Install and compile kernel on Linux Ubuntu

Iman Rasouli Parto

University of Tehran

Abstract*— *This article provides instructions for installing and compiling a new version of kernel in the Ubuntu distribution.

Keywords*—*module, Bootloader, kernel, compile, config menu

Introduction: The Linux kernel is the main component of a **Linux operating system (OS)** and is the core interface between a computer's hardware and its processes. It communicates between the 2, managing resources as efficiently as possible.

The kernel is so named because—like a seed inside a hard shell—it exists within the OS and controls all the major functions of the hardware, whether it's a phone, laptop, server, or any other kind of computer.

At the beginning, we need to update the system using the command below:

```
sudo apt update && sudo apt upgrade
```

Before compiling the kernel, we will need some required build tools. The following command will install them:

```
sudo apt install git fakeroot build-essential ncurses-dev libssl-dev bc flex libelf-dev bison dwarves
```

Now, we are ready to compile and install the kernel.

Step 1: Download the source code

We can download the Linux kernel source code from kernel.org website or use the command below (we'll consider 6.1.39 version to install):

```
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.39.tar.xz
```

Step 2: Extract the source code

Extract downloaded source code using the command below:

```
tar xvf linux-6.1.39.tar.xz
```

Step 3: Configure kernel

After extracting the source code, navigate to the source directory:

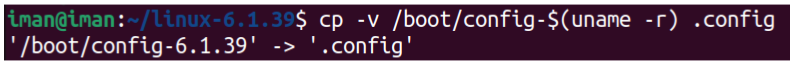
```
cd linux-6.1.39
```

The configuration of kernel can be specified in a `.config` file. For this, we can use the config file of current kernel installed on OS:

```
cp -v /boot/config-$(uname -r) .config
```

This command used to copy current configuration of current system and not necessary if you are configuring a new system.

The terminal's output would be like this:



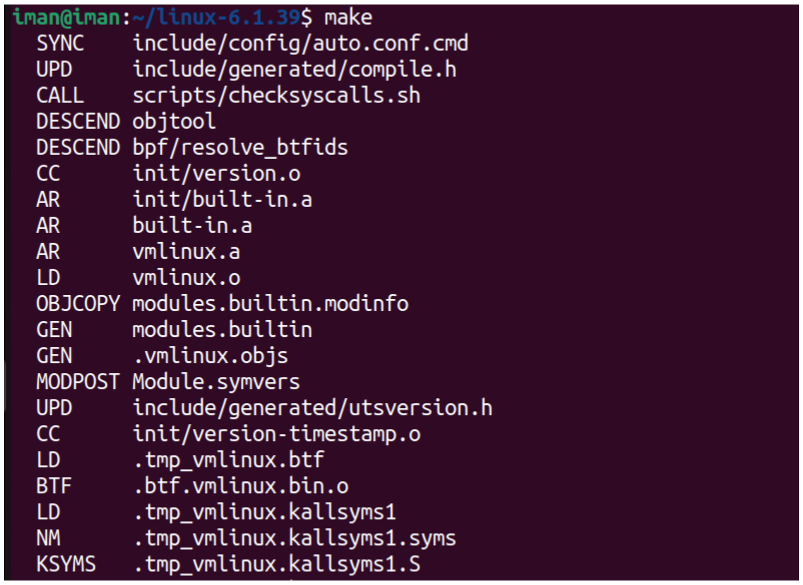
```
iman@iman:~/linux-6.1.39$ cp -v /boot/config-$(uname -r) .config
'/boot/config-6.1.39' -> '.config'
```

Figure 1. Copy current kernel's config file to new kernel directory

(The 'uname' -r command output is current version of system's kernel.)

Step 4: Build the kernel

Build the kernel using make command.

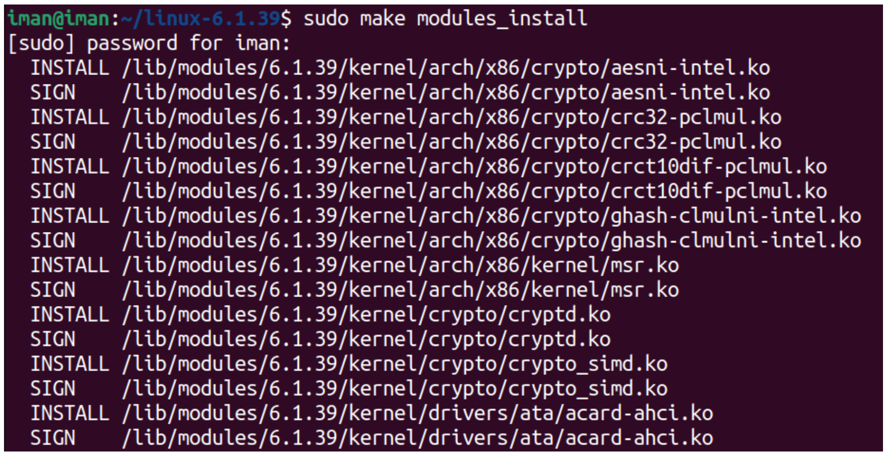


```
iman@iman:~/linux-6.1.39$ make
  SYNC    include/config/auto.conf.cmd
  UPD     include/generated/compile.h
  CALL    scripts/checksyscalls.sh
  DESCEND objtool
  DESCEND bpf/resolve_btfids
  CC      init/version.o
  AR      init/built-in.a
  AR      built-in.a
  AR      vmlinux.a
  LD      vmlinux.o
  OBJCOPY modules.builtin.modinfo
  GEN     modules.builtin
  GEN     .vmlinux.objs
  MODPOST Module.symvers
  UPD     include/generated/utsversion.h
  CC      init/version-timestamp.o
  LD      .tmp_vmlinux.btf
  BTF     .btf.vmlinux.bin.o
  LD      .tmp_vmlinux.kallsyms1
  NM      .tmp_vmlinux.kallsyms1.syms
  KSYMS   .tmp_vmlinux.kallsyms1.S
```

Figure 2. Compile and build the kernel

There will be some modules required. Install them with the command below:

```
sudo make install_modules
```

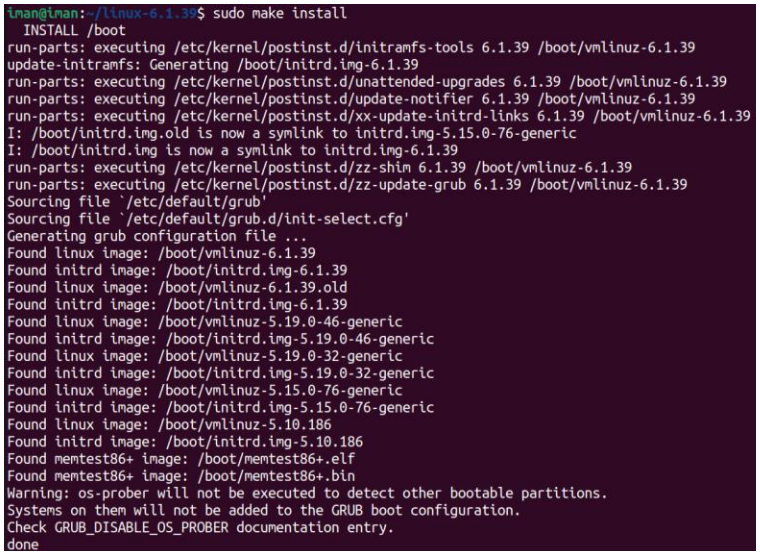


```
iman@iman:~/linux-6.1.39$ sudo make modules_install
[sudo] password for iman:
  INSTALL /lib/modules/6.1.39/kernel/arch/x86/crypto/aesni-intel.ko
  SIGN    /lib/modules/6.1.39/kernel/arch/x86/crypto/aesni-intel.ko
  INSTALL /lib/modules/6.1.39/kernel/arch/x86/crypto/crc32-pclmul.ko
  SIGN    /lib/modules/6.1.39/kernel/arch/x86/crypto/crc32-pclmul.ko
  INSTALL /lib/modules/6.1.39/kernel/arch/x86/crypto/crct10dif-pclmul.ko
  SIGN    /lib/modules/6.1.39/kernel/arch/x86/crypto/crct10dif-pclmul.ko
  INSTALL /lib/modules/6.1.39/kernel/arch/x86/crypto/ghash-clmulni-intel.ko
  SIGN    /lib/modules/6.1.39/kernel/arch/x86/crypto/ghash-clmulni-intel.ko
  INSTALL /lib/modules/6.1.39/kernel/arch/x86/kernel/msr.ko
  SIGN    /lib/modules/6.1.39/kernel/arch/x86/kernel/msr.ko
  INSTALL /lib/modules/6.1.39/kernel/crypto/cryptd.ko
  SIGN    /lib/modules/6.1.39/kernel/crypto/cryptd.ko
  INSTALL /lib/modules/6.1.39/kernel/crypto/crypto_simd.ko
  SIGN    /lib/modules/6.1.39/kernel/crypto/crypto_simd.ko
  INSTALL /lib/modules/6.1.39/kernel/drivers/ata/acard-ahci.ko
  SIGN    /lib/modules/6.1.39/kernel/drivers/ata/acard-ahci.ko
```

Figure 3. Installing kernel's required modules

Then, install the kernel:

sudo make install



```
iman@iman:~/linux-6.1.39$ sudo make install
  INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.1.39 /boot/vmlinuz-6.1.39
update-initramfs: Generating /boot/initrd.img-6.1.39
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 6.1.39 /boot/vmlinuz-6.1.39
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.1.39 /boot/vmlinuz-6.1.39
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 6.1.39 /boot/vmlinuz-6.1.39
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-76-generic
I: /boot/initrd.img is now a symlink to initrd.img-6.1.39
run-parts: executing /etc/kernel/postinst.d/zz-shim 6.1.39 /boot/vmlinuz-6.1.39
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.1.39 /boot/vmlinuz-6.1.39
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.39
Found initrd image: /boot/initrd.img-6.1.39
Found linux image: /boot/vmlinuz-6.1.39.old
Found initrd image: /boot/initrd.img-6.1.39
Found linux image: /boot/vmlinuz-5.19.0-46-generic
Found initrd image: /boot/initrd.img-5.19.0-46-generic
Found linux image: /boot/vmlinuz-5.19.0-32-generic
Found initrd image: /boot/initrd.img-5.19.0-32-generic
Found linux image: /boot/vmlinuz-5.15.0-76-generic
Found initrd image: /boot/initrd.img-5.15.0-76-generic
Found linux image: /boot/vmlinuz-5.10.186
Found initrd image: /boot/initrd.img-5.10.186
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
done
```

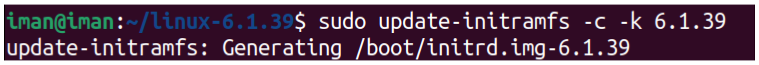
Figure 4. Installing the kernel

The output shows "**done**" when installing finished.

Step 5: Update the Bootloader

The GRUB bootloader is the first program that runs when the system powers on. First update "initramfs" to the installed kernel version:

sudo update-initramfs -c -k 6.1.39

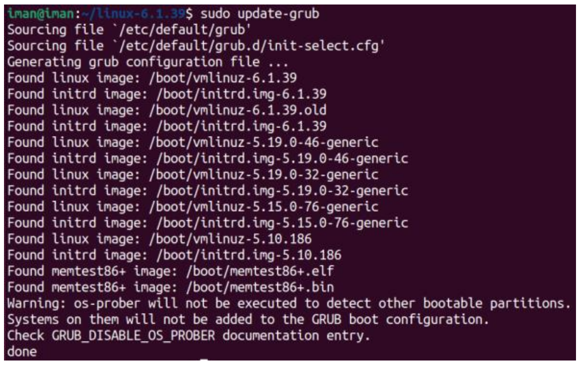


```
iman@iman:~/linux-6.1.39$ sudo update-initramfs -c -k 6.1.39
update-initramfs: Generating /boot/initrd.img-6.1.39
```

Figure 5. Update "inittramfs"

Then, update "GURB" bootloader:

sudo update-grub

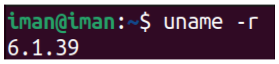


```
iman@iman:~/linux-6.1.39$ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.1.39
Found initrd image: /boot/initrd.img-6.1.39
Found linux image: /boot/vmlinuz-6.1.39.old
Found initrd image: /boot/initrd.img-6.1.39
Found linux image: /boot/vmlinuz-5.19.0-46-generic
Found initrd image: /boot/initrd.img-5.19.0-46-generic
Found linux image: /boot/vmlinuz-5.19.0-32-generic
Found initrd image: /boot/initrd.img-5.19.0-32-generic
Found linux image: /boot/vmlinuz-5.15.0-76-generic
Found initrd image: /boot/initrd.img-5.15.0-76-generic
Found linux image: /boot/vmlinuz-5.10.186
Found initrd image: /boot/initrd.img-5.10.186
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
done
```

Figure 6. Update "GURB" bootloader

Step6: Reboot and verify the kernel version

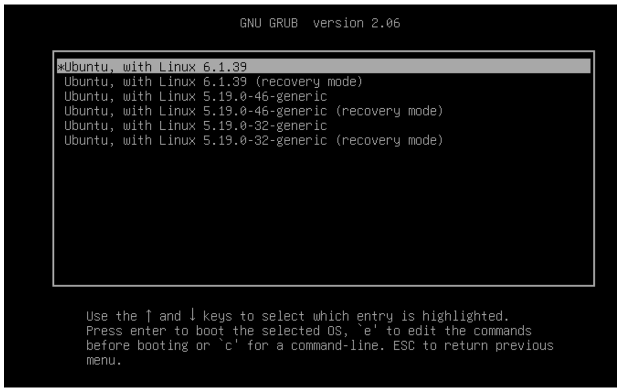
As we completed the steps above, reboot the machine. When the system boots up, verify the kernel version using 'uname -r' command.



```
iman@iman:~$ uname -r
6.1.39
```

Figure 7. Kernel's new version has been installed

We can boot the system with different kernel version each time. If we hold the shift key when system boots up, the boot menu will load, and we can choose which kernel version boot the system with.



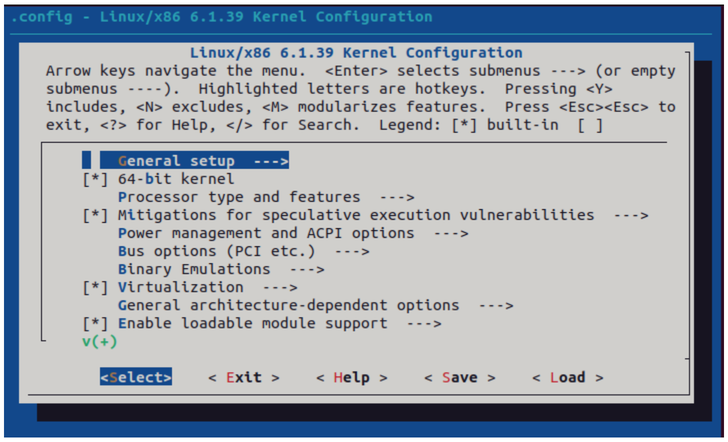
```
                    GNU GRUB  version 2.06

*Ubuntu, with Linux 6.1.39
 Ubuntu, with Linux 6.1.39 (recovery mode)
 Ubuntu, with Linux 5.19.0-46-generic
 Ubuntu, with Linux 5.19.0-46-generic (recovery mode)
 Ubuntu, with Linux 5.19.0-32-generic
 Ubuntu, with Linux 5.19.0-32-generic (recovery mode)

     Use the ↑ and ↓ keys to select which entry is highlighted.
     Press enter to boot the selected OS, `e' to edit the commands
     before booting or `c' for a command-line. ESC to return previous
     menu.
```

Figure 8. Boot menu shows all kernel version installed on the system

Config menu

"menuconfig" is one of several configuration tools available for customizing the Linux kernel build options. It is a text-based configuration interface that allows users to select various kernel features, drivers, and settings.

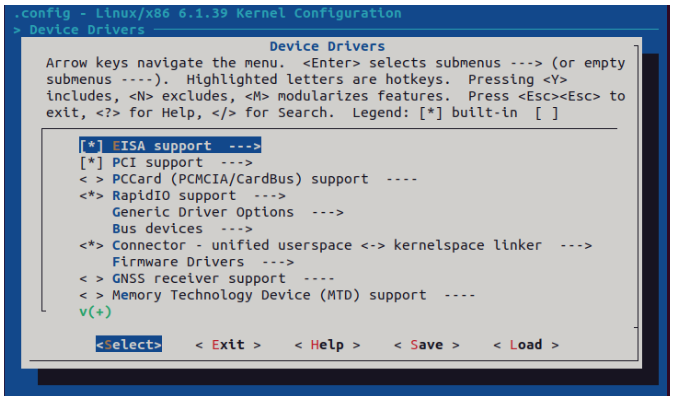
make menuconfig



```
.config - Linux/x86 6.1.39 Kernel Configuration
                Linux/x86 6.1.39 Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
 submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
 includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
 exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

        General setup  --->
    [*] 64-bit kernel
        Processor type and features  --->
    [*] Mitigations for speculative execution vulnerabilities  --->
        Power management and ACPI options  --->
        Bus options (PCI etc.)  --->
        Binary Emulations  --->
    [*] Virtualization  --->
        General architecture-dependent options  --->
    [*] Enable loadable module support  --->
    v(+)

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

Figure 9. Kernel config menu

We can choose features in this menu to be compiled. If we navigate to "Device drivers", we can see the list of drivers in the system, and we can choose whether to enable or disable each drive in the kernel.



```
.config - Linux/x86 6.1.39 Kernel Configuration
> Device Drivers
                                Device Drivers
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      [*] EISA support  --->
      [*] PCI support  --->
      < > PCCard (PCMCIA/CardBus) support  ----
      <*> RapidIO support  --->
          Generic Driver Options  --->
          Bus devices  --->
      <*> Connector - unified userspace <-> kernelspace linker  --->
          Firmware Drivers  --->
      < > GNSS receiver support  ----
      < > Memory Technology Device (MTD) support  ----
      v(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

Figure 10. List of drivers in system

There are some symbols to activate or deactivate the drivers:

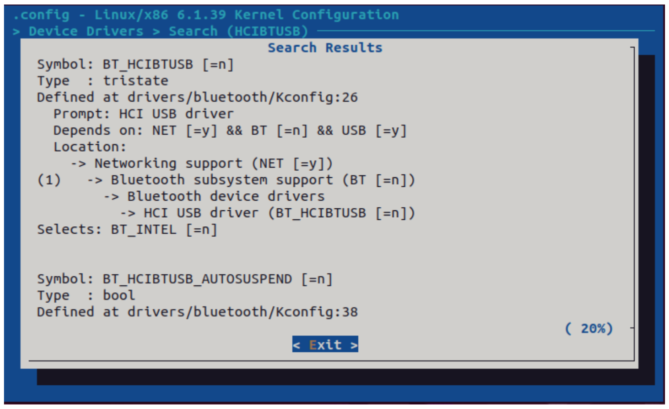
1. **[ ]**, **[*]**: Options in square brackets can be activated or deactivated. The asterisk marks the menu entry as activated. The value can be changed with the space key. It is also possible to press Y key (**Y**es) to activate or N key (**N**o) to deactivate the selected entry.

2. **< >**, **<M>, <*>**: Options in angle brackets can be activated or deactivated, but also activated as module (indicated by a *M*). The values can be modified by pressing Y/N keys as before or by pressing the M key to activate the feature/driver as a module.

3. **{M}**, **{*}**: Options in curly brackets can be activated or activated as module but not be deactivated. This happens because another feature/driver is dependent on this feature.

4. **-M-**, **-*-**: Options between hyphens are activated in the shown way by another feature/driver. There is no choice.

If we search for a module name (for example, Bluetooth module), the menu would jump straight to the option "*Bluetooth device drivers*" in the menu structure.

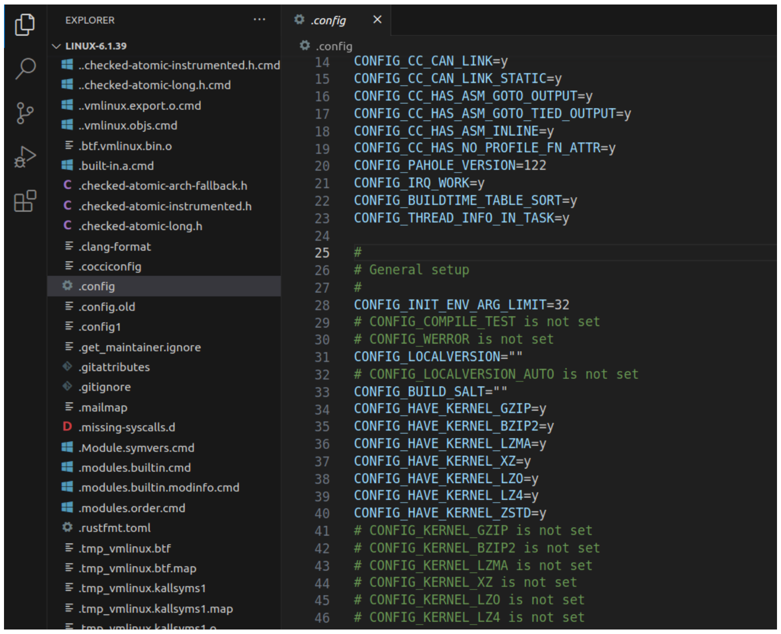


```
.config - Linux/x86 6.1.39 Kernel Configuration
> Device Drivers > Search (HCIBTUSB)
                                Search Results
 Symbol: BT_HCIBTUSB [=n]
 Type  : tristate
 Defined at drivers/bluetooth/Kconfig:26
   Prompt: HCI USB driver
   Depends on: NET [=y] && BT [=n] && USB [=y]
   Location:
     -> Networking support (NET [=y])
 (1)   -> Bluetooth subsystem support (BT [=n])
         -> Bluetooth device drivers
           -> HCI USB driver (BT_HCIBTUSB [=n])
 Selects: BT_INTEL [=n]


 Symbol: BT_HCIBTUSB_AUTOSUSPEND [=n]
 Type  : bool
 Defined at drivers/bluetooth/Kconfig:38
                                                          ( 20%)
                                  < Exit >
```

Figure 11. Bluetooth device (search using "HCIBTUSB")

After configuring the kernel and quitting "menuconfig" it will save as ". /config". The ". /config" file is a plain text file used to store the configuration options for the Linux kernel. "menuconfig" can also be launched using a text editor such as "code" using the following command:

```
code ./config
```

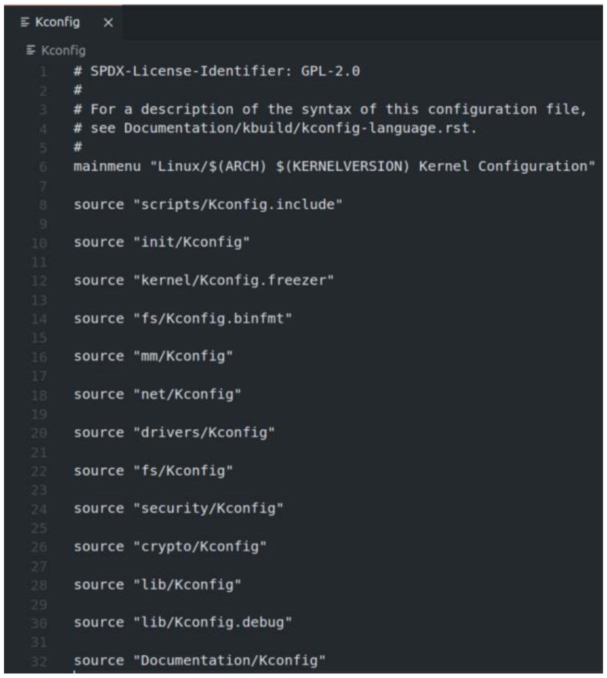


```
14  CONFIG_CC_CAN_LINK=y
15  CONFIG_CC_CAN_LINK_STATIC=y
16  CONFIG_CC_HAS_ASM_GOTO_OUTPUT=y
17  CONFIG_CC_HAS_ASM_GOTO_TIED_OUTPUT=y
18  CONFIG_CC_HAS_ASM_INLINE=y
19  CONFIG_CC_HAS_NO_PROFILE_FN_ATTR=y
20  CONFIG_PAHOLE_VERSION=122
21  CONFIG_IRQ_WORK=y
22  CONFIG_BUILDTIME_TABLE_SORT=y
23  CONFIG_THREAD_INFO_IN_TASK=y
24
25  #
26  # General setup
27  #
28  CONFIG_INIT_ENV_ARG_LIMIT=32
29  # CONFIG_COMPILE_TEST is not set
30  # CONFIG_WERROR is not set
31  CONFIG_LOCALVERSION=""
32  # CONFIG_LOCALVERSION_AUTO is not set
33  CONFIG_BUILD_SALT=""
34  CONFIG_HAVE_KERNEL_GZIP=y
35  CONFIG_HAVE_KERNEL_BZIP2=y
36  CONFIG_HAVE_KERNEL_LZMA=y
37  CONFIG_HAVE_KERNEL_XZ=y
38  CONFIG_HAVE_KERNEL_LZO=y
39  CONFIG_HAVE_KERNEL_LZ4=y
40  CONFIG_HAVE_KERNEL_ZSTD=y
41  # CONFIG_KERNEL_GZIP is not set
42  # CONFIG_KERNEL_BZIP2 is not set
43  # CONFIG_KERNEL_LZMA is not set
44  # CONFIG_KERNEL_XZ is not set
45  # CONFIG_KERNEL_LZO is not set
46  # CONFIG_KERNEL_LZ4 is not set
```

Figure 12. Open "menuconfig" with vs code

As shown in Figure 12, we can change the configuration here. For example, we can modify the activation of "xz" module on line 37 (change y to n).

There are some config keys, every key is connected to a C code by the Kconfig file. When we want to add a C code to kernel then we should create a Config key and define it for a C or directory of code.



```
# SPDX-License-Identifier: GPL-2.0
#
# For a description of the syntax of this configuration file,
# see Documentation/kbuild/kconfig-language.rst.
#
mainmenu "Linux/$(ARCH) $(KERNELVERSION) Kernel Configuration"

source "scripts/Kconfig.include"

source "init/Kconfig"

source "kernel/Kconfig.freezer"

source "fs/Kconfig.binfmt"

source "mm/Kconfig"

source "net/Kconfig"

source "drivers/Kconfig"

source "fs/Kconfig"

source "security/Kconfig"

source "crypto/Kconfig"

source "lib/Kconfig"

source "lib/Kconfig.debug"

source "Documentation/Kconfig"
```

Figure 13. Kconfig file format

Kconfig file is a plain text file that's readable and we can customize the kernel to be installed.

Appendix

A list of installed packages required to compile the kernel is provided:

- ✓ git: Tracks and makes a record of all changes during development in the source code. It also allows reverting the changes.
- ✓ fakeroot: Creates the fake root environment. Used to see the errors while running the command.
- ✓ build-essential: Installs development tools such as C, C++, gcc, and g++ (C, C++ compilers).
- ✓ ncurses-dev: Provides API[1] for the text-based terminals.
- ✓ libssl-dev: Supports **SSL** and **TLS** that encrypt data and make the internet connection secure.
- ✓ bc (Basic Calculator): Supports the interactive execution of statements.
- ✓ flex (Fast Lexical Analyzer Generator): Generates lexical analyzers that convert characters into tokens.
- ✓ libelf-dev: Provides a shared library for managing ELF files (executable files, core dumps and object code).
- ✓ bison: Converts grammar description to a C program.
- ✓ dwarves: A set of tools that use debugging information inserted in ELF binaries by compilers such as gcc, used by well-known debuggers such as gdb, and more recent ones such as systemtap.

References

[1] *Linux Kernel Programming (Kaiwan N Billimoria)*

[2] https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel

[3] https://phoenixnap.com/kb/build-linux-kernel

[4] https://davidaugustat.com/linux/how-to-compile-linux-kernel-on-ubuntu

[5] https://yum-info.contradodigital.com/view-package/epel/dwarves/

[6] https://wiki.gentoo.org/wiki/Kernel/Configuration

[7] https://linuxconfig.org/in-depth-howto-on-linux-kernel-configuration

[1] API stands for **application programming interface**, which is a set of definitions and protocols for building and integrating application software.