

OS ASSIGNMENT

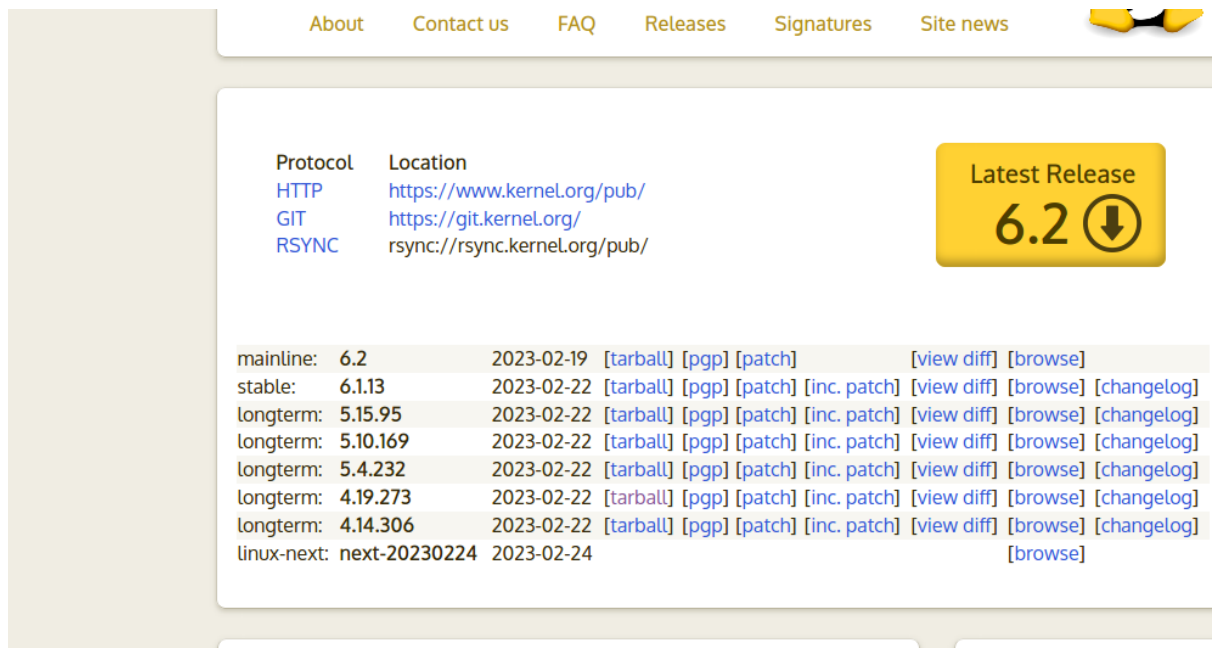
Adding a new System Call to your Kernel

1- Initially, I added used these commands to provide an environment for OS kernel to work

Prerequisites:

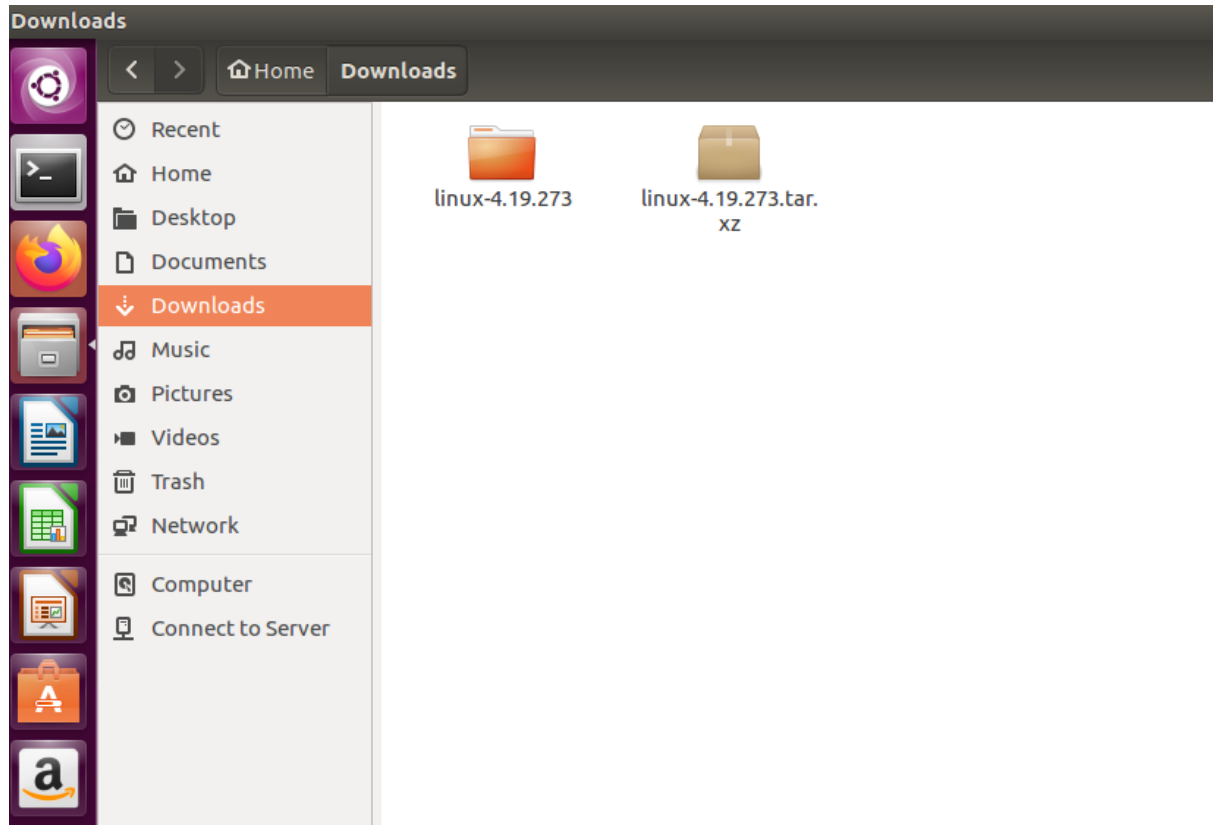
- `sudo apt-get install gcc`
- `sudo apt-get install libncurses5-dev`
- `sudo apt-get install bison`
- `sudo apt-get install flex`
- `sudo apt install make`
- `sudo apt-get install libssl-dev`
- `sudo apt-get install libelf-dev`
- `sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) main universe"`
- `sudo apt-get update`
- `sudo apt-get upgrade`

2- After adding those commands I downloaded kernel version to match my OS version which was 4.19.273



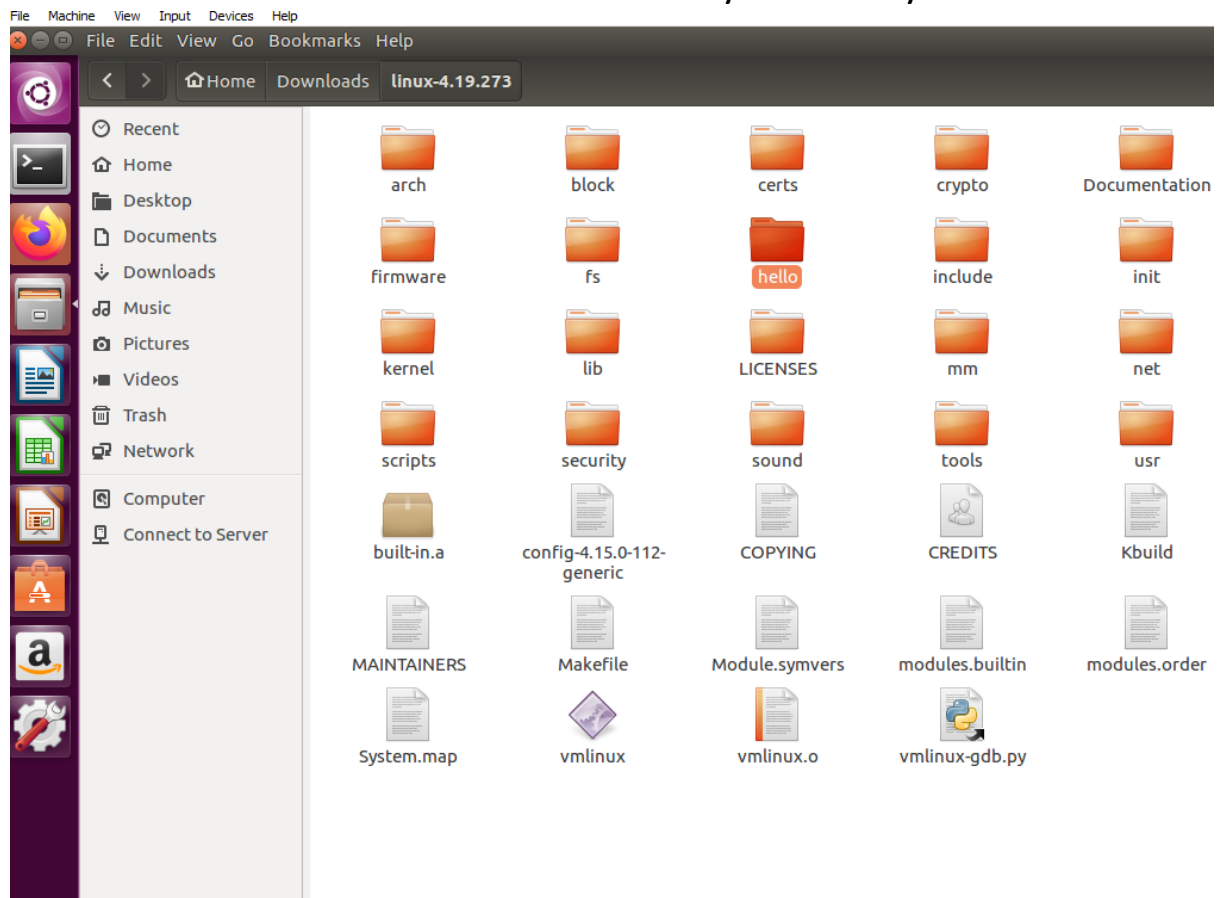
Click on tarball and click save the file to download it.

3- Now, go to the folder where the kernel is downloaded and extract the folder.



After extraction one new folder will appear.

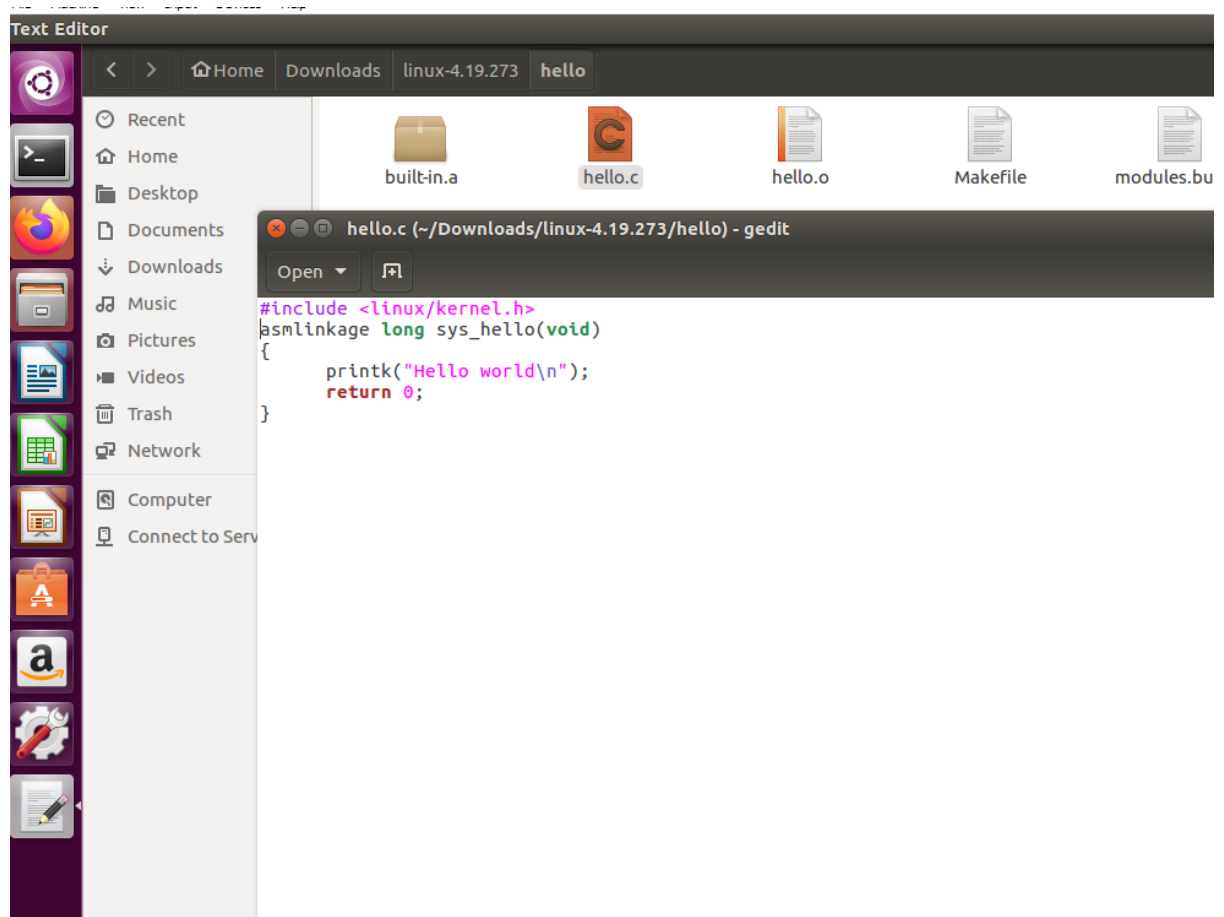
- 4- Go into the folder where you extracted the kernel and go inside the kernel's folder and make a new directory named by hello



- 5- After creating the folder, Now, go to the folder which we created just now and open the terminal by clicking the right side of the mouse there and create a new C code file by typing “gedit hello.c” and paste the following code there:

```
#include <linux/kernel.h>
asmlinkage long sys_hello(void)
{
    printk("Hello world\n");
    return 0;
}
```

The above code will invoke the hello function to call after we the custom kernel is done.

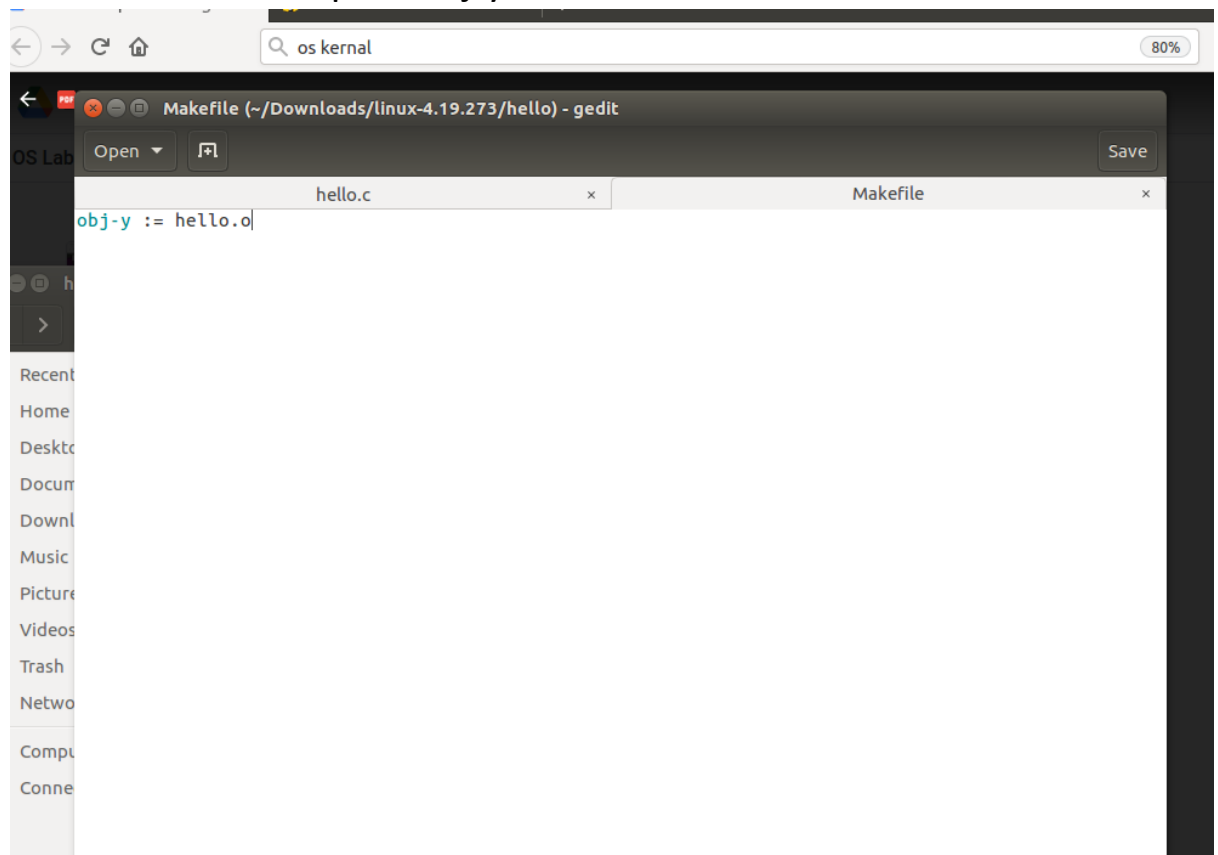


Code explanation:

- a. We used `#include <linux/kernel.h>` because we are building a system call for our linux kernel.
- b. `asmmlinkage` simply means that the arguments for this function will be on the stack instead of the CPU registers.
- c. `Printk` is used instead of `printf` because we are going to print in the kernel's log file.
- d. If the code is run and it returns 0, then it will mean that our program ran successfully and Hello world is written to out kernel's log file.

6- Now, we have to create a Makefile for our new folder to ensure that the code in the folder is always compiled whenever the

kernel is compiled. In order to do this, we type in our terminal “gedit Makefile” and put “obj-y := hello.o” command.



7- Adding the new code into the system table file:

Since we are creating a 64-bit system call according to our system we have to add the system call entry into the `syscall_64.tbl` file which keeps the name of all the system calls in our system. If our system was a 32-bit system, we would have to add our system call into `syscall_32.tbl` (We can check the type of our system by typing “`uname -m`” in a terminal). This `tbl` file is located inside the kernel folder in `/arch/x86/entry/syscalls/syscall_64.tbl`. We can go into this directory by using `cd` and then edit the file by typing “`gedit syscall_64.tbl`”.

The image shows a terminal window and a gedit editor window. The terminal window shows the user navigating through directories and editing files. The gedit window shows the contents of the file `syscall_64.tbl`, which lists various system calls and their corresponding kernel functions.

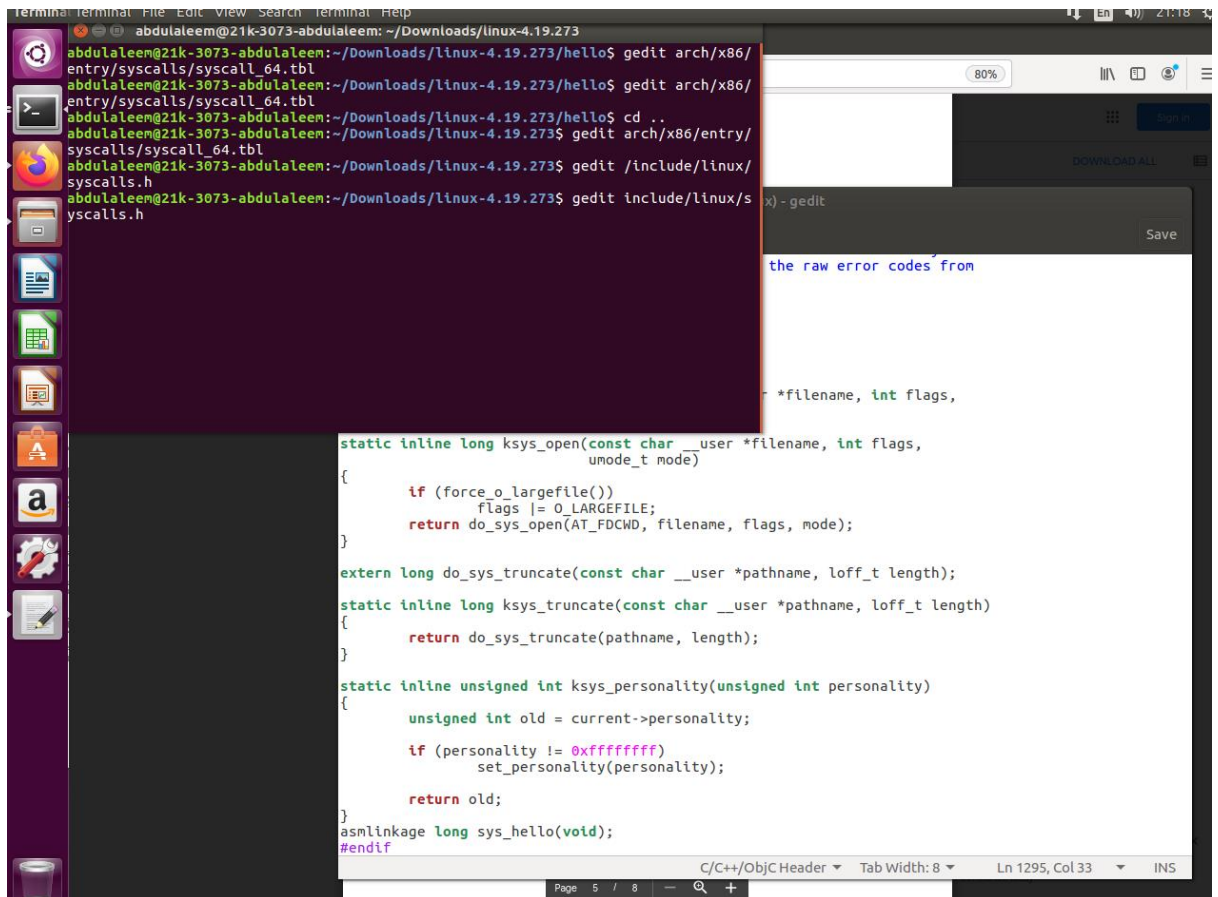
```
abdulaleem@21k-3073-abdulaleem: ~/Downloads/linux-4.19.273
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ cd ..
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit arch/x86/entry/syscalls/syscall_64.tbl
```

The gedit window shows the following content:

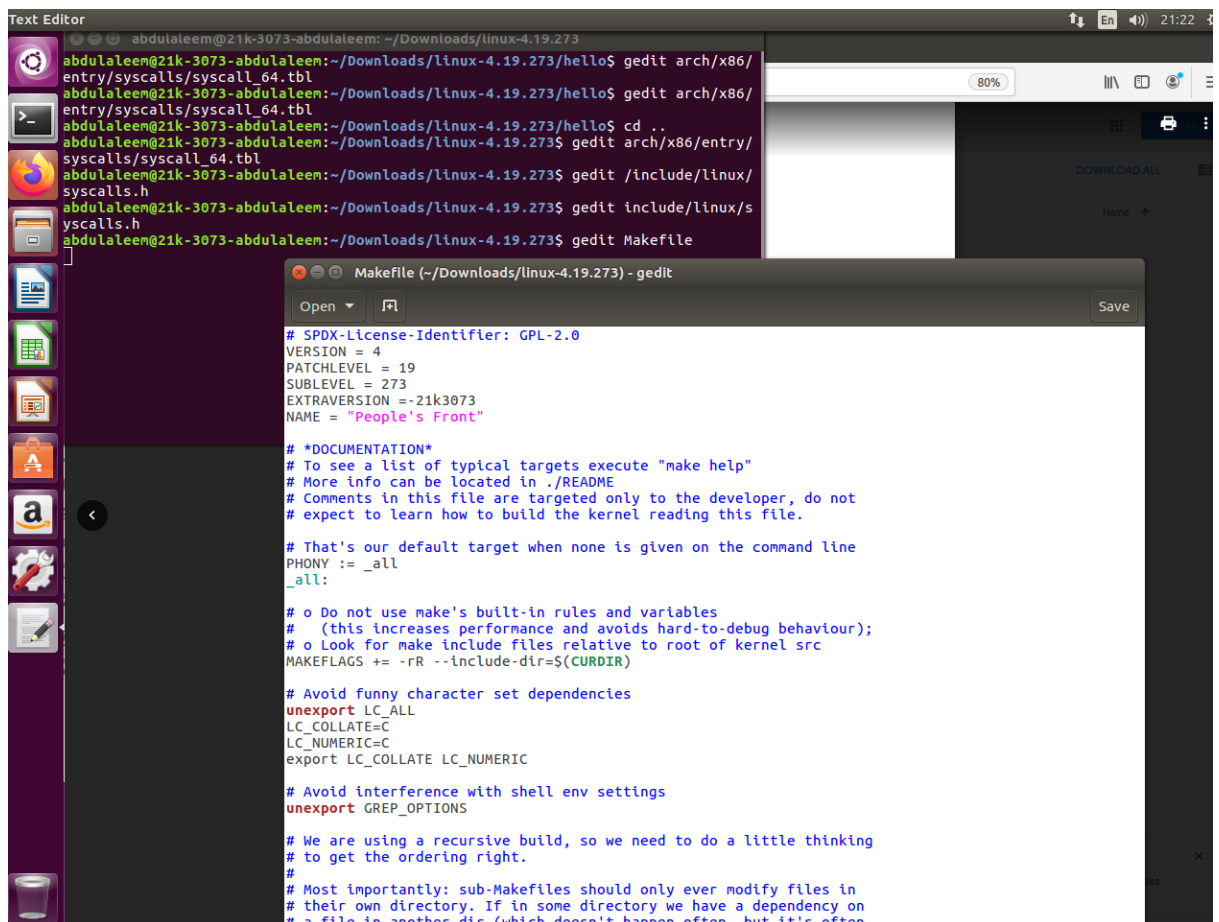
Line	Number	Function	Kernel Function
310	64	process_vm_readv	__x64_sys_process_vm_readv
311	64	process_vm_writev	__x64_sys_process_vm_writev
312	common	kcmp	__x64_sys_kcmp
313	common	fininit_module	__x64_sys_finit_module
314	common	sched_setattr	__x64_sys_sched_setattr
315	common	sched_getattr	__x64_sys_sched_getattr
316	common	renameat2	__x64_sys_renameat2
317	common	seccomp	__x64_sys_seccomp
318	common	getrandom	__x64_sys_getrandom
319	common	memfd_create	__x64_sys_memfd_create
320	common	kexec_file_load	__x64_sys_kexec_file_load
321	common	bpf	__x64_sys_bpf
322	64	execveat	__x64_sys_execveat/ptregs
323	common	userfaultfd	__x64_sys_userfaultfd
324	common	membarrier	__x64_sys_membarrier
325	common	mlock2	__x64_sys_mlock2
326	common	copy_file_range	__x64_sys_copy_file_range
327	64	preadv2	__x64_sys_preadv2
328	64	pwritev2	__x64_sys_pwritev2
329	common	pkey_mprotect	__x64_sys_pkey_mprotect
330	common	pkey_alloc	__x64_sys_pkey_alloc
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
335	64	hello	sys_hello

8- Now we have to add the prototype of our system call in the system's header file which is located in the kernel folder then `"/include/linux/syscalls.h"`. We have to add the prototype of our system call function in this file.

In the end, we will type the function of hello world that we have created it in the `hello.c` file and paste that function in the end above `endif` and after pasting the code put a `;"` this sign otherwise in the end it will cause errors.



9- Now, we have to add our roll number in the extraversion of the kernel's make file type "gedit Makefile" and write your roll number with "--" sign and we have to add the new module that we created into our kernel's make file.



```
abdulaleem@21k-3073-abdulaleem: ~/Downloads/linux-4.19.273
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ cd ..
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit /include/linux/syscalls.h
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit include/linux/syscalls.h
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit Makefile

# SPDX-License-Identifier: GPL-2.0
VERSION = 4
PATCHLEVEL = 19
SUBLEVEL = 273
EXTRAVERSION = -21k3073
NAME = "People's Front"

# *DOCUMENTATION*
# To see a list of typical targets execute "make help"
# More info can be located in ./README
# Comments in this file are targeted only to the developer, do not
# expect to learn how to build the kernel reading this file.

# That's our default target when none is given on the command line
PHONY := _all
_all:

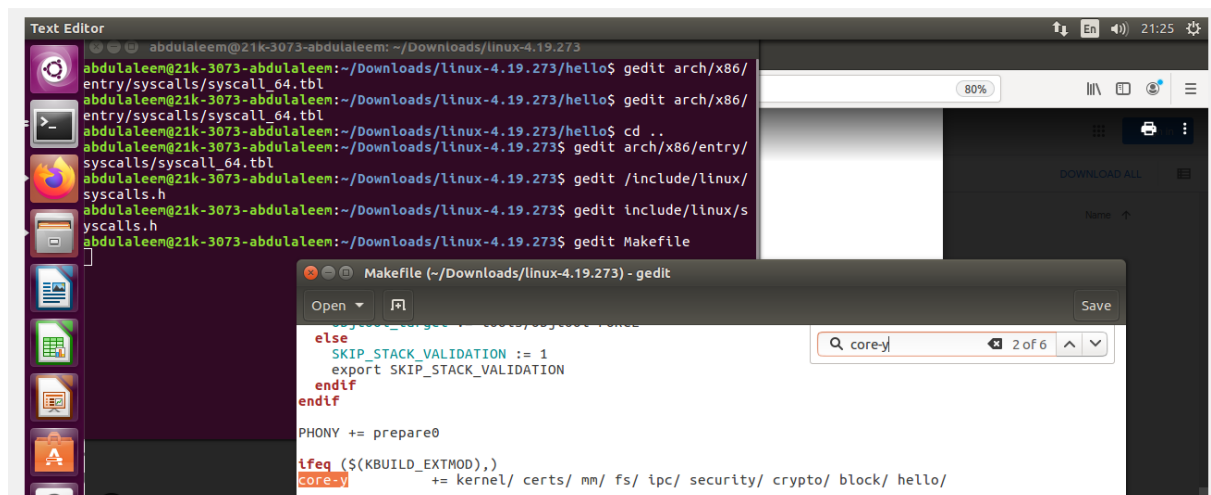
# Do not use make's built-in rules and variables
# (this increases performance and avoids hard-to-debug behaviour);
# o Look for make include files relative to root of kernel src
MAKEFLAGS += -rR --include-dir=$(CURDIR)

# Avoid funny character set dependencies
unexport LC_ALL
LC_COLLATE=C
LC_NUMERIC=C
export LC_COLLATE LC_NUMERIC

# Avoid interference with shell env settings
unexport GREP_OPTIONS

# We are using a recursive build, so we need to do a little thinking
# to get the ordering right.
#
# Most importantly: sub-Makefiles should only ever modify files in
# their own directory. If in some directory we have a dependency on
# a file in another dir (which doesn't happen often, but it's often
```

For this, we open the Makefile of the kernel and search for “core-y” and go to it’s second instance which is under “KBUILD_EXTMOD” and add our new module which is “hello” at the end of it. At the end, our make file will look something like this:



```
abdulaleem@21k-3073-abdulaleem: ~/Downloads/linux-4.19.273
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ cd ..
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit /include/linux/syscalls.h
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit include/linux/syscalls.h
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit Makefile

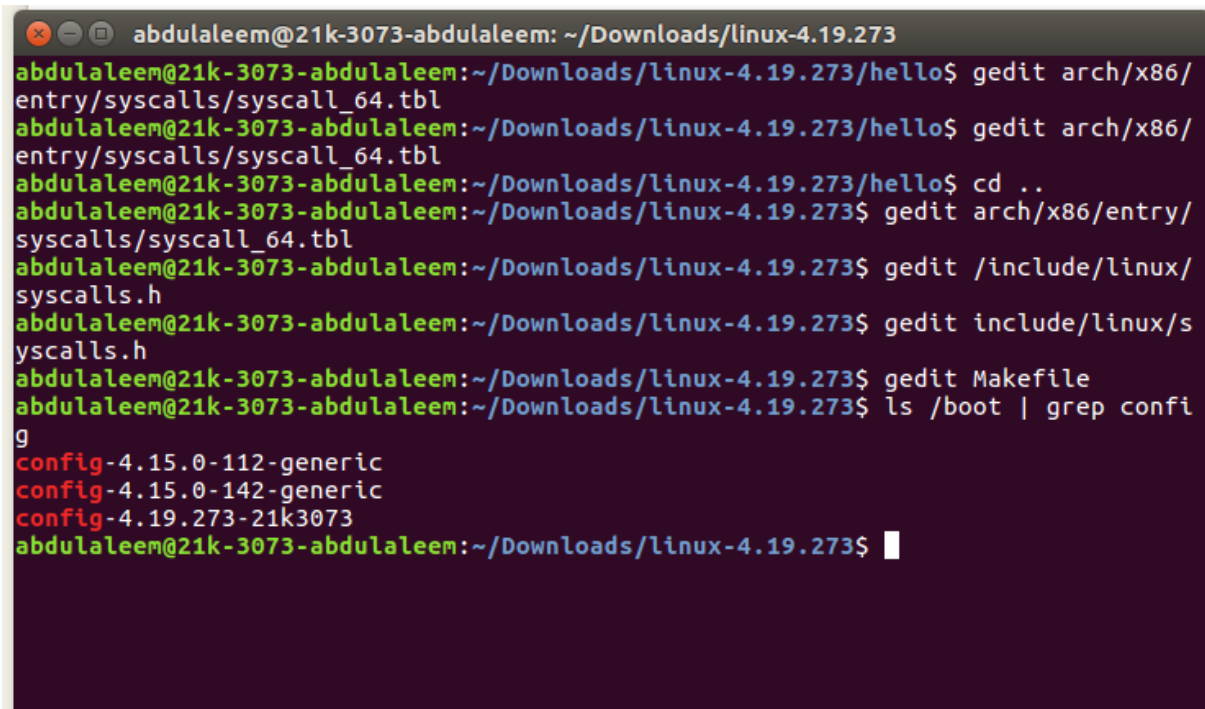
else
    SKIP_STACK_VALIDATION := 1
    export SKIP_STACK_VALIDATION
endif
endif

PHONY += prepare0

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


10- Creating a config file:

Now we have to create a config file for our kernel. The order of the steps before this can change but the order of this step and the steps coming right after it can't change. We can either create a Menuconfig or simply copy the oldconfig. I will be copying the oldconfig and using that config for my new kernel. First of all, we search for the config that we currently have by typing "ls /boot | grep config"

A terminal window with a dark background and light-colored text. The window title is 'abdulaleem@21k-3073-abdulaleem: ~/Downloads/linux-4.19.273'. The terminal shows a series of commands and their outputs. The final command executed is 'ls /boot | grep config', which returns three lines of output: 'config-4.15.0-112-generic', 'config-4.15.0-142-generic', and 'config-4.19.273-21k3073'. The prompt for the next command is visible at the bottom.

```
abdulaleem@21k-3073-abdulaleem: ~/Downloads/linux-4.19.273
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273/hello$ cd ..
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit arch/x86/entry/syscalls/syscall_64.tbl
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit /include/linux/syscalls.h
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit include/linux/syscalls.h
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ gedit Makefile
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ ls /boot | grep config
config-4.15.0-112-generic
config-4.15.0-142-generic
config-4.19.273-21k3073
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$
```

and then we copy the config that is shown to us by typing "cp /boot/config-4.10.0-28-generic *our linux kernel directory*". First add 'pwd' command to ender inside the folder.

```

abdualeem@21k-3073-abdualeem: ~/Downloads/linux-4.19.273
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273/hello$ cd ..
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ gedit arch/x86/entry/
syscalls/syscall_64.tbl
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ gedit /include/linux/
syscalls.h
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ gedit include/linux/s
yscalls.h
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ gedit Makefile
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ ls /boot | grep confi
g
config-4.15.0-112-generic
config-4.15.0-142-generic
config-4.19.273-21k3073
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ cp /boot/config-4.15.
0-112-generic
cp: missing destination file operand after '/boot/config-4.15.0-112-generic'
Try 'cp --help' for more information.
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ pwd
/home/abdualeem/Downloads/linux-4.19.273
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ cp /boot/config-4.15.
0-112-generic /home/abdualeem/Downloads/linux-4.19.273
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ cp /boot/config-4.15.
0-112-generic /home/abdualeem/Downloads/linux-4.19.273
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$

```

Then we create the old config by typing “yes "" | make oldconfig -j4”, by doing so, the system will automatically create the new config for us and select the default option for everything.

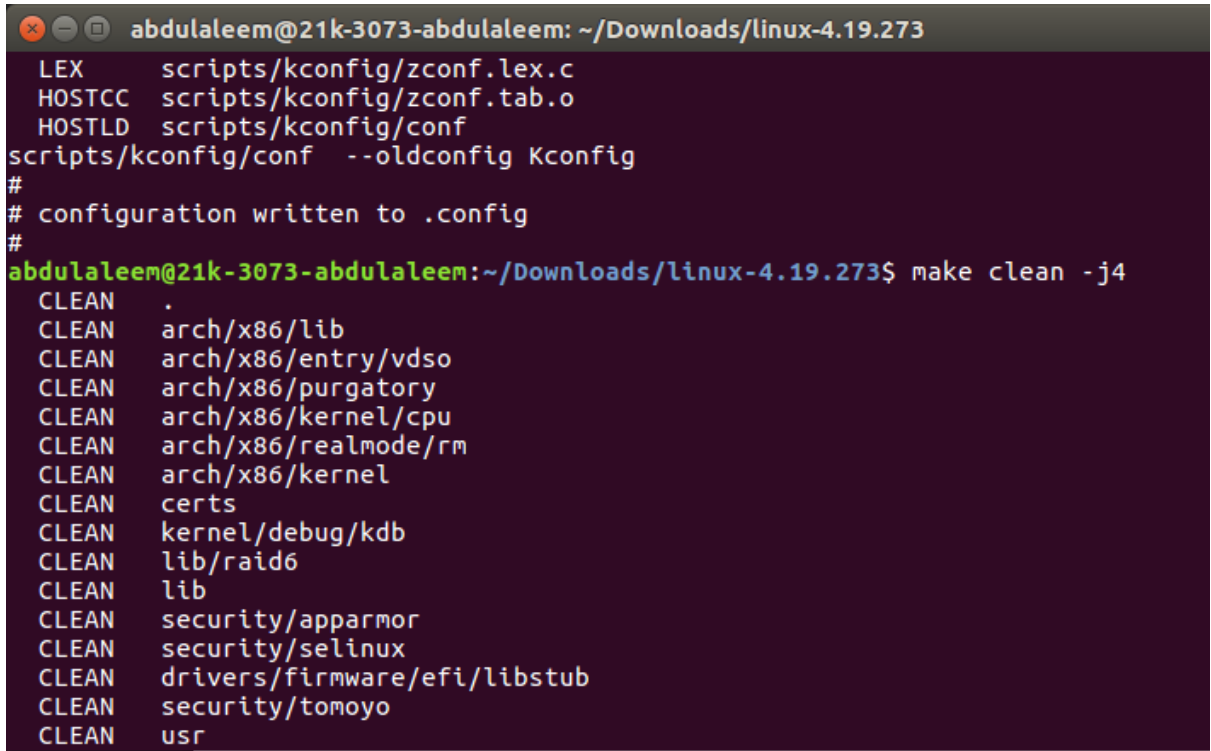
```

abdualeem@21k-3073-abdualeem: ~/Downloads/linux-4.19.273
config-4.15.0-142-generic
config-4.19.273-21k3073
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ cp /boot/config-4.15.
0-112-generic
cp: missing destination file operand after '/boot/config-4.15.0-112-generic'
Try 'cp --help' for more information.
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ pwd
/home/abdualeem/Downloads/linux-4.19.273
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ cp /boot/config-4.15.
0-112-generic /home/abdualeem/Downloads/linux-4.19.273
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ cp /boot/config-4.15.
0-112-generic /home/abdualeem/Downloads/linux-4.19.273
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ yes "" | make oldconf
ig -j4
HOSTCC scripts/kconfig/conf.o
YACC scripts/kconfig/zconf.tab.c
LEX scripts/kconfig/zconf.lex.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --oldconfig Kconfig
#
# configuration written to .config
#
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$

```

11- 10. Cleaning and Compiling the kernel:

We have to clean all of our existing object and executable file because compiler sometimes link or compile files incorrectly and to avoid this, we delete all of our old object and executable files by typing “make clean -j4” (It is better to switch to super user mode by typing “sudo su” before running the commands after this)

A terminal window with a dark purple background. The title bar shows the user 'abdulaleem' at host '21k-3073-abdulaleem' in the directory '~/Downloads/linux-4.19.273'. The terminal output shows the configuration of Kconfig files, followed by the command 'make clean -j4'. The output of this command lists various directories and files that have been cleaned, including 'arch/x86/lib', 'arch/x86/entry/vdso', 'arch/x86/purgatory', 'arch/x86/kernel/cpu', 'arch/x86/realmode/rm', 'arch/x86/kernel', 'certs', 'kernel/debug/kdb', 'lib/raid6', 'lib', 'security/apparmor', 'security/selinux', 'drivers/firmware/efi/libstub', 'security/tomoyo', and 'usr'.

```
abdulaleem@21k-3073-abdulaleem: ~/Downloads/linux-4.19.273
LEX      scripts/kconfig/zconf.lex.c
HOSTCC   scripts/kconfig/zconf.tab.o
HOSTLD   scripts/kconfig/conf
scripts/kconfig/conf  --oldconfig Kconfig
#
# configuration written to .config
#
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ make clean -j4
CLEAN    .
CLEAN    arch/x86/lib
CLEAN    arch/x86/entry/vdso
CLEAN    arch/x86/purgatory
CLEAN    arch/x86/kernel/cpu
CLEAN    arch/x86/realmode/rm
CLEAN    arch/x86/kernel
CLEAN    certs
CLEAN    kernel/debug/kdb
CLEAN    lib/raid6
CLEAN    lib
CLEAN    security/apparmor
CLEAN    security/selinux
CLEAN    drivers/firmware/efi/libstub
CLEAN    security/tomoyo
CLEAN    usr
```

this) and when this all is done, we type “make -j4” to start building our kernel (-j4 allocates the multiple cores that our system have for compiling. In my case I have used make -j8 cuz I allocated 8 cosres to my cpu and you can check it out by using lscpu for your

cores.

```
abdualeem@21k-3073-abdualeem: ~/Downloads/linux-4.19.273
CLEAN drivers/tty/vt
CLEAN .tmp_versions
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 8
On-line CPU(s) list:   0-7
Thread(s) per core:     1
Core(s) per socket:    8
Socket(s):              1
NUMA node(s):          1
Vendor ID:              GenuineIntel
CPU family:             6
Model:                 158
Model name:             Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz
Stepping:               13
CPU MHz:               2904.000
BogoMIPS:               5808.00
Hypervisor vendor:     KVM
Virtualization type:    full
L1d cache:              32K
L1i cache:              32K
```

after checking it out do this.

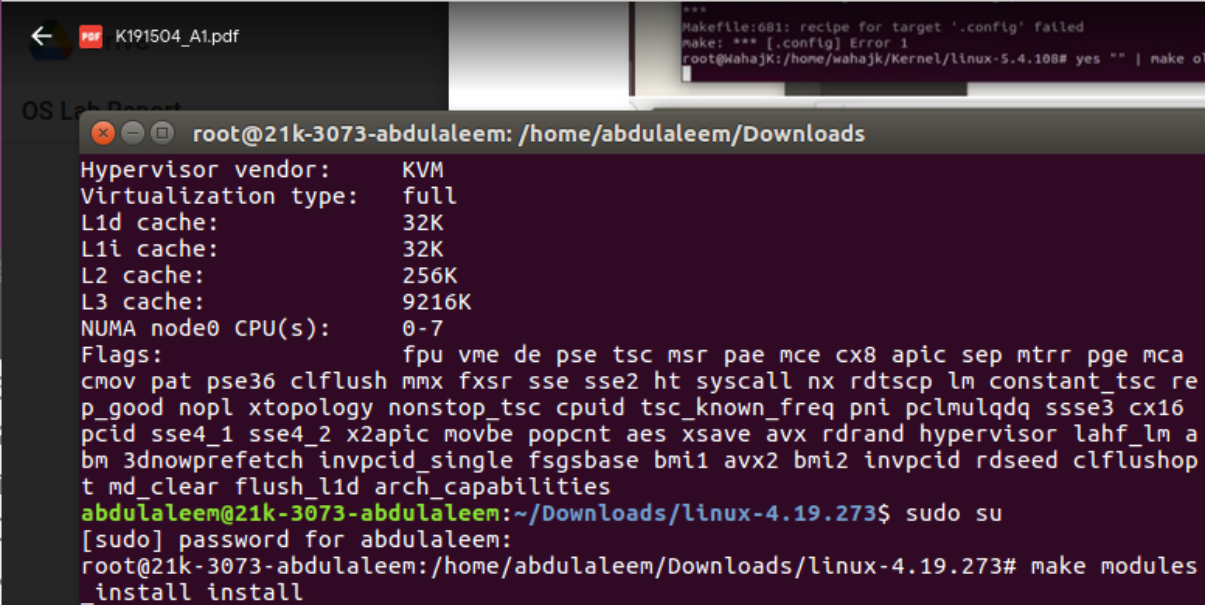
```
abdualeem@21k-3073-abdualeem: ~/Downloads/linux-4.19.273
Core(s) per socket:    8
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                158
Model name:            Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz
Stepping:              13
CPU MHz:              2904.000
BogoMIPS:              5808.00
Hypervisor vendor:     KVM
Virtualization type:    full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              9216K
NUMA node0 CPU(s):    0-7
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx rdtscp lm constant_tsc re
o_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni pclmulqdq sse3 cx16
ocid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx rdrand hypervisor lahf_lm a
om 3dnowprefetch invpcid_single fsgsbase bmi1 avx2 bmi2 invpcid rdseed clflushop
t md_clear flush_l1d arch_capabilities
abdualeem@21k-3073-abdualeem:~/Downloads/linux-4.19.273$ make -j8
```

Now we have to wait until our Kernel image is built and ready. If we see “Kernel image is

ready” when the command is done executing, that means that our kernel image is ready to be installed.

12- Installing modules:

Now we have to install the kernel that we built by typing “make modules_install install” which will install the kernel and update our grub as well. When this all is done and the terminal says “done”, then we can restart our laptop either manually or by typing “shutdown -r now” and hold the “Shift” key while it is restarting to open up the grub menu and switch to the new kernel which we just installed. (I forgot to take screenshot of this point as well.)



```
root@21k-3073-abdulaleem: /home/abdulaleem/Downloads
Hypervisor vendor:      KVM
Virtualization type:    full
L1d cache:              32K
L1i cache:              32K
L2 cache:               256K
L3 cache:               9216K
NUMA node0 CPU(s):     0-7
Flags:                  fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx rdtscp lm constant_tsc re
p_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni pclmulqdq ssse3 cx16
pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx rdrand hypervisor lahf_lm a
bm 3dnowprefetch invpcid_single fsgsbase bmi1 avx2 bmi2 invpcid rdseed clflushop
t md_clear flush_l1d arch_capabilities
abdulaleem@21k-3073-abdulaleem:~/Downloads/linux-4.19.273$ sudo su
[sudo] password for abdulaleem:
root@21k-3073-abdulaleem:/home/abdulaleem/Downloads/linux-4.19.273# make modules
_install install
```

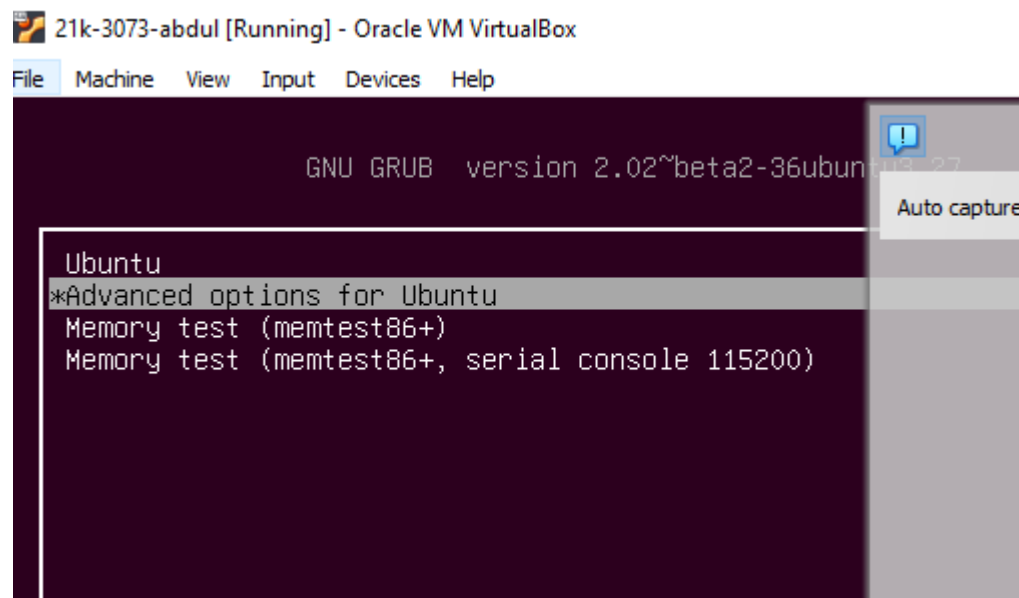
13- Checking if the System call is Working Properly:

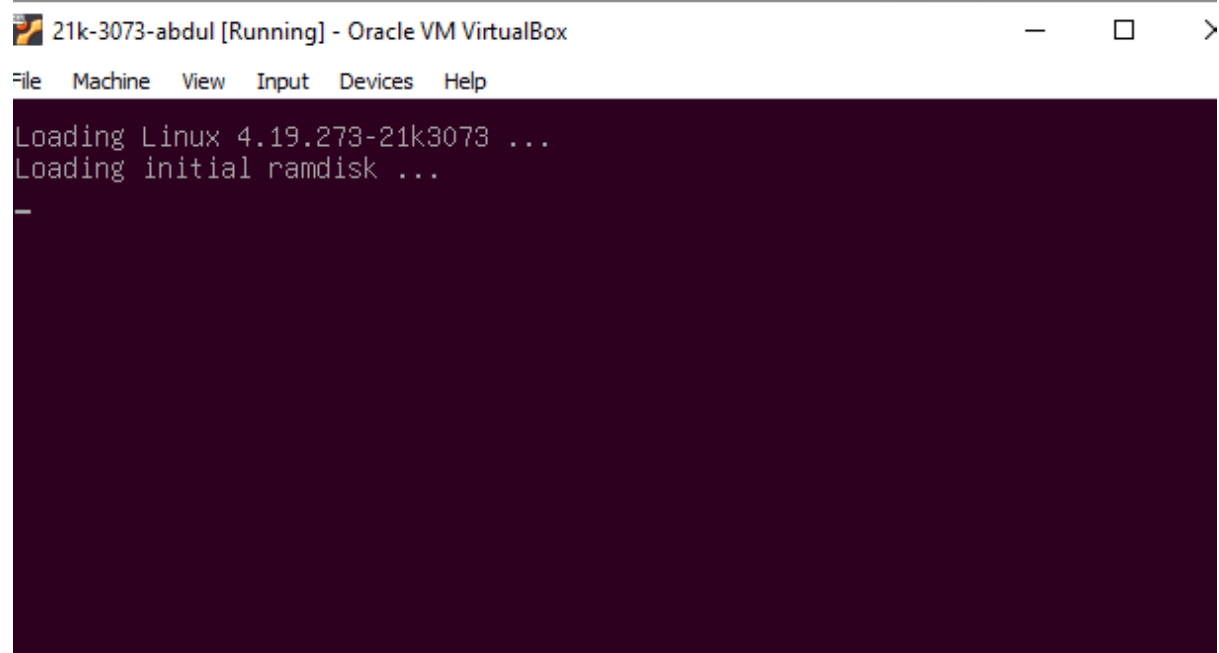
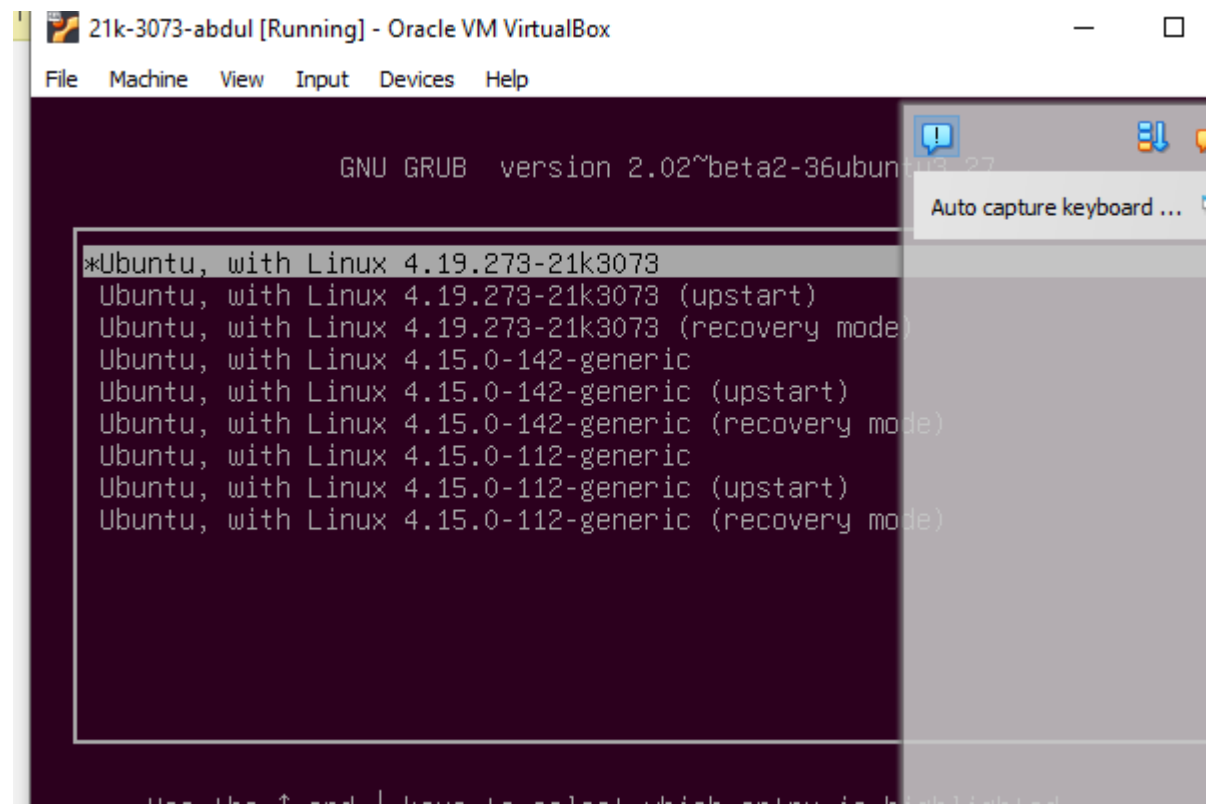
After logging into the newly compiled kernel, we check the system call by making a C code

named “userspace.c” and putting the following code in it:

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
int main()
{
long int i = syscall(335);
printf("System call sys_hello returned %ld\n", i);
return 0;
}
```

Now we compile the code by typing “gcc userspace.c” and executing it by typing “./a.out”. If it returns 0, this means that our code has compiled successfully and the system call is working fine (Note that in calling syscall(335), 335 is the number where we added our system call in the table) and finally, we run “dmesg” to see the kernel messages and we will find “Hello World” written at the end of it.





21k-3073-abdul [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminal

```
abdulaleem@21k-3073-abdulaleem: ~  
abdulaleem@21k-3073-abdulaleem:~$ uname -r  
4.19.273-21k3073  
abdulaleem@21k-3073-abdulaleem:~$
```

Text Editor

```
abdulaleem@21k-3073-abdulaleem: ~  
abdulaleem@21k-3073-abdulaleem:~$ uname -r  
4.19.273-21k3073  
abdulaleem@21k-3073-abdulaleem:~$ gedit userspace.c
```

userspace.c (~/) - gedit

```
Open [icon]  
#include <stdio.h>  
#include <linux/kernel.h>  
#include <sys/syscall.h>  
#include <unistd.h>  
int main()  
{  
    long int i = syscall(335);  
    printf("System call sys_hello returned %ld\n", i);  
    return 0;  
}
```


21k-3073-abdul [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Terminal

```
abdulaleem@21k-3073-abdulaleem: ~
abdulaleem@21k-3073-abdulaleem:~$ uname -r
4.19.273-21k3073
abdulaleem@21k-3073-abdulaleem:~$ gedit userspace.c
abdulaleem@21k-3073-abdulaleem:~$ ./a.out
System call sys_hello returned 0
abdulaleem@21k-3073-abdulaleem:~$
```

Terminal

```
abdulaleem@21k-3073-abdulaleem: ~
[ 22.591240] audit: type=1400 audit(1677257694.200:7): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/usr/sbin/cups-browsed" pid=584 comm="apparmor_parser"
[ 22.592063] audit: type=1400 audit(1677257694.200:8): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/usr/sbin/tcpdump" pid=588 comm="apparmor_parser"
[ 22.597561] audit: type=1400 audit(1677257694.204:9): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/sbin/dhclient" pid=579 comm="apparmor_parser"
[ 22.597571] audit: type=1400 audit(1677257694.204:10): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-client.action" pid=579 comm="apparmor_parser"
[ 22.597577] audit: type=1400 audit(1677257694.204:11): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/usr/lib/NetworkManager/nm-dhcp-helper" pid=579 comm="apparmor_parser"
[ 24.742966] Adding 999420k swap on /dev/sda5. Priority:-2 extents:1 across:999420k FS
[ 35.822335] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ 35.828859] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[ 35.837374] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 35.837924] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[ 211.199472] Hello world
abdulaleem@21k-3073-abdulaleem:~$
```

DONE!