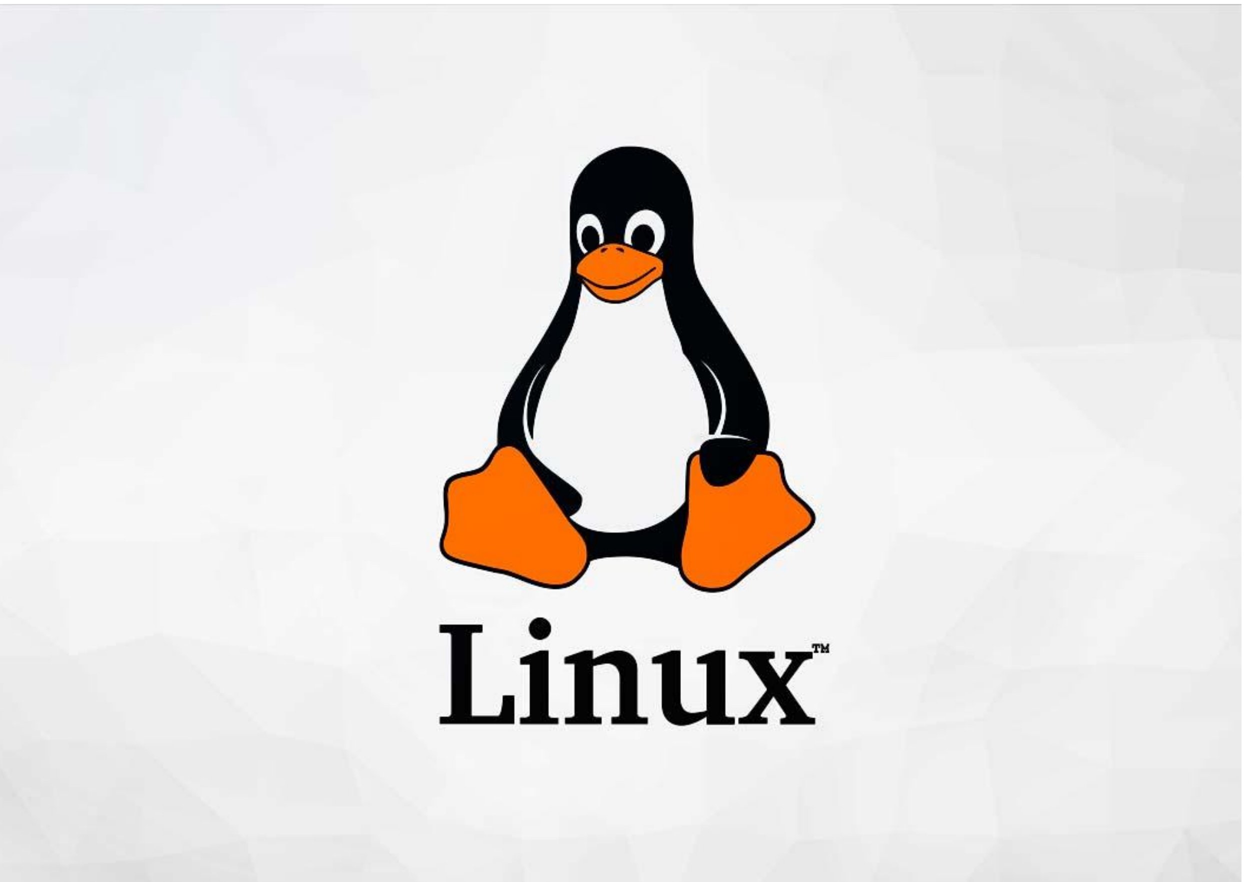
Assignment 1 - Report

CS3003D: Operating Systems



Hadif Yassin Hameed

B190513CS
CSE - B

Problem Statement

Download the latest stable Linux kernel from kernel.org, compile it, and dual boot it with your current Linux version. Your current version as well as the new version should be present in the grub-menu.

Introduction

Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net. It aims towards POSIX and Single UNIX Specification compliance. The Linux kernel is a free and open-source, monolithic, modular, multitasking, Unix-like operating system kernel.

Compiling the Linux kernel from source is a great exercise to deepen your understanding of the Linux kernel and operating systems in general. Doing so you can customize the default kernel to reflect your computer's precise specifications and make it function most efficiently.

Although the methodology remains the same, this document talks about configuring, building and installing the latest Linux kernel (5.14.1) on a UEFI system running an Arch based Linux distro, specifically Manjaro Linux, with kernel version 5.13.13.

Methodology

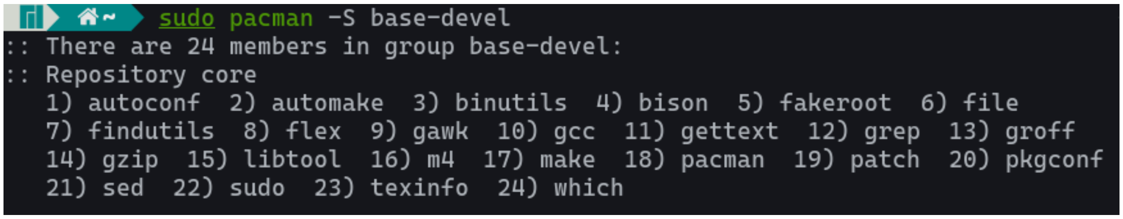
1. Install the core development packages.
2. Obtain the kernel source from kernel.org.
3. Fine tune the kernel configuration.
4. Compile the kernel.
5. Install the kernel and kernel modules.
6. Add bootloader entry for the new kernel.
7. Reboot into the newly installed kernel.

Explanation

1. Installing pre-requisites

Install the `base-devel` package group which contains packages necessary for the build process like `make` and `gcc`.

```
$     sudo pacman -S base-devel
```

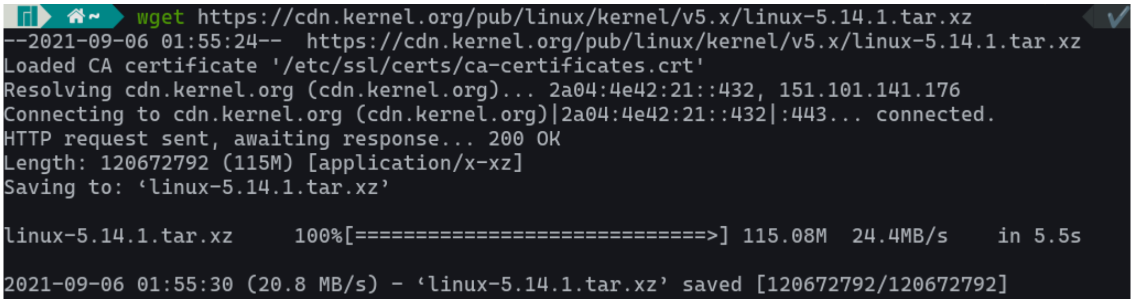


```
~ sudo pacman -S base-devel
:: There are 24 members in group base-devel:
:: Repository core
   1) autoconf  2) automake  3) binutils  4) bison  5) fakeroot  6) file
   7) findutils  8) flex  9) gawk  10) gcc  11) gettext  12) grep  13) groff
   14) gzip  15) libtool  16) m4  17) make  18) pacman  19) patch  20) pkgconf
   21) sed  22) sudo  23) texinfo  24) which
```

2. Obtaining the kernel Source

Download the source from kernel.org and unpack it. It is a good idea to verify the PGP signature of the downloaded tarball to ensure that it is legitimate but we will be skipping over that in this document.

```
$     wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.14.1.tar.xz
$     xz -cd linux-5.14.1.tar.xz | tar xvf -
$     cd linux-5.14.1
```

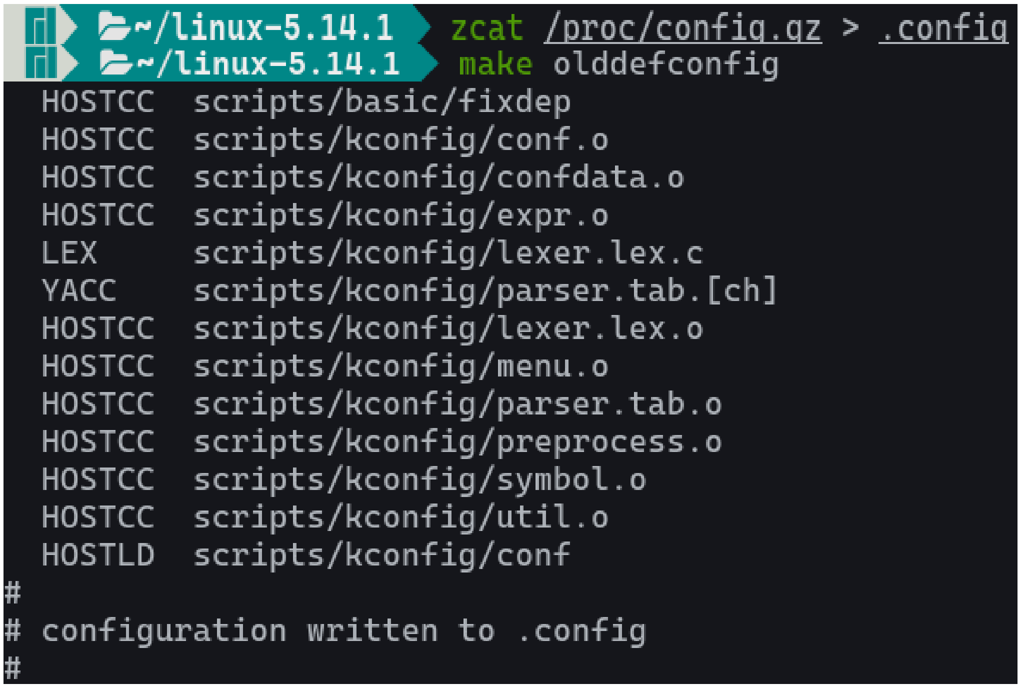


```
~ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.14.1.tar.xz
--2021-09-06 01:55:24--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.14.1.tar.xz
Loaded CA certificate '/etc/ssl/certs/ca-certificates.crt'
Resolving cdn.kernel.org (cdn.kernel.org)... 2a04:4e42:21::432, 151.101.141.176
Connecting to cdn.kernel.org (cdn.kernel.org)|2a04:4e42:21::432|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 120672792 (115M) [application/x-xz]
Saving to: ‘linux-5.14.1.tar.xz’

linux-5.14.1.tar.xz     100%[=============================>] 115.08M  24.4MB/s    in 5.5s

2021-09-06 01:55:30 (20.8 MB/s) - ‘linux-5.14.1.tar.xz’ saved [120672792/120672792]
```

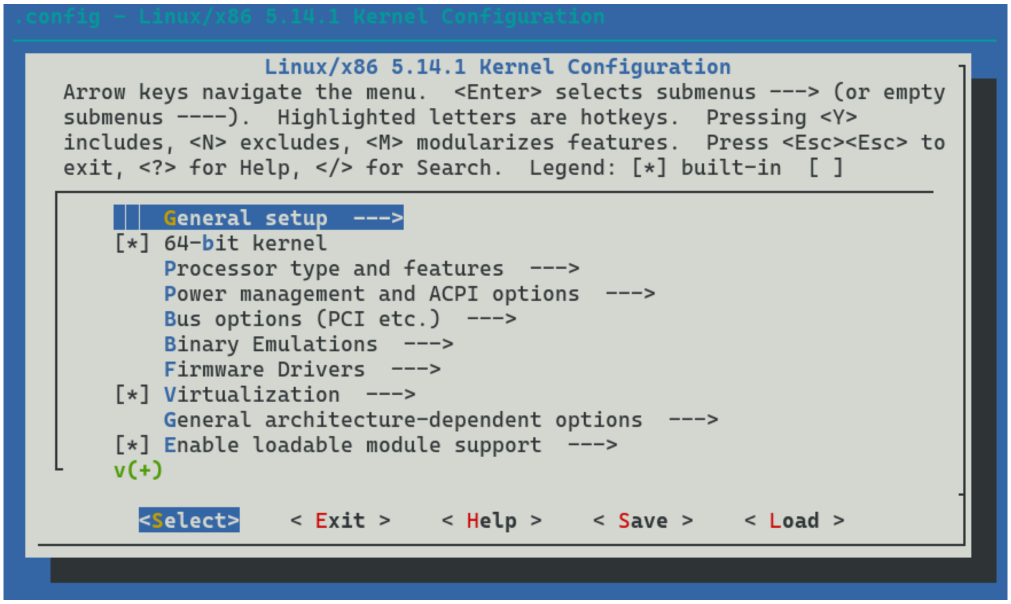
3. Kernel configuration

This is the most crucial step in customizing the default kernel to reflect the computer's precise specifications. Kernel configuration, including the use of kernel modules, is set in its `.config` file. By setting the options in `.config` properly, the kernel and computer will function most efficiently. We first copy the config file of the currently running kernel into the source directory and modify it to suit our needs. To do so by accepting default values for all new config flags use `make olddefconfig`. To fine-tune the configuration use `make menuconfig`.

```
$     zcat /proc/config.gz > .config
$     make olddefconfig
$     make menuconfig
```



```
~/linux-5.14.1  zcat /proc/config.gz > .config
~/linux-5.14.1  make olddefconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
```

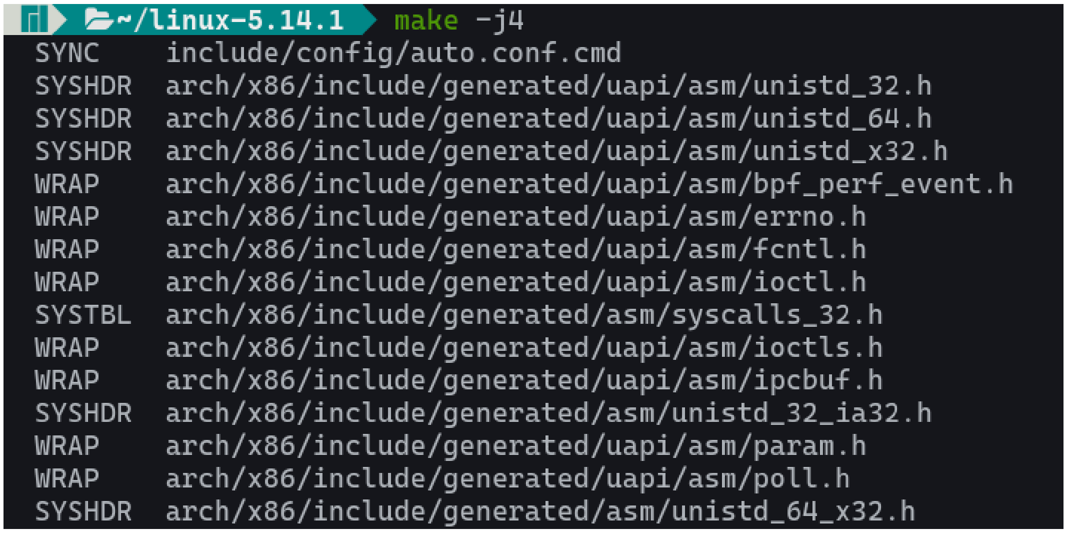


```
.config - Linux/x86 5.14.1 Kernel Configuration
                    Linux/x86 5.14.1 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

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

4. Compilation

Compilation time will vary depending on kernel configuration and processor capability. To exploit parallelism and compile faster use the `-jX` argument, `X` being the number of parallel jobs. The number of CPU cores in the machine (`$(nproc)`) usually works best. On the test machine it took 1 hour 29 minutes and printed 29000 lines of output.

```
$     make -j$(nproc)
```



```
 ~/linux-5.14.1  make -j4
  SYNC    include/config/auto.conf.cmd
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  WRAP    arch/x86/include/generated/uapi/asm/bpf_perf_event.h
  WRAP    arch/x86/include/generated/uapi/asm/errno.h
  WRAP    arch/x86/include/generated/uapi/asm/fcntl.h
  WRAP    arch/x86/include/generated/uapi/asm/ioctl.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_32.h
  WRAP    arch/x86/include/generated/uapi/asm/ioctls.h
  WRAP    arch/x86/include/generated/uapi/asm/ipcbuf.h
  SYSHDR  arch/x86/include/generated/asm/unistd_32_ia32.h
  WRAP    arch/x86/include/generated/uapi/asm/param.h
  WRAP    arch/x86/include/generated/uapi/asm/poll.h
  SYSHDR  arch/x86/include/generated/asm/unistd_64_x32.h
```

5. Installation

After compilation the kernel modules can be installed. This step needs to be done as the root user.

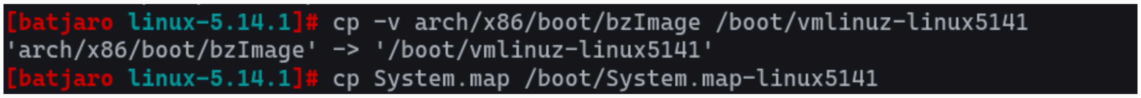
```
$     su
#     make modules_install
```

```
[batjaro linux-5.14.1]# make modules_install
```

This will copy the compiled modules into `lib/modules/5.14.1`.

The compressed bzImage (big zImage) of the kernel generated after compilation must be copied to the `/boot` directory and renamed. The generated `System.map` file must also be copied to the same directory. The `System.map` contains a list of kernel symbols and their corresponding addresses.

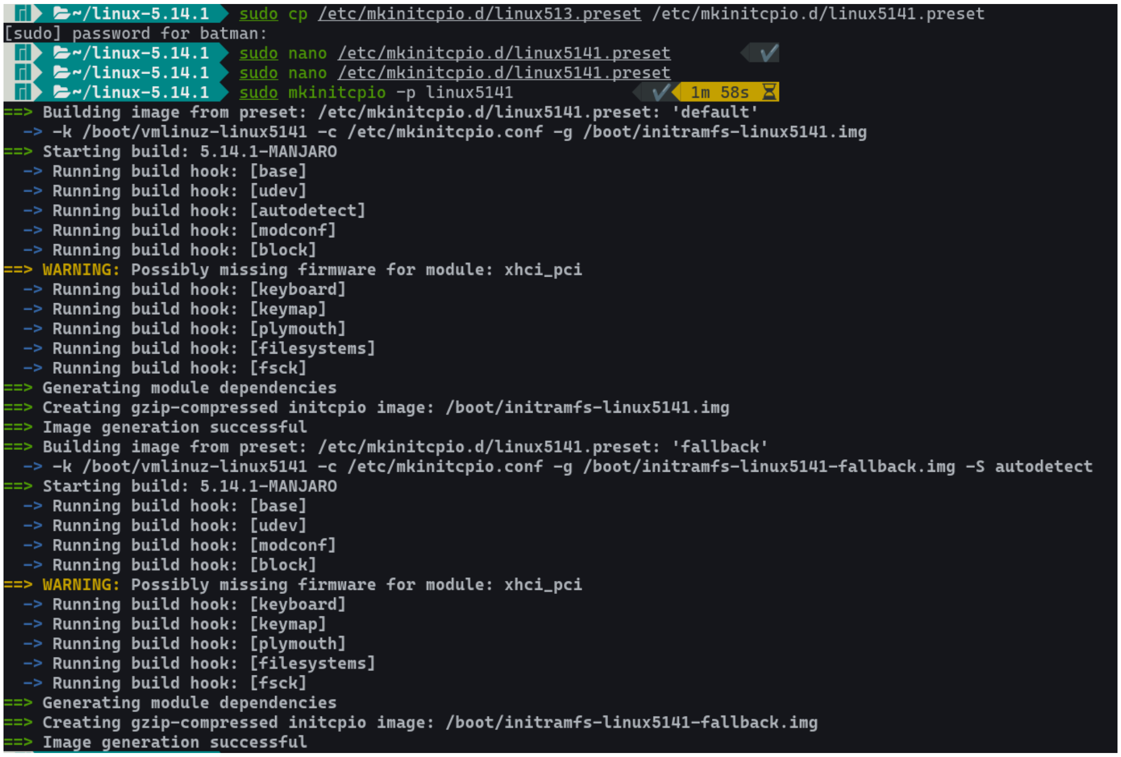
```
#     cp -v arch/x86/boot/bzImage /boot/vmlinuz-linux5141
#     cp System.map /boot/System.map-linux5141
```



```
[batjaro linux-5.14.1]# cp -v arch/x86/boot/bzImage /boot/vmlinuz-linux5141
'arch/x86/boot/bzImage' -> '/boot/vmlinuz-linux5141'
[batjaro linux-5.14.1]# cp System.map /boot/System.map-linux5141
```

The initial RAM disk for the kernel can be created by generating `initramfs` images from a modified `mkinitcpio` preset. `initramfs` is a scheme to load a temporary root file system into memory on OS startup. Update the version numbers in the copied preset and generate the `initramfs` images.

```
#     cp /etc/mkinitcpio.d/linux513.preset /etc/mkinitcpio.d/linux5141.preset
#     nano /etc/mkinitcpio.d/linux5141.preset
#     mkinitcpio -p linux5141
```

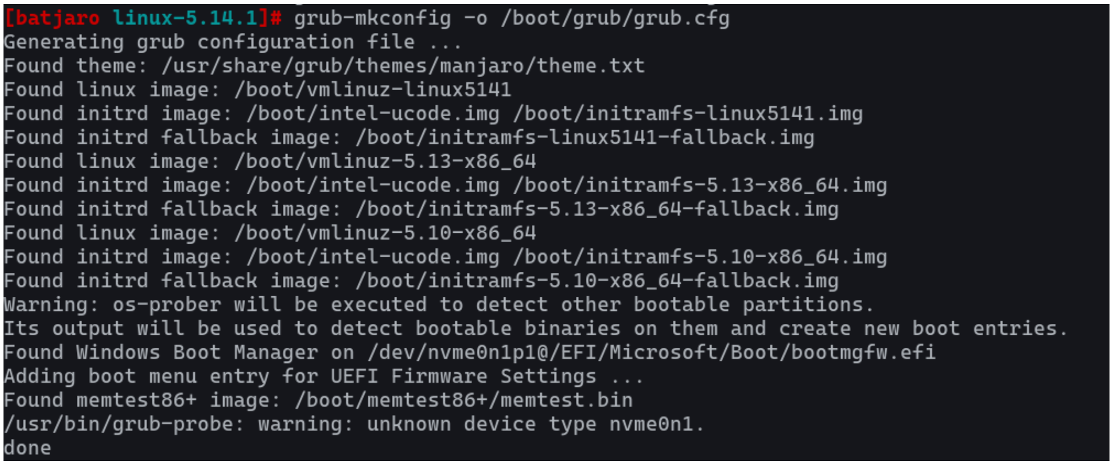


```
 ~/linux-5.14.1  sudo cp /etc/mkinitcpio.d/linux513.preset /etc/mkinitcpio.d/linux5141.preset
[sudo] password for batman:
 ~/linux-5.14.1  sudo nano /etc/mkinitcpio.d/linux5141.preset        ✓
 ~/linux-5.14.1  sudo nano /etc/mkinitcpio.d/linux5141.preset
 ~/linux-5.14.1  sudo mkinitcpio -p linux5141                    ✓ 1m 58s ⌛
==> Building image from preset: /etc/mkinitcpio.d/linux5141.preset: 'default'
  -> -k /boot/vmlinuz-linux5141 -c /etc/mkinitcpio.conf -g /boot/initramfs-linux5141.img
==> Starting build: 5.14.1-MANJARO
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [autodetect]
  -> Running build hook: [modconf]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: xhci_pci
  -> Running build hook: [keyboard]
  -> Running build hook: [keymap]
  -> Running build hook: [plymouth]
  -> Running build hook: [filesystems]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux5141.img
==> Image generation successful
==> Building image from preset: /etc/mkinitcpio.d/linux5141.preset: 'fallback'
  -> -k /boot/vmlinuz-linux5141 -c /etc/mkinitcpio.conf -g /boot/initramfs-linux5141-fallback.img -S autodetect
==> Starting build: 5.14.1-MANJARO
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [modconf]
  -> Running build hook: [block]
==> WARNING: Possibly missing firmware for module: xhci_pci
  -> Running build hook: [keyboard]
  -> Running build hook: [keymap]
  -> Running build hook: [plymouth]
  -> Running build hook: [filesystems]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating gzip-compressed initcpio image: /boot/initramfs-linux5141-fallback.img
==> Image generation successful
```

6. Adding GRUB entry

`grub-mkconfig` will scan your drives for bootable operating systems and generate a GRUB configuration file with the appropriate entries.

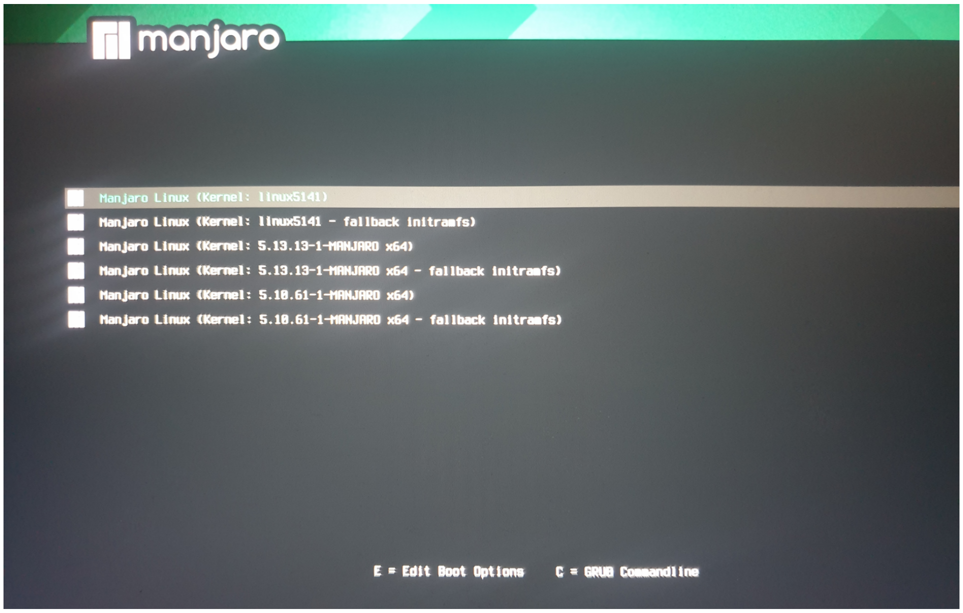
```
#     grub-mkconfig -o /boot/grub/grub.cfg
```



```
[batjaro linux-5.14.1]# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found theme: /usr/share/grub/themes/manjaro/theme.txt
Found linux image: /boot/vmlinuz-linux5141
Found initrd image: /boot/intel-ucode.img /boot/initramfs-linux5141.img
Found initrd fallback image: /boot/initramfs-linux5141-fallback.img
Found linux image: /boot/vmlinuz-5.13-x86_64
Found initrd image: /boot/intel-ucode.img /boot/initramfs-5.13-x86_64.img
Found initrd fallback image: /boot/initramfs-5.13-x86_64-fallback.img
Found linux image: /boot/vmlinuz-5.10-x86_64
Found initrd image: /boot/intel-ucode.img /boot/initramfs-5.10-x86_64.img
Found initrd fallback image: /boot/initramfs-5.10-x86_64-fallback.img
Warning: os-prober will be executed to detect other bootable partitions.
Its output will be used to detect bootable binaries on them and create new boot entries.
Found Windows Boot Manager on /dev/nvme0n1p1@/EFI/Microsoft/Boot/bootmgfw.efi
Adding boot menu entry for UEFI Firmware Settings ...
Found memtest86+ image: /boot/memtest86+/memtest.bin
/usr/bin/grub-probe: warning: unknown device type nvme0n1.
done
```

7. **Booting into the new kernel.**

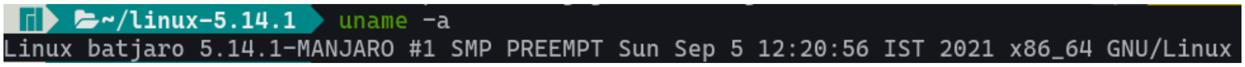
On reboot the system will try to boot into the new kernel by default. To choose between installed kernel versions, select `Advanced options for Manjaro Linux`.



```
Manjaro Linux (Kernel: linux5141)
Manjaro Linux (Kernel: linux5141 - fallback initramfs)
Manjaro Linux (Kernel: 5.13.13-1-MANJARO x64)
Manjaro Linux (Kernel: 5.13.13-1-MANJARO x64 - fallback initramfs)
Manjaro Linux (Kernel: 5.10.61-1-MANJARO x64)
Manjaro Linux (Kernel: 5.10.61-1-MANJARO x64 - fallback initramfs)
```

To check the currently running kernel version, run:

```
$    uname -a
```



```
~/linux-5.14.1  uname -a
Linux batjaro 5.14.1-MANJARO #1 SMP PREEMPT Sun Sep 5 12:20:56 IST 2021 x86_64 GNU/Linux
```

Conclusion

We have successfully configured and compiled the latest Linux kernel from source. The compiled kernel was installed and is now the default kernel on the system.

References

https://en.wikipedia.org/wiki/Linux_kernel

https://wiki.archlinux.org/title/Kernel/Traditional_compilation

https://www.kernel.org/doc/html/latest/admin-guide/README.html

https://wiki.archlinux.org/title/Arch_boot_process#Feature_comparison