

                        printk("Process PID: %ld\n" , (long)task->pid);
                        strcat(data , "Process PID: ");
                        sprintf(temp , "%ld\n" , (long)task->pid);
                        strcat(data , temp);

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

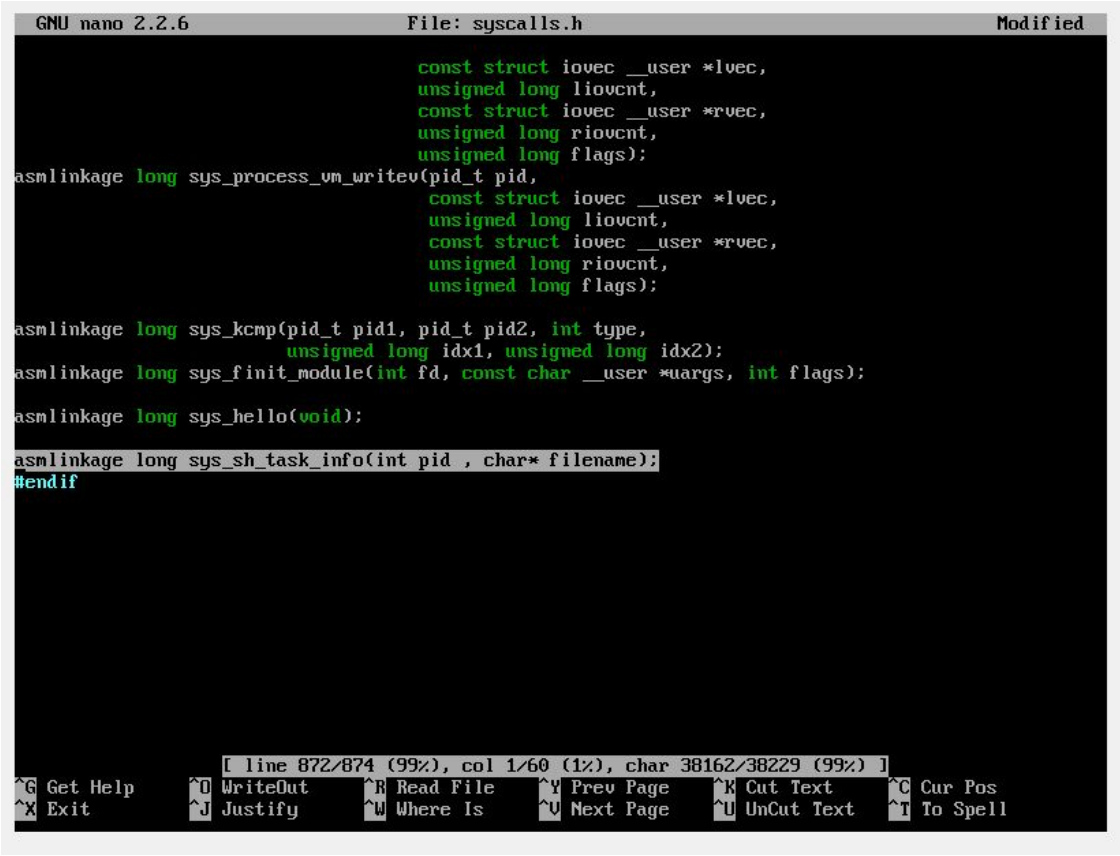
-> Now, in the main Makefile, add the new directory name

```
 GNU nano 2.2.6                File: Makefile                          Modified

mod_sign_cmd = perl $(srctree)/scripts/sign-file $(CONFIG_MODULE_SIG_HASH) $(MODSECKEY) $(MODPUBKEY)
else
mod_sign_cmd = true
endif
export mod_sign_cmd


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

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

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

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

# Externally visible symbols (used by link-vmlinux.sh)
export KBUILD_VMLINUX_INIT := $(head-y) $(init-y)
export KBUILD_VMLINUX_MAIN := $(core-y) $(libs-y) $(drivers-y) $(net-y)
export KBUILD_LDS          := arch/$(SRCARCH)/kernel/vmlinux.lds
export LDFLAGS_vmlinux
                [ line 842/1546 (54%), col 79/79 (100%), char 27562/52356 (52%) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

For 64 bit systems, add the new system call in serial order.

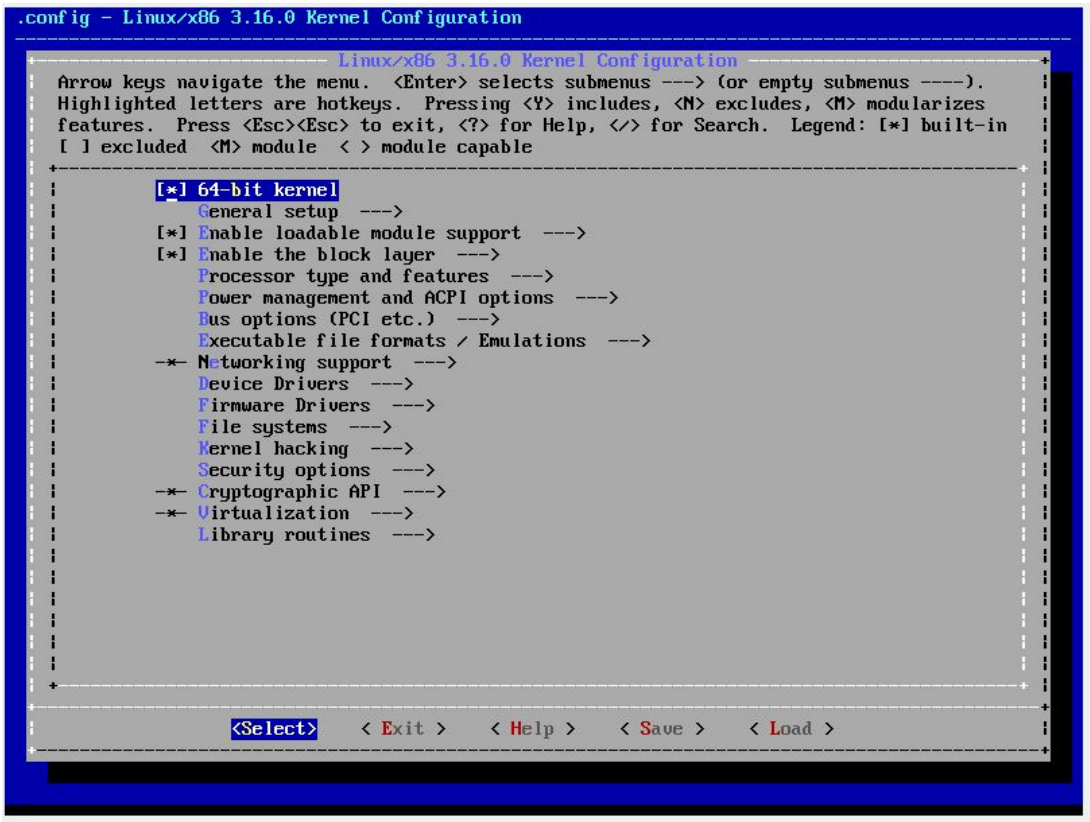
```
 GNU nano 2.2.6                File: syscall_64.tbl                    Modified

298     common  perf_event_open        sys_perf_event_open
299     64      recvmmsg                sys_recvmmsg
300     common  fanotify_init           sys_fanotify_init
301     common  fanotify_mark           sys_fanotify_mark
302     common  prlimit64               sys_prlimit64
303     common  name_to_handle_at       sys_name_to_handle_at
304     common  open_by_handle_at       sys_open_by_handle_at
305     common  clock_adjtime           sys_clock_adjtime
306     common  syncfs                  sys_syncfs
307     64      sendmmsg                sys_sendmmsg
308     common  setns                   sys_setns
309     common  getcpu                  sys_getcpu
310     64      process_vm_readv        sys_process_vm_readv
311     64      process_vm_writev       sys_process_vm_writev
312     common  kcmp                    sys_kcmp
313     common  finit_module            sys_finit_module
314     common  sched_setattr           sys_sched_setattr
315     common  sched_getattr           sys_sched_getattr
316     common  renameat2               sys_renameat2
317     64      hello                   sys_hello
318     64      sh_task_info            sys_sh_task_info

#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
512     x32     rt_sigaction            compat_sys_rt_sigaction
513     x32     rt_sigreturn            stub_x32_rt_sigreturn
514     x32     ioctl                   compat_sys_ioctl
515     x32     readv                   compat_sys_readv
516     x32     writev                  compat_sys_writev
517     x32     recvfrom                compat_sys_recvfrom

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

In include/linux/syscalls.h, prototype your new syscall.



-> Now the kernel is ready to be compiled.
First, do a *“sudo -s”* for root privileges.
Use :
“make menuconfig”
“make -j4”
“make modules_install -j4”
“make install”
To compile kernel.

Now, for updating grub, use :
“update-grub”




```
make[1]: *** [arch/x86/kernel] Interrupt
Makefile:1064: recipe for target 'arch/x86/modules.builtin' failed
make: *** [arch/x86/modules.builtin] Interrupt
scripts/Makefile.modbuiltin:54: recipe for target 'kernel/power' failed
make[1]: *** [kernel/power] Interrupt
Makefile:1064: recipe for target 'kernel/modules.builtin' failed
make: *** [kernel/modules.builtin] Interrupt

root@ojas:/usr/src/linux-3.16# make -j 4 && make modules_install
  SYSTBL  arch/x86/syscalls/../include/generated/asm/syscalls_64.h
  SYSTBL  arch/x86/syscalls/../include/generated/asm/syscalls_32.h
  CHK     include/config/kernel.release
  CHK     include/generated/uapi/linux/version.h
  HOSTCC  scripts/genksyms/genksyms.o
  SHIPPED scripts/genksyms/lex.lex.c
  SHIPPED scripts/genksyms/keywords.hash.c
  SHIPPED scripts/genksyms/parse.tab.h
  SHIPPED scripts/genksyms/parse.tab.c
  HOSTCC  scripts/genksyms/lex.lex.o
  SYSHDR  arch/x86/syscalls/../include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/syscalls/../include/generated/uapi/asm/unistd_64.h
  HOSTCC  scripts/genksyms/parse.tab.o
  SYSHDR  arch/x86/syscalls/../include/generated/uapi/asm/unistd_x32.h
  CC      scripts/mod/empty.o
  HOSTCC  scripts/mod/mk_elfconfig
  CC      scripts/mod/devicetable-offsets.s
  MKELF   scripts/mod/elfconfig.h
  GEN     scripts/mod/devicetable-offsets.h
  HOSTLD  scripts/genksyms/genksyms
  HOSTCC  scripts/mod/modpost.o
  HOSTCC  scripts/selinux/genheaders/genheaders
  HOSTCC  scripts/selinux/mdp/mdp
  HOSTCC  scripts/kallsyms
  HOSTCC  scripts/conmakehash
  HOSTCC  scripts/recordmcount
  HOSTCC  scripts/sortextable
```

-> Reboot the system and create a test file which invokes the newly created system call and run it.

```
root@zenbook:/usr/src/linux-3.16-UPDATED/taskinfo/ToTest# gcc test.c -o customSyscall
root@zenbook:/usr/src/linux-3.16-UPDATED/taskinfo/ToTest# ./customSyscall 3 info

        -X-
Task Struct Info Syscall Successful!
Check info file for details.
        -X-

root@zenbook:/usr/src/linux-3.16-UPDATED/taskinfo/ToTest# cat info
Process Name: ksoftirqd/0
Process PID: 3
Process Priority: 120
Process Parent Name: kthreadd
Process Exit State: 0
root@zenbook:/usr/src/linux-3.16-UPDATED/taskinfo/ToTest# ./customSyscall -3 info

        -X-
Something Went Wrong
Error : Invalid argument
Error No. : 22
        -X-

root@zenbook:/usr/src/linux-3.16-UPDATED/taskinfo/ToTest# _
```

To see the kernel messages, use *“dmesg”*.

```
[   17.902189] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[   18.063612] init: failsafe main process (606) killed by TERM signal
[   18.119353] audit: type=1400 audit(1569573340.320:5): apparmor="STATUS" operation="profile_replac
e" name="/sbin/dhclient" pid=813 comm="apparmor_parser"
[   18.119359] audit: type=1400 audit(1569573340.320:6): apparmor="STATUS" operation="profile_replac
e" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=813 comm="apparmor_parser"
[   18.119362] audit: type=1400 audit(1569573340.320:7): apparmor="STATUS" operation="profile_replac
e" name="/usr/lib/connman/scripts/dhclient-script" pid=813 comm="apparmor_parser"
[   18.120667] audit: type=1400 audit(1569573340.320:8): apparmor="STATUS" operation="profile_load"
name="/usr/sbin/tcpdump" pid=813 comm="apparmor_parser"
[   18.610266] init: plymouth-upstart-bridge main process ended, respawning
[   20.324849] floppy0: no floppy controllers found
[   20.324879] work still pending
[   24.617279] systemd-logind[1244]: New seat seat0.
[   24.618074] systemd-logind[1244]: Watching system buttons on /dev/input/event0 (Power Button)
[   24.618118] systemd-logind[1244]: Watching system buttons on /dev/input/event1 (Sleep Button)
[   24.618159] systemd-logind[1244]: Watching system buttons on /dev/input/event5 (Video Bus)
[   24.623132] systemd-logind[1244]: Failed to start unit user@1000.service: Unknown unit: user@1000
.service
[   24.623168] systemd-logind[1244]: Failed to start user service: Unknown unit: user@1000.service
[   24.627742] systemd-logind[1244]: New session c1 of user ojas.
[  169.624158] Process Name: rcuos/0
[  169.624162] Process PID: 8
[  169.624164] Process Priority: 120
[  169.624166] Process Parent Name: kthreadd
[  169.624168] Process Exit State : 0
[  483.271133] Process Name: rcuos/0
[  483.271137] Process PID: 8
[  483.271140] Process Priority: 120
[  483.271143] Process Parent Name: kthreadd
[  483.271146] Process Exit State : 0
[  501.732600] Process Name: ksoftirqd/0
[  501.732607] Process PID: 3
[  501.732610] Process Priority: 120
[  501.732613] Process Parent Name: kthreadd
[  501.732616] Process Exit State : 0
root@zenbook:/usr/src/linux-3.16-UPDATED/taskinfo/ToTest# _
```

-> Create a diff file of the updated and original kernel to show the differences, and store it in a .patch file using :
“diff -urN linux-3.16-UPDATED/ linux-3.16-ORG/ > diffFile.patch”

Code Summary

- The system call takes in 2 arguments at the beginning of runtime : the pid (the process ID integer) and the filename (the file in which the output is to be saved), which has to be supplied by the user as input.
- In case of an invalid PID, the 'Invalid Argument' error is returned (EINVAL).
- A task struct, a file with the required filename is initialised.
- For filing in kernel space, the following resource was used : https://www.linuxjournal.com/article/8110
- sys_open is used to open the specified file supplied.
- The character data to be entered in file was declared.
- If the file was unable to be opened, an 'Is a Directory' error was flagged (EISDIR).
- The for_each_process(task) loop will loop through all the tasks (processes/threads) and check for the one that matches the required PID.
- Then, the important information was stored and printed on the kernel using printk() :
    - Process Name (comm)
      Shows the name of the task whose PID was queried.
    - Process PID (pid)
      Displays the PID of the task.
    - Process Parent Name (*parent->comm)
      Displays the name of the parent of the task which was queried.
    - Process Exit State (exit_code)
      Shows the returned code of the task when it is exited.
- The stored data was then written to the successfully opened file.
- The file is closed.