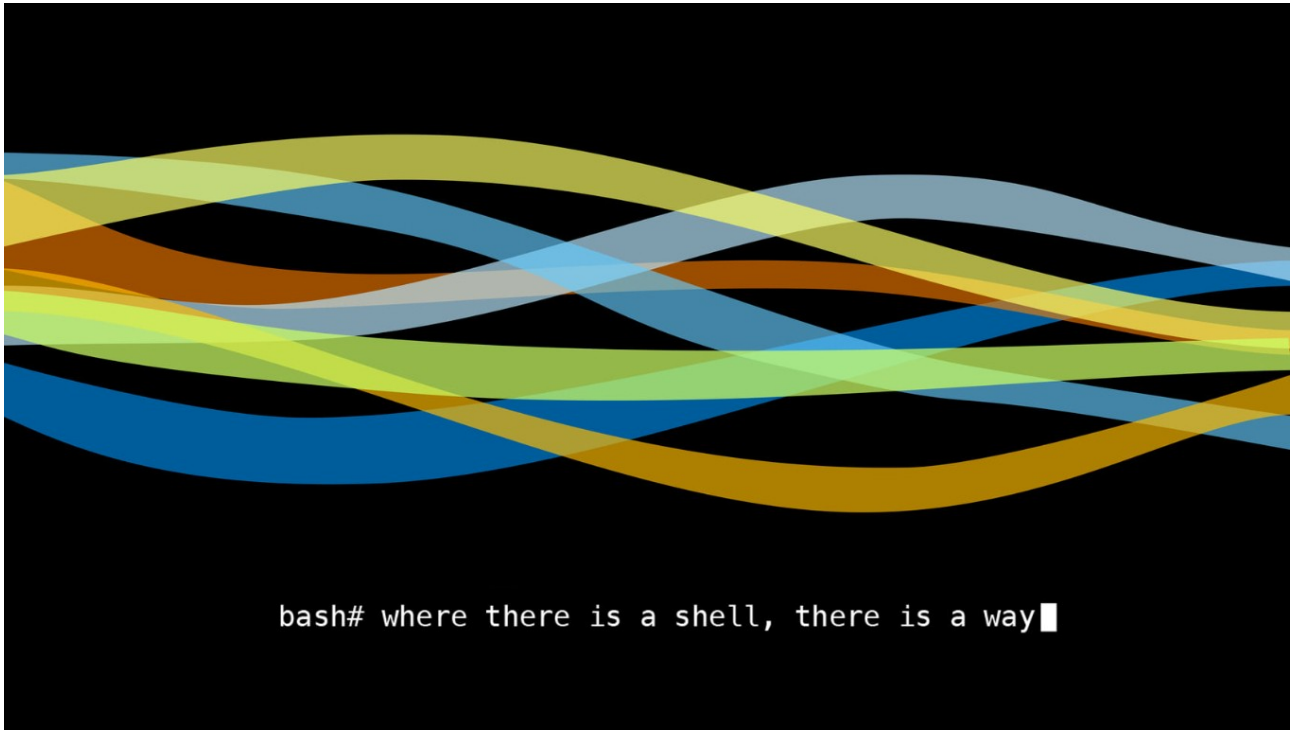
Operating Systems

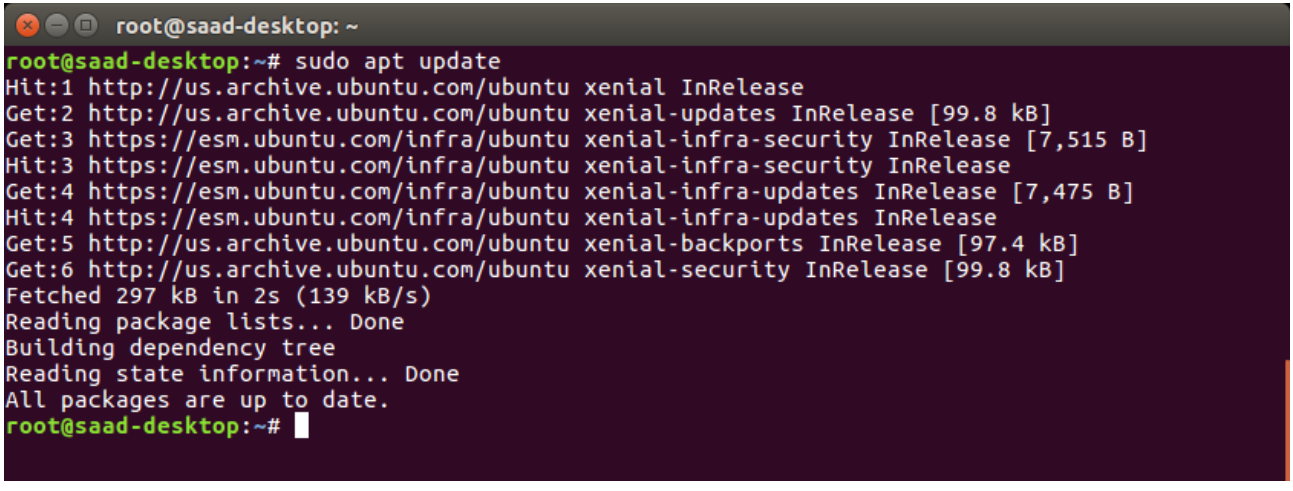
System Call Assignment



Saad Bin Khalid 20K-0161 23 March 2022

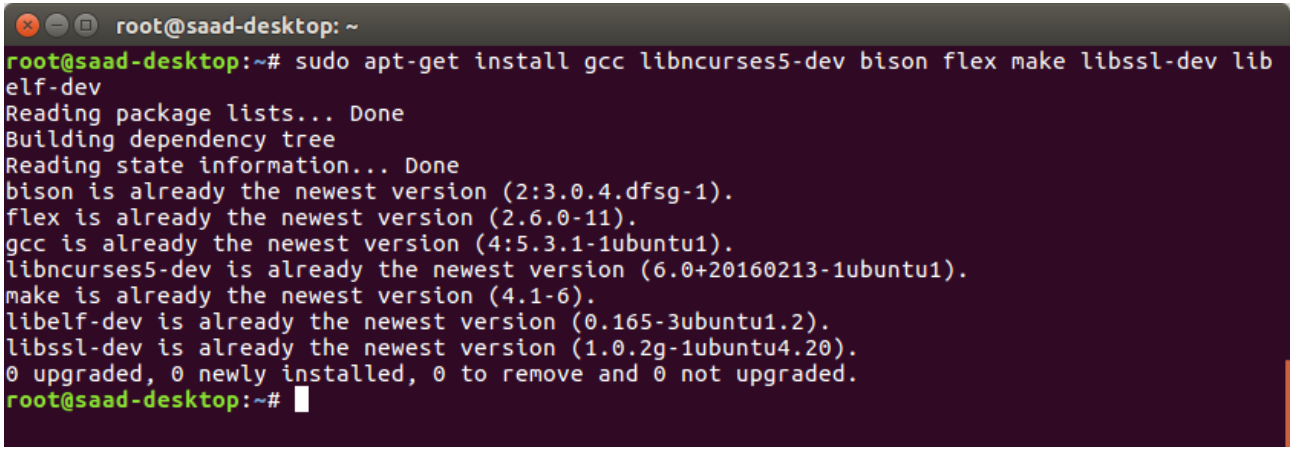
1. Update your apt repositories and sources:

```
sudo apt update
```



2. Install some important dependencies :

```
sudo apt install gcc libncurses5-dev bison flex make libssl-dev libelf-dev
```



3. Downloading the new kernel:

   1. Display the current version of our kernel (4.4.0).

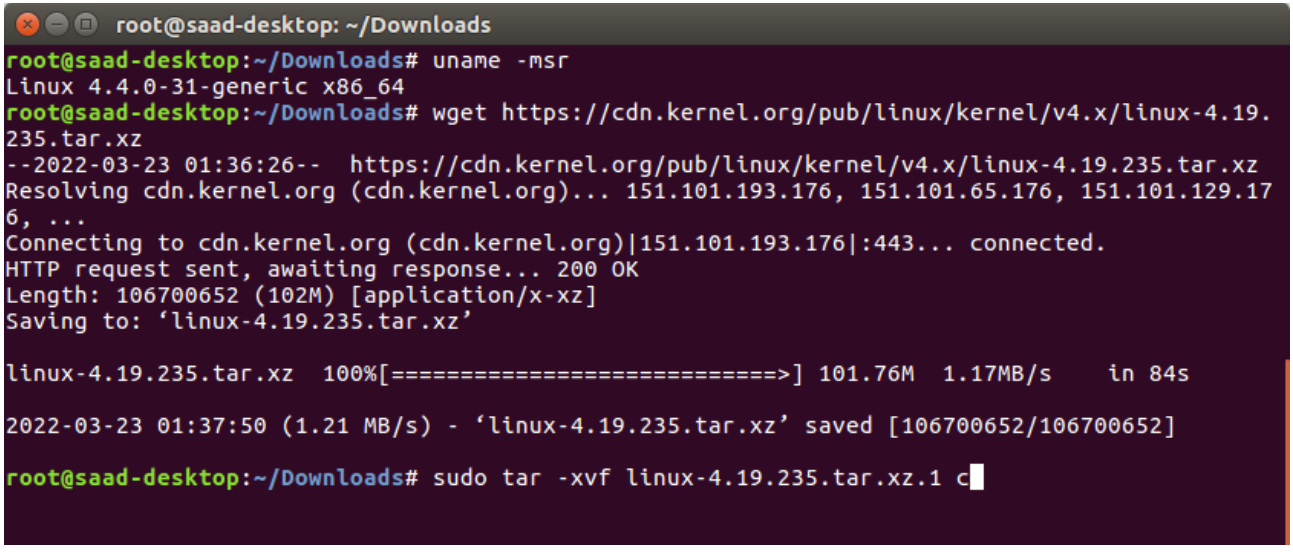
      ```
      uname -msr
      ```

   2. Download the compressed source code of the new kernel. You can also download it manually from kernel.org website.

      ```
      wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.19.235.tar.xz
      ```

   3. Decompress it to extract all files in the '~/Downloads' directory. You can also use a GUI tool e.g. right-click the file and select extract here.

      ```
      tar -xvf linux-4.19.235.tar.xz
      ```



4. Tweaking the configurations (the lines to change are in **bold**):

   1. Create a system call in linux.

```
cd linux-4.19.235
mkdir hello && cd hello && nano hello.c
```

```
line 1   |  #include <linux/kernel.h>
line 2   |  asmlinkage long sys_hello(void) {
line 3   |        printk("Hello from Saad – k200161\n");
line 4   |        return 0; }
```

   2. Create a Makefile for our system call.

```
nano Makefile
```

```
line 1   |  obj-y := hello.c
```

   3. Include our system call in the kernel Makefile.

```
cd.. && nano Makefile
```

```
line 5   |  EXTRAVERSION =-200161   # you can use your roll number
line 987 |  core-y += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/
             block/ hello/
```

   4. Index the system call. (335 was the next index in my case).

```
nano arch/x86/entry/syscalls/syscall_64.tbl
```

```
line 346 |  335   64    hello            sys_hello
```

   5. Add the prototype of our system call in kernel.

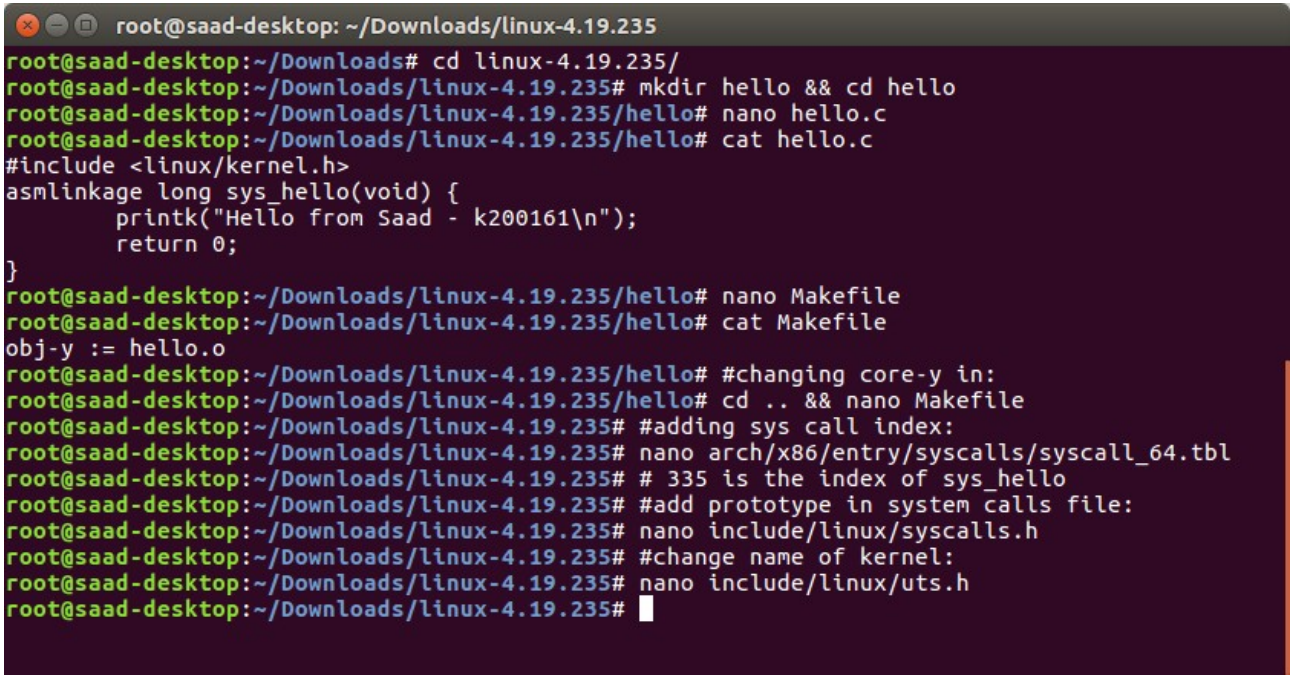
```
nano include/linux/syscalls.h
```

```
line -2  |  asmlinkage long sys_hello(void);
line -1  |  #endif                                  # last line
```

   6. Rename the kernel.

```
nano include/linux/uts.h
```

```
line 8   |  #define UTS_SYSNAME "Saad_Linux"
```



5. Making the configuration file:

   1. Locate the configuration file of current kernel.

```
ls boot | grep config
```

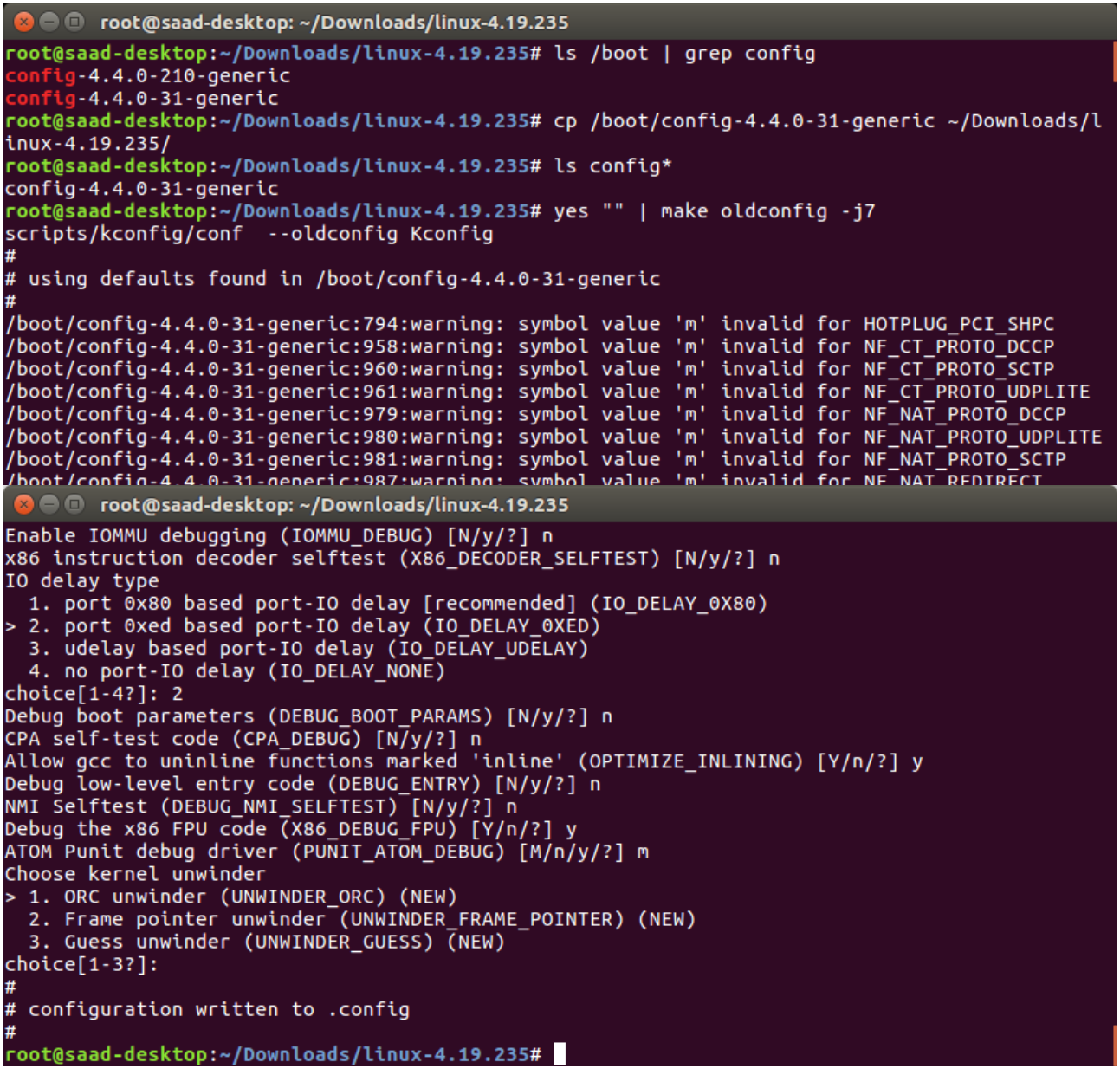
   2. Copy one of the located configuration file in our kernel directory.

```
cp  /boot/config-4.40-31-generic  ~/Downloads/linux-4.19.235
```

   3. Make the configuration:

```
yes “” | make oldconfig -j7
```

```
-jX is an optional flag – X is the number of cores you want to use while
compiling.
```


root@saad-desktop: ~/Downloads/linux-4.19.235
root@saad-desktop:~/Downloads/linux-4.19.235# ls /boot | grep config
config-4.4.0-210-generic
config-4.4.0-31-generic
root@saad-desktop:~/Downloads/linux-4.19.235# cp /boot/config-4.4.0-31-generic ~/Downloads/linux-4.19.235/
root@saad-desktop:~/Downloads/linux-4.19.235# ls config*
config-4.4.0-31-generic
root@saad-desktop:~/Downloads/linux-4.19.235# yes "" | make oldconfig -j7
scripts/kconfig/conf --oldconfig Kconfig
#
using defaults found in /boot/config-4.4.0-31-generic
#
/boot/config-4.4.0-31-generic:794:warning: symbol value 'm' invalid for HOTPLUG_PCI_SHPC
/boot/config-4.4.0-31-generic:958:warning: symbol value 'm' invalid for NF_CT_PROTO_DCCP
/boot/config-4.4.0-31-generic:960:warning: symbol value 'm' invalid for NF_CT_PROTO_SCTP
/boot/config-4.4.0-31-generic:961:warning: symbol value 'm' invalid for NF_CT_PROTO_UDPLITE
/boot/config-4.4.0-31-generic:979:warning: symbol value 'm' invalid for NF_NAT_PROTO_DCCP
/boot/config-4.4.0-31-generic:980:warning: symbol value 'm' invalid for NF_NAT_PROTO_UDPLITE
/boot/config-4.4.0-31-generic:981:warning: symbol value 'm' invalid for NF_NAT_PROTO_SCTP
root@saad-desktop: ~/Downloads/linux-4.19.235
Enable IOMMU debugging (IOMMU_DEBUG) [N/y/?] n
x86 instruction decoder selftest (X86_DECODER_SELFTEST) [N/y/?] n
IO delay type
  1. port 0x80 based port-IO delay [recommended] (IO_DELAY_0X80)
> 2. port 0xed based port-IO delay (IO_DELAY_0XED)
  3. udelay based port-IO delay (IO_DELAY_UDELAY)
  4. no port-IO delay (IO_DELAY_NONE)
choice[1-4?]: 2
Debug boot parameters (DEBUG_BOOT_PARAMS) [N/y/?] n
CPA self-test code (CPA_DEBUG) [N/y/?] n
Allow gcc to uninline functions marked 'inline' (OPTIMIZE_INLINING) [Y/n/?] y
Debug low-level entry code (DEBUG_ENTRY) [N/y/?] n
NMI Selftest (DEBUG_NMI_SELFTEST) [N/y/?] n
Debug the x86 FPU code (X86_DEBUG_FPU) [Y/n/?] y
ATOM Punit debug driver (PUNIT_ATOM_DEBUG) [M/n/y/?] m
Choose kernel unwinder
> 1. ORC unwinder (UNWINDER_ORC) (NEW)
  2. Frame pointer unwinder (UNWINDER_FRAME_POINTER) (NEW)
  3. Guess unwinder (UNWINDER_GUESS) (NEW)
choice[1-3?]:
#
configuration written to .config
#
root@saad-desktop:~/Downloads/linux-4.19.235#

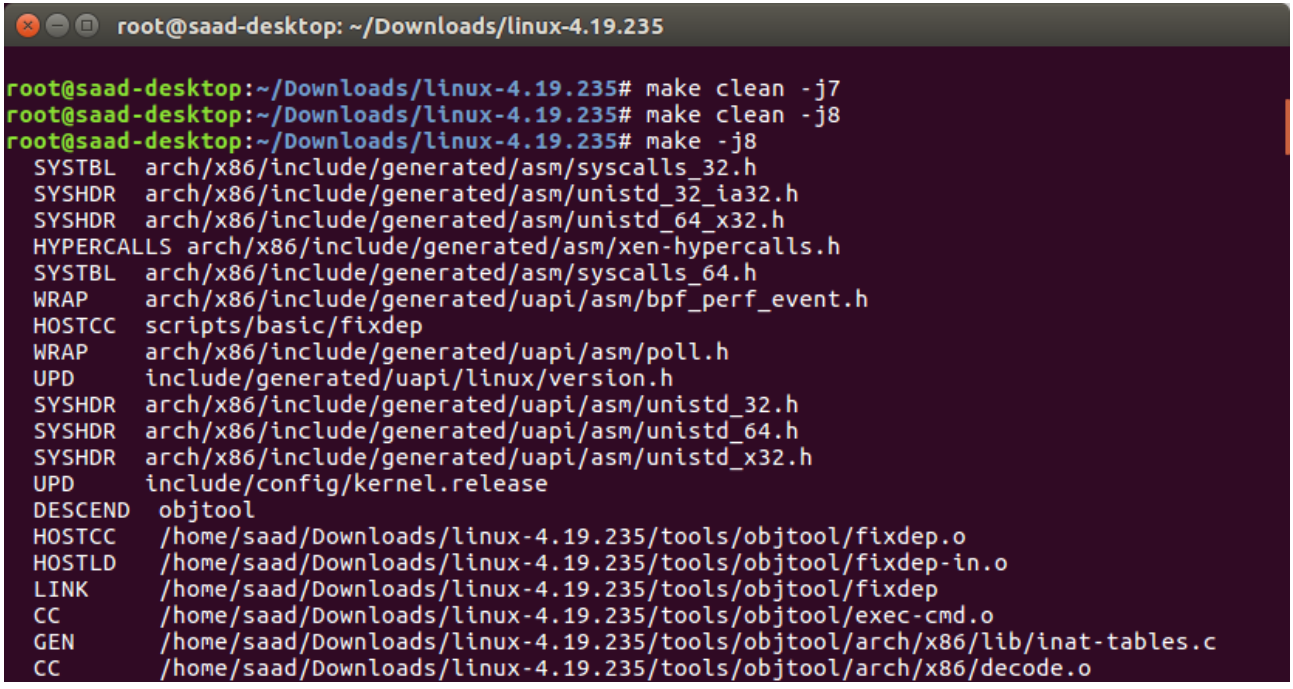
6. Compile the kernel:

   1. Clean any old compilations:

```
make clean -j7
```

   2. Compile new kernel and wait. It can take a lot of time (depending on your system's performance).

```
make -j8
```



7. Install the compiled kernel:

```
make modules_install install
```

8. Restart your computer:

```
shutdown -r now
```

9. While computer is booting up, make sure to open the grub menu and choose ‘Advanced options for Ubuntu’. You will see some kernels to choose from. Choose the the kernel which we just installed in our system. (4.19.235 in my case)

10. After rebooting:

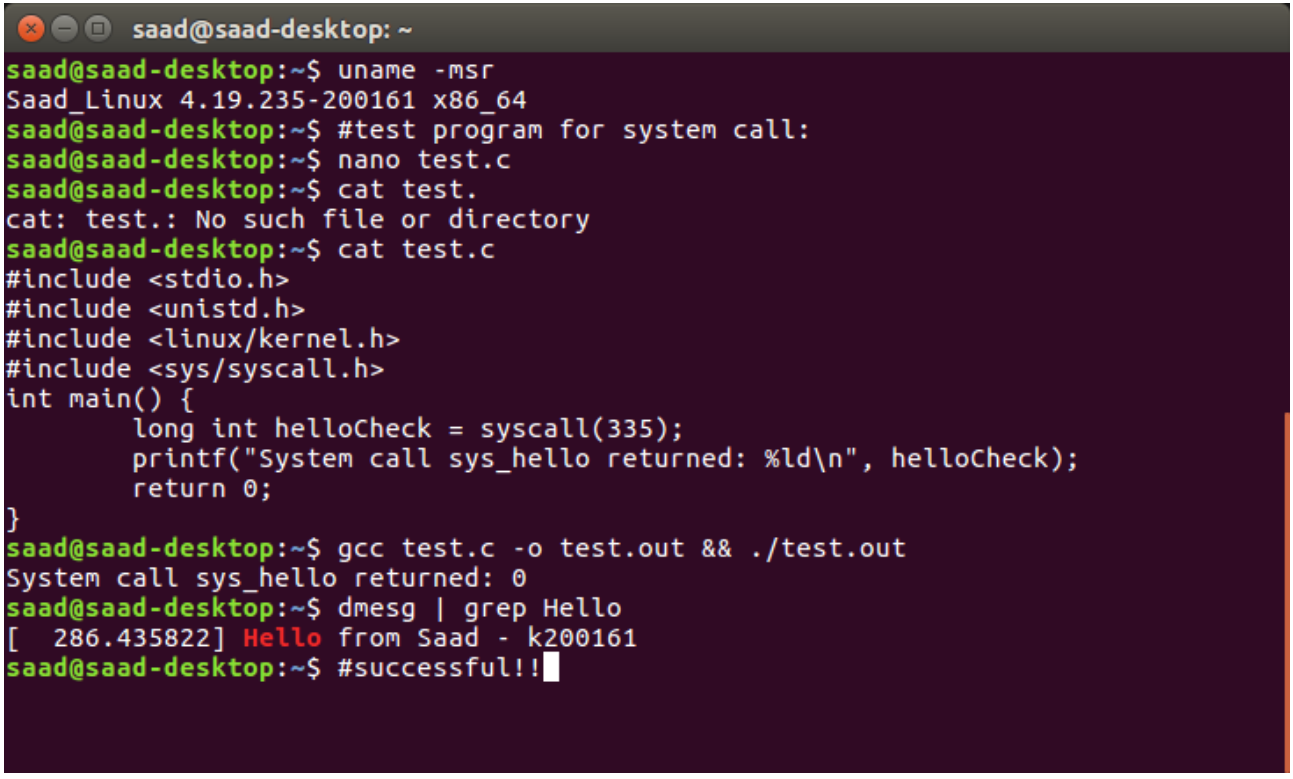
    1. Check the kernel version:

    ```
    uname -msr
    ```

    2. Test the system call we just developed. Compile and run the following C program:

    ```
    line 1   |  #include <linux/kernel.h>
    line 2   |  #include <sys/syscall.h>
    line 3   |  #include <unistd.h>
    line 4   |  #include <stdio.h>
    line 5   |  int main(void) {
    line 6   |        printf(“syscall(335) returned %ld\n”, syscall(335));
    line 7   |        // note: 335 was the index we found in Step 4.4
    line 8   |        return 0;
    line 9   |  }
    ```

    3. If the returned value is 0, then congratulations! You just implemented your own system call.


saad@saad-desktop: ~
saad@saad-desktop:~$ uname -msr
Saad_Linux 4.19.235-200161 x86_64
saad@saad-desktop:~$ #test program for system call:
saad@saad-desktop:~$ nano test.c
saad@saad-desktop:~$ cat test.
cat: test.: No such file or directory
saad@saad-desktop:~$ cat test.c
#include <stdio.h>
#include <unistd.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
int main() {
long int helloCheck = syscall(335);
printf("System call sys_hello returned: %ld\n", helloCheck);
return 0;
}
saad@saad-desktop:~$ gcc test.c -o test.out && ./test.out
System call sys_hello returned: 0
saad@saad-desktop:~$ dmesg | grep Hello
[286.435822] Hello from Saad - k200161
saad@saad-desktop:~$ #successful!!