

Lab01

Ahmed Siradj Eddine Bekkari

November 2024

Source Code Download

Download Methods:

I have used the `wget` command to download the **linux kernel**. Then i extracted it.

```

siradj@siradj:/usr/src$ sudo wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.10.229.tar.xz
--2024-11-15 16:28:25-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.10.229.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.193.176, 151.101.1.176, 151.101.65.176, ...
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.193.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 126741940 (115M) [application/x-xz]
Saving to: 'linux-5.10.229.tar.xz'

linux-5.10.229.tar.xz      100%[=====] 115.15M  1.51MB/s   in 79s

2024-11-15 16:29:45 (1.45 MB/s) - 'linux-5.10.229.tar.xz' saved [126741940/126741940]

siradj@siradj:/usr/src$ _

siradj@siradj:/usr/src$ sudo tar -xvf linux-5.10.229.tar.xz
siradj@siradj:/usr/src$ ls
linux-2.6.32  linux-5.10.229  linux-5.10.229.tar.xz  linux-5.15.171  linux-headers-6.8.0-48  linux-headers-6.8.0-48-generic
siradj@siradj:/usr/src$ cd linux-5.10.229/
siradj@siradj:/usr/src/linux-5.10.229$ ls
arch  certs  CREDITS  Documentation  fs  init  ipc  Kconfig  lib  MAINTAINERS  mm  README  scripts  sound  usr
block  COPYING  crypto  drivers  include  io_uring  Kbuild  kernel  LICENSES  Makefile  net  samples  security  tools  vmlinux
siradj@siradj:/usr/src/linux-5.10.229$

```

Figure 1: Download And Extract

Source Code Structure

Files and Directorie description:

- **arch/**: Contains architecture-specific code for various platforms (e.g., x86, ARM, MIPS). Each subdirectory corresponds to a specific architecture, housing code optimized for that architecture's needs.
- **drivers/**: Houses device driver code for various hardware components like network interfaces, USB, sound, and storage. This is where hardware-specific control logic is implemented.
- **include/**: Contains header files with declarations and macros used throughout the kernel. These headers are shared across different parts of the kernel, ensuring consistency and modularity.
- **kernel/**: Holds core kernel functionalities and subsystems, including process scheduling, signals, and low-level system management.
- **Documentation/**: Provides detailed documentation, guidelines, and explanations for kernel modules, APIs, and development practices. Essential for developers to understand kernel features and coding standards.
- **Makefile**: The main build script used to compile the kernel. It orchestrates how different components are compiled, assembled, and linked, handling dependencies and configuration settings.

The location of drivers for your hardware is in `/dev`, it's stand for **devices**

```

siradj@siradj:~$ cd /dev/
siradj@siradj:/dev$ ls
autofs          dm-0          initctl       mapper        rkill        stderr        tty17          tty29          tty40          tty52          tty7          ttyS17          ttyS29          uinput         vcsa         vcsu6
block           dm9         input        mcelog       rtc          stdin        tty18          tty3          tty41          tty53          tty8          ttyS18          ttyS9          urandom        vcsa1        vfi0
lsp             dm1         kmsg        pcm          rtc0        stdout       tty19          tty30          tty42          tty54          tty9          ttyS19          ttyS30          userfaultfd   vcsa2        vga_arbiter
btfs-control    ecrptfs     log          queue        sda         tty          tty2          tty31          tty43          tty55          ttyprintk     ttyS2          ttyS31          userio        vcsa3        vhc1
bus             fb0         loop0        net          sda1        tty0         tty20          tty32          tty44          tty56          ttyS0         ttyS20          ttyS4          vboxguest     vcsa4        vhost-net
cdrom           fd          loop1        null         sda2        tty1         tty21          tty33          tty45          tty57          ttyS1         ttyS21          ttyS5          vboxuser      vcsa5        vhost-vsock
char            full        loop2        nvram        sda3        tty10        tty22          tty34          tty46          tty58          ttyS10        ttyS22          ttyS6          vcs          vcsa6        zero
console         fuse        loop3        port         sdb         tty11        tty23          tty35          tty47          tty59          ttyS11        ttyS23          ttyS7          vcs1         vcsu        zfs
core            hidraw0     loop4        ppp          sxl         tty12        tty24          tty36          tty48          tty6         ttyS12        ttyS24          ttyS8          vcs2         vcsu1
cpu             hpet        loop5        psaux        st          tty13        tty25          tty37          tty49          tty60         ttyS13        ttyS25          ttyS9          vcs3         vcsu2
cpu_dma_latency hugepages   loop6        ptmx         snapshot    tty14        tty26          tty38          tty5         tty61          ttyS14        ttyS26          ubuntu-vg     vcs4         vcsu3
cuse            hwrng      loop7        pts          snd          tty15        tty27          tty39          tty50         tty62          ttyS15        ttyS27          udmabuf       vcs5         vcsu4
disk            l2c-0      loop-control random        sr0         tty16        tty28          tty4          tty51          tty63         ttyS16        ttyS28          uhid          vcs6         vcsu5

```

Figure 2: Drivers path

Kernel Configuration

Understanding Configuration Options

- **CONFIG_***: In the Linux kernel, configuration options are prefixed with CONFIG_. These are settings used to enable or disable specific kernel features and modules. Each CONFIG_* option corresponds to a feature, hardware driver, or subsystem within the kernel. For example : **CONFIG_NET** enables networking capabilities.
- **Option types:**
 - **y**: Yes. Compiles the feature directly into the kernel.
 - **m**: Compiles the feature as a loadable module, meaning it can be loaded and unloaded from memory as needed.
 - **n**: No. Disables the feature entirely.
- **Configuration locations:** On .config file, for example in my machine i have found it on this path :
/usr/src/linux-5.10.229/.config

```
siradj@siradj:/dev$ sudo head /usr/src/linux-5.10.229/.config
#
# Automatically generated file; DO NOT EDIT.
# Linux/x86 5.10.229 Kernel Configuration
#
CONFIG_CC_VERSION_TEXT="gcc (Ubuntu 13.2.0-23ubuntu4) 13.2.0"
CONFIG_CC_IS_GCC=y
CONFIG_GCC_VERSION=130200
CONFIG_LD_VERSION=242000000
CONFIG_CLANG_VERSION=0
CONFIG_AS_IS_GNU=y
siradj@siradj:/dev$
```

Figure 3: config file path

Configuration Methods:

I will use the default configuration using this command `sudo make defconfig`

```
siradj@siradj:/usr/src/linux-5.10.229$ sudo make defconfig_
```

Figure 4: default config

Essential Configuration Options:

- **Processor type and features:** This option determines the type of CPU your system has and enables/disables various CPU features. It's necessary to ensure compatibility and optimal performance.
- **Clock and power management:** This option includes settings for power saving and frequency scaling. It's necessary to manage power consumption and heat generation, especially in laptops and mobile devices.
- **Memory management:** This option includes settings for managing system memory. It's necessary to ensure efficient memory usage and to prevent out-of-memory errors.
- **File systems:** This option includes settings for the file systems your kernel will support. It's necessary to include the file systems that your operating system and applications require.
- **Device drivers:** This option includes settings for the drivers of your system's devices. It's necessary to include the correct drivers for your hardware to function properly.
- **Networking:** This option includes settings for network protocols and drivers. It's necessary for network communication and internet connectivity.
- **Security:** This option includes settings for kernel security features. It's necessary to protect the system from unauthorized access and attacks.
- **Executable file formats:** This option includes settings for the types of executable files your kernel can run. It's necessary to include the formats that your applications use.
- **Kernel modules:** This option determines whether kernel modules are enabled or not. Kernel modules allow for dynamic loading and unloading of kernel code, which can be necessary for supporting certain hardware or features.
- **Debugging:** This option includes settings for kernel debugging features. It's necessary for troubleshooting and fixing issues in the kernel.

Kernel Compilation:

To clean previous build run this two commands `sudo make clean` and `sudo make mrproper`

1. After i run the default configuration command `sudo make defconfig`, i run this command `sudo make -j(nproc)` to compile the kernel using all cpu cores when compiling. After a few minutes i got this :

```
CC      arch/x86/boot/compressed/cpuflags.o
LD [M]  drivers/thermal/intel/x86_pkg_temp_thermal.ko
LD [M]  fs/efivarfs/efivarfs.ko
LD [M]  net/ipv4/netfilter/iptables_net.ko
LD [M]  net/ipv4/netfilter/nf_log_arp.ko
LD [M]  net/ipv4/netfilter/nf_log_ipv4.ko
LD [M]  net/ipv6/netfilter/nf_log_ipv6.ko
LD [M]  net/netfilter/nf_log_common.ko
LD [M]  net/netfilter/xt_LOG.ko
LD [M]  net/netfilter/xt_MASQUERADE.ko
LD [M]  net/netfilter/xt_addrtype.ko
CC      arch/x86/boot/compressed/early_serial_console.o
LD [M]  net/netfilter/xt_mark.ko
LD [M]  net/netfilter/xt_nat.ko
CC      arch/x86/boot/compressed/kaslr.o
CC      arch/x86/boot/pm.o
CC      arch/x86/boot/compressed/ldent_map_64.o
AS      arch/x86/boot/pmjump.o
CC      arch/x86/boot/printk.o
CC      arch/x86/boot/compressed/ldt_64.o
CC      arch/x86/boot/regs.o
CC      arch/x86/boot/string.o
AS      arch/x86/boot/compressed/ldt_handlers_64.o
CC      arch/x86/boot/tty.o
AS      arch/x86/boot/compressed/mem_encrypt.o
CC      arch/x86/boot/compressed/pgtable_64.o
CC      arch/x86/boot/video.o
CC      arch/x86/boot/video-mode.o
CC      arch/x86/boot/version.o
CC      arch/x86/boot/video-vesa.o
CC      arch/x86/boot/video-vesa.o
CC      arch/x86/boot/compressed/acpi.o
CC      arch/x86/boot/video-bios.o
HOSTCC  arch/x86/boot/tools/build
AS      arch/x86/boot/compressed/efi_thunk_64.o
CPUSTR  arch/x86/boot/cpustr.h
CC      arch/x86/boot/compressed/misc.o
GZIP    arch/x86/boot/compressed/vmlinux.bin.gz
CC      arch/x86/boot/cpu.o
WIPIDG  arch/x86/boot/compressed/piggy.S
AS      arch/x86/boot/compressed/piggy.o
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
kernel: arch/x86/boot/bzImage is ready (#1)
61rad18@rad18:~/usr/src/linux-5.10.229$
```


2. install modules : `sudo make modules_install`

```
siradj@siradj:/usr/src/linux-5.10.229$ sudo make modules_install
INSTALL drivers/thermal/intel/x86_pkg_temp_thermal.ko
INSTALL fs/efivarfs/efivarfs.ko
INSTALL net/ipv4/netfilter/iptables_nat.ko
INSTALL net/ipv4/netfilter/nf_log_arp.ko
INSTALL net/ipv4/netfilter/nf_log_ipv4.ko
INSTALL net/ipv6/netfilter/nf_log_ipv6.ko
INSTALL net/netfilter/nf_log_common.ko
INSTALL net/netfilter/xt_LOG.ko
INSTALL net/netfilter/xt_MASQUERADE.ko
INSTALL net/netfilter/xt_addrtype.ko
INSTALL net/netfilter/xt_mark.ko
INSTALL net/netfilter/xt_nat.ko
DEPMOD 5.10.229
siradj@siradj:/usr/src/linux-5.10.229$ _
```

3. Install kernel : sudo make install

```
siradj@siradj:/usr/src/linux-5.10.229$ sudo make install
sh ./arch/x86/boot/install.sh 5.10.229 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.10.229 /boot/vmlinuz-5.10.229
update-initramfs: Generating /boot/initrd.img-5.10.229
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.10.229 /boot/vmlinuz-5.10.229
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.10.229 /boot/vmlinuz-5.10.229
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.10.229 /boot/vmlinuz-5.10.229
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.171
I: /boot/initrd.img is now a symlink to initrd.img-5.10.229
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.10.229 /boot/vmlinuz-5.10.229
Sourcing file `./etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.8.0-48-generic
Found initrd image: /boot/initrd.img-6.8.0-48-generic
Found linux image: /boot/vmlinuz-5.15.171
Found initrd image: /boot/initrd.img-5.15.171
Found linux image: /boot/vmlinuz-5.10.229
Found initrd image: /boot/initrd.img-5.10.229
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.
Adding boot menu entry for UEFI Firmware Settings ...
done
siradj@siradj:/usr/src/linux-5.10.229$
```



```
GNU nano 7.2      grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux 5.10.229"
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`( . /etc/os-release; echo ${NAME:-Ubuntu} ) 2>/dev/null || echo`
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_CMDLINE_LINUX=""

# If your computer has multiple operating systems installed, then you
# probably want to run os-prober. However, if your computer is a host
# for guest OSes installed via LVM or raw disk devices, running
# os-prober can cause damage to those guest OSes as it mounts
# filesystems to look for things.
#GRUB_DISABLE_OS_PROBER=false

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
[ Read 40 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^\ Replace  ^U Paste     ^J Justify  ^_ Go To Line
```

Figure 6: After Modification

5. then update the grub command : `sudo update-grub`
6. Finally : reboot the system, command : `sudo reboot`

```
slradj@slradj:/usr/src/linux-5.10.229$ sudo update-grub
Sourcing file `/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.8.0-48-generic
Found initrd image: /boot/initrd.img-5.8.0-48-generic
Found linux image: /boot/vmlinuz-5.15.171
Found initrd image: /boot/initrd.img-5.15.171
Found linux image: /boot/vmlinuz-5.10.229
Found initrd image: /boot/initrd.img-5.10.229
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.
Adding boot menu entry for UEFI Firmware Settings ...
done
slradj@slradj:/usr/src/linux-5.10.229$ _
```

7. Result:

```
siradj@siradj:~$ uname -r  
5.8.0-48-generic  
siradj@siradj:~$
```

Figure 7: Original Kernel

```
siradj@siradj:~$ uname -r  
5.10.229  
siradj@siradj:~$
```

Figure 8: Custome Kernel