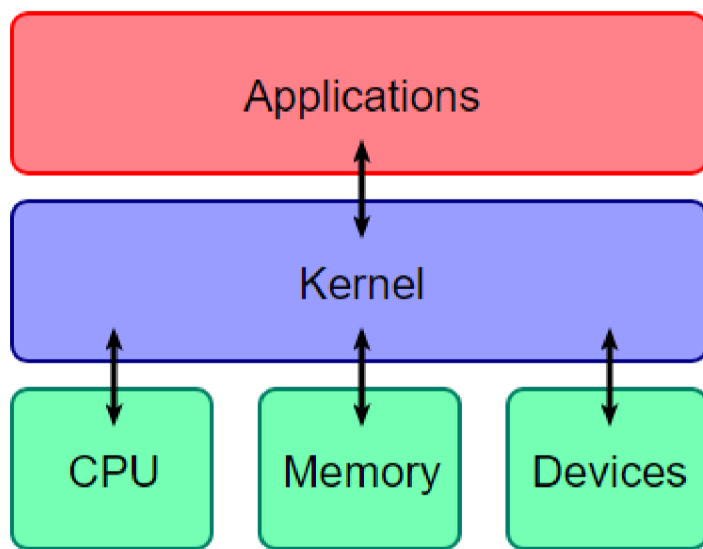


# Contents

Content .....	Error! Bookmark not defined.
Introduction to Kernel.....	2
Configuring New Kernel .....	2
Downloading Dependencies.....	2
Curses Development Kit (CDK) .....	2
OpenSSH/OpenSSL .....	2
Downloading Kernel from <a href="http://www.kernel.org">www.kernel.org</a> .....	3
Configuring Kernel in Linux System .....	4
Copy Old Config File to Our New Kernel Folder .....	5
Naming the New Kernel .....	5
Setting Configuration Parameter.....	6
Building Kernel.....	6
Make the New Kernel Bootable.....	7
Verification.....	7

## Introduction to Kernel

The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system. On most systems, it is one of the first programs loaded on start-up (after the bootloader). It handles the rest of start-up as well as input/output requests from software, translating them into data-processing instructions for the central processing unit. It handles memory and peripherals like keyboards, monitors, printers, and speakers.



## Configuring New Kernel

### Downloading Dependencies

#### Curses Development Kit (CDK)

CDK is a library written in C that provides a collection of widgets for text user interfaces (TUI) development. The widgets wrap n-curses functionality to make writing full screen curses programs faster. Perl and Python bindings are also available.

There are two versions of the library. It was originally written by Mike Glover, introduced as version 4.6 in computer sources UNIX. The other version was extended beginning in May 1999 by Thomas Dickey.

To download CDK, use the following commands

```
$ sudo add-apt-repository "deb http://archive.ubuntu.com/ubuntu $(lsb_release -sc) main universe"
$ sudo apt-get update
$ sudo apt-get install libncurses5-dev
```

### OpenSSH/OpenSSL

OpenSSL is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library.

To download this, write this command in your terminal as a superuser.

```
$ sudo apt-get install libssl-dev
```

Downloading Kernel from [www.kernel.org](http://www.kernel.org)

The website provides three ways to download a kernel copy from their website.

- 1. GIT Clone
- 2. Rsync
- 3. Wget

We will use ‘wget’ as this is the simplest and easiest ways to download a file from a link from the terminal. To get a downloadable link of kernel you wish to download simply log on to their website which is kernel.org, copy the tarball link of the version you want to download and then on the terminal type:

```
$ wget <link of kernel archive>
```

Example of how to download kernel version 4.13.1 is given below, Now we wish to download this kernel in our root (/) directory under a directory named ‘kernels’ hence we will execute the following commands

```
~/ $ sudo -i
$ mkdir /kernels
$ cd /kernels
/kernels$ wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.13.1.tar.xz
```

Now that our kernel is downloaded, we will extract the tar file to do this use the command below:

```
/kernels$ tar xf linux-4.13.1.tar.xz
/kernels$ cd linux-4.13.1
/kernel/linux-4.13.1$
```

Once the extraction is completed, we can now look inside of what we have downloaded. Following is the description of the directories inside the kernel package which we have downloaded.

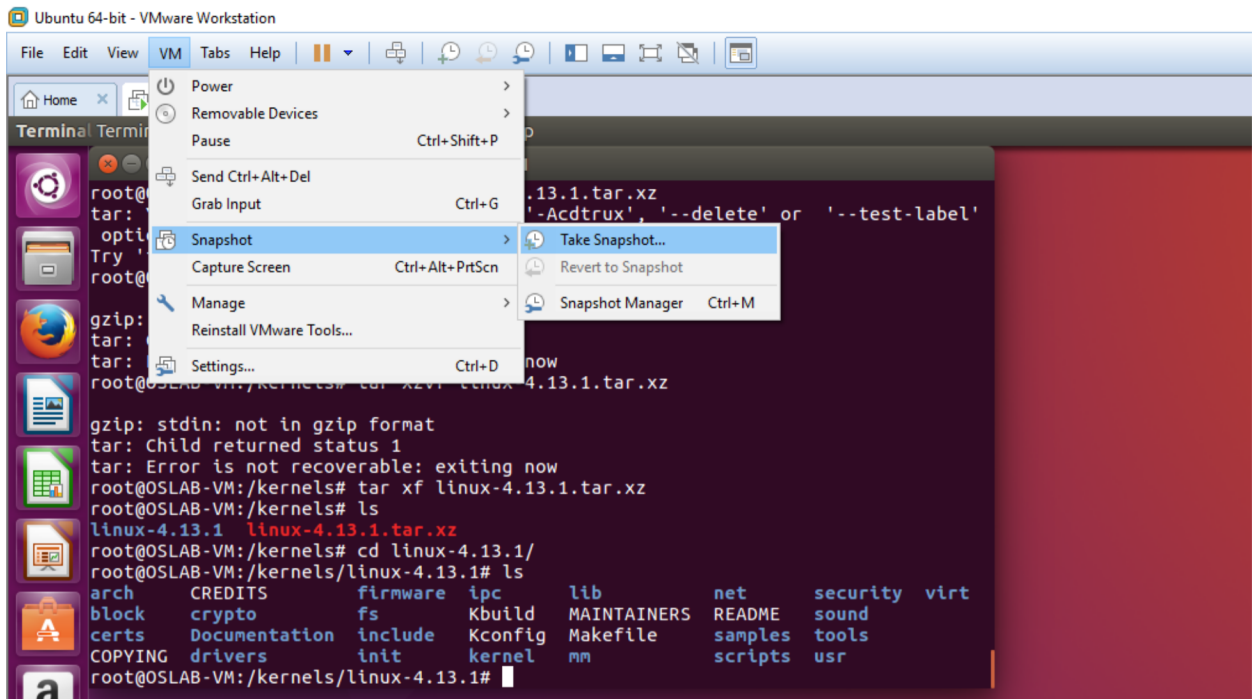
Directory	Description
arch	architecture specific source
Documentation	Linux kernel documentation about each any everything that is a part of Linux kernel from hardware to software.
Init	this directory contains source code that is essential to create initial RAM file system ofthe Kernel
lib	this directory contains library files important for Kernel development
README	this file contains information on Kernel source
Sound	this directory contains sound drivers for popular architectures

<b>block</b>	this directory contains files that are important for memory blocks management
<b>drivers</b>	this directory contains device drivers for Linux
<b>ipc</b>	this directory contains source code that implements locks and semaphores along with multithreading support and inter-process communication means such as shared memory and pipes.
<b>MAINTAINERS</b>	this file contains contact information on the people around the world that are contributing to the maintenance of Linux kernel source.
<b>REPORTING-BUG</b>	this file contains the links where you can report bugs in the source
<b>Tools</b>	this directory contains source files for tools that come in package with Linux distributions, to help users customize Linux.
<b>COPYING</b>	this file contains copyright information for the kernel source
<b>firmware</b>	this directory contains source that helps you customizing Linux for embedded devices
<b>Kbuild</b>	this filehelps in building kernel
<b>Makefile</b>	this file contains important configuration settings and the directory paths to important files that are required to compile Linux Kernel
<b>Samples</b>	this directory containssome sample Kernel programming codes
<b>Usr</b>	saves configuration files for users
<b>CREDITS</b>	this file contains the list of people who contributed to the development of this version of Linux Kernel
<b>fs</b>	source files related to Linux supported file systems
<b>Kconfig</b>	kernel configuration details
<b>mm</b>	memory management scripts
<b>scripts</b>	miscellaneous scripts to run various files at once –usually used for maintenance purposes
<b>virt</b>	virtualization support –virtual memory support
<b>crypto</b>	encryption and file security related sources
<b>include</b>	general development header files
<b>kernel</b>	kernel core functionality source file
<b>net</b>	network support files
<b>security</b>	general security related source file

Now that we have downloaded the kernel, it is time to configure this kernel into the system.

### Configuring Kernel in Linux System

Before we actually get down to configure the kernel let us take a snapshot of our VM machine. This will allow us to revert back if anything is broke and we have to start all over again from installing the operating system. To take the snapshot click to **VM -> take snapshot**, A pop up will appear asking for the name of the snapshot and optional description of the snapshot. Type the appropriate information and click ‘Create’ to create a snapshot. The screenshot below will demonstrate how to take snapshot in VMWare Workstation.

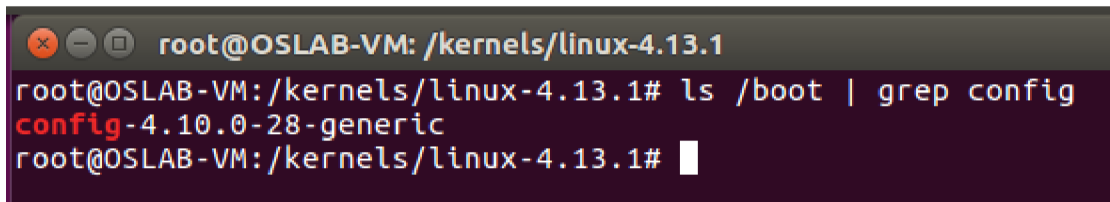


### Copy Old Config File to Our New Kernel Folder

We copy the configuration file of the running kernel to our new kernel; the configuration files are in /boot directory. We can see these files using the following command.

```
/kernels/linux-4.13.1$ ls /boot | grep config
```

The above command would generate the following output



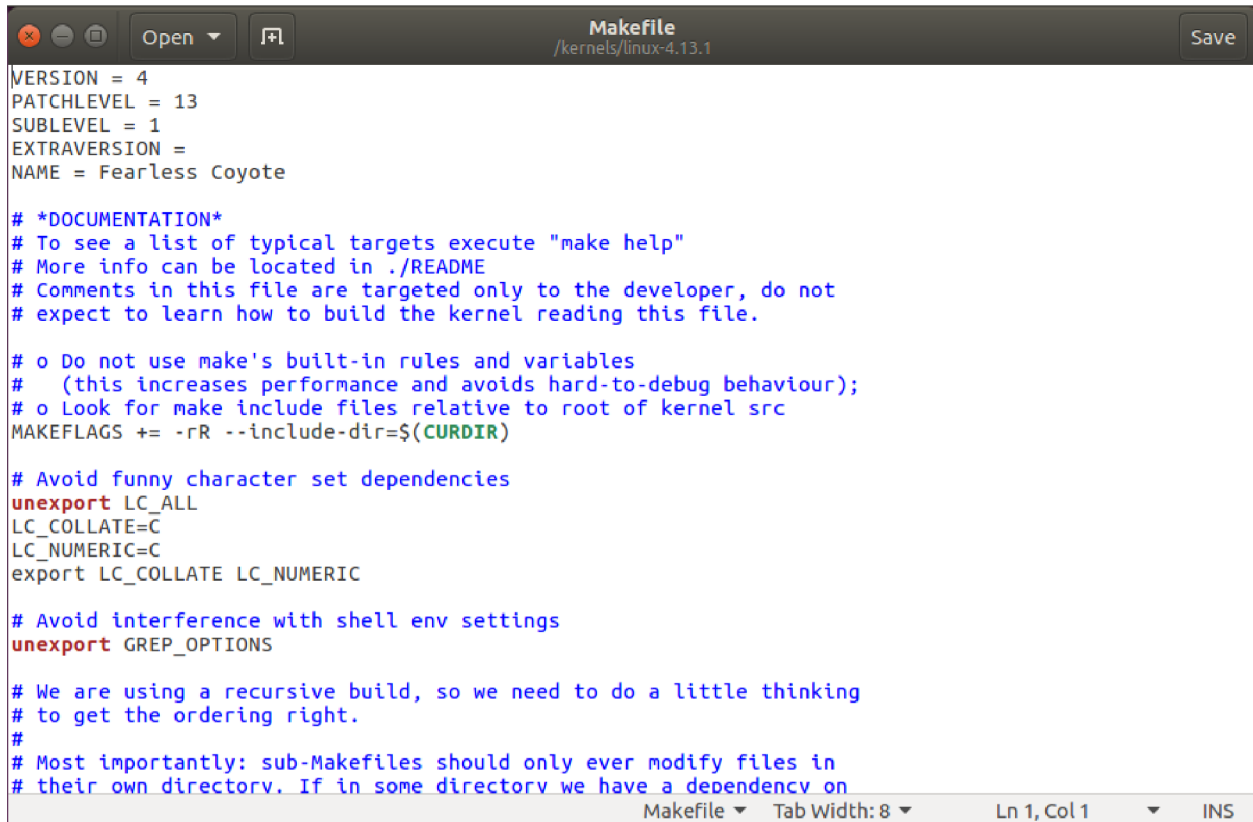
Hence the listed file needed to be copied in our new kernel directory. To do this use the following command. The below command downloads and copies the configuration of the running kernel to the hidden configurations of our downloaded kernel.

```
/kernels/linux-4.13.1$ cp /boot/config-4.10.0-28-generic .config
```

### Naming the New Kernel

We change the new kernel's name to another, in order to make it distinguishable. To do the we can edit 'Makefile' which is present in our new kernel directory. To do that the below command would show the Makefile content and the screenshot below will make it clear.

```
/kernels/linux-4.13.1$ gedit Makefile
```



```
VERSION = 4
PATCHLEVEL = 13
SUBLEVEL = 1
EXTRAVERSION =
NAME = Fearless Coyote

# *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.

# o 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
```

Setting Configuration Parameter

Now we will set the configuration parameters to the new kernel of our running kernel we need to issue the following command.

```
/kernels/linux-4.13.1$ make oldconfig -j4
```

OR

```
/kernels/linux-4.13.1$ yes " | make oldconfig -j4
```

Building Kernel

We need to create a bootable image (bzImage) of the downloaded kernel, first we need to clean up before building. To do that run the following command. This command will clean intermediate temporary files before building kernel.

```
/kernels/linux-4.13.1$ make clean -j4
```

**Question: what is -j4?**

Now we will build the kernel into an installable image file, to do this run the command on your terminal

```
/kernel/linux-4.13.1$ make bzImage -j4
```

If you have done correctly so far now, you should have the same terminal output as below screenshot.

```
AS      arch/x86/boot/compressed/piggy.o
DATAREL arch/x86/boot/compressed/vmlinux
LD      arch/x86/boot/compressed/vmlinux
ZOFFSET arch/x86/boot/zoffset.h
OBJCOPY arch/x86/boot/vmlinux.bin
AS      arch/x86/boot/header.o
LD      arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
BUILD   arch/x86/boot/bzImage
Setup is 17244 bytes (padded to 17408 bytes).
System is 7393 kB
CRC 7e3eae8
Kernel: arch/x86/boot/bzImage is ready  (#1)
root@OSLAB-VM:/kernels/linux-4.13.1#
```

### Make the New Kernel Bootable

Now you take the result of the image we just created and make it bootable so that GRUB can find it at the boot time. To make it bootable we need to follow the three steps

#### *Step 1) Install Compress Kernel Image into the Boot Directory*

To install the kernel image into the /boot directory, we need to execute the following three commands

```
/kernel/linux-4.13.1$ make modules -j4
/kernel/linux-4.13.1$ make modules_install -j4
/kernel/linux-4.13.1$ make install -j4
```

#### *Step 2) Install Compress Kernel Image into the /boot Directory*

```
/kernel/linux-4.13.1$ sudo initramfs -c -k 4.13.1-Ali+
```

#### *Step 3) Update GRUB*

```
/kernel/linux-4.13.1$ sudo update-grub
```

### Verification

To verify that the new kernel is configured and is ready, we need to answer the following questions positively:

- Your '/boot' directory contain new kernel, initrd image and configuration files?
- Is there a new module directory?
- Has your GRUB been updated to include your new kernel.

If all the above questions' answer is yes, then we are ready to boot into our new kernel, to do this reboot your system and press 'shift' key and hold it when VMWare logo appears to get into our GRUB Menu.

After the reboot process is completed and you are logged back into your desktop, run the following command to see which kernel you are currently running.

```
$ uname -r
```