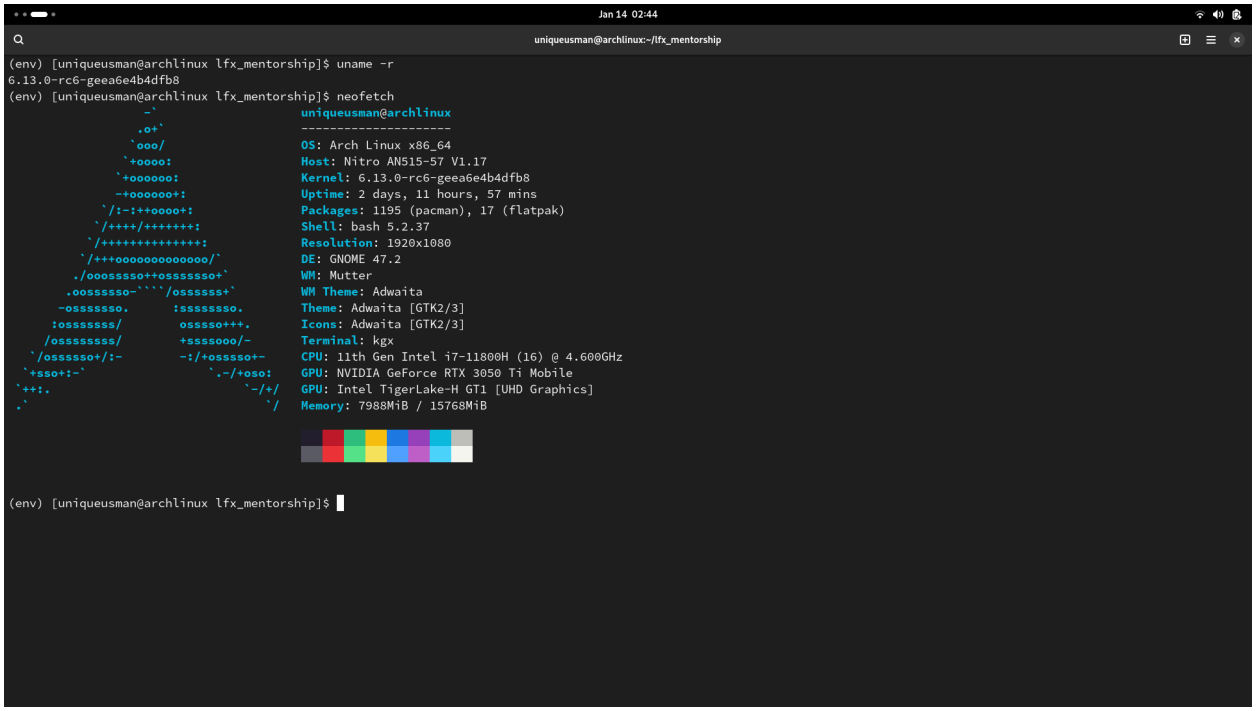
# Configure, Build, Install, Customize, and Test a Linux Kernel - My Whole Process.

I actually just built a Linux kernel right from the source a few days ago. But, it is the mainline one. The version 6.13.0(Linus Torvald Github).




For this task, I will be using the stable 6.12.9 version.

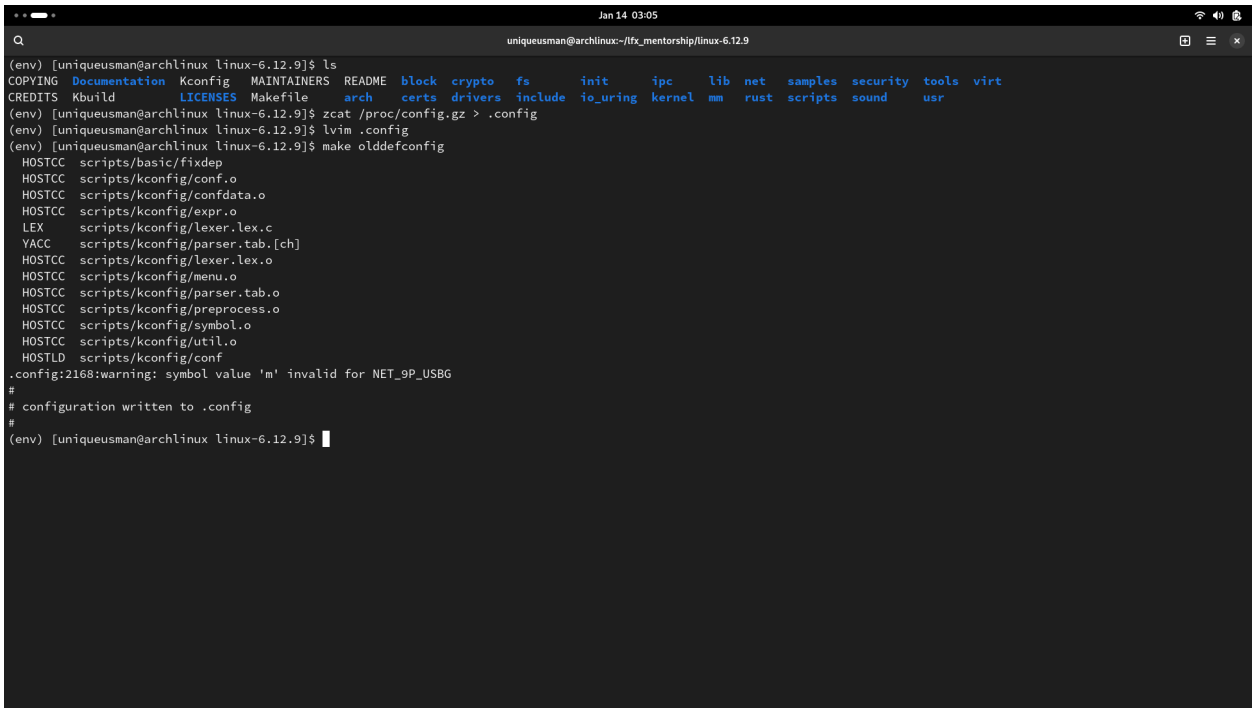
1. Downloaded the Kernel.


When I did **make olddefconfig** command, I got an error,

.config:2168:warning: symbol value 'm' invalid for NET_9P_USBG

I think the problem was that, my system is running on Kernel 6.13.0 and the CONFIG_NET_9P_USBG option is configured as a module (m) in my 6.13.0 .config file, but this setting isn't valid for the Linux kernel version 6.12.9. In kernel 6.12.9, CONFIG_NET_9P_USBG is defined as a boolean (bool), meaning it can only be set to y (built-in) or n (disabled).

This discrepancy likely stems from me using a configuration file from kernel version 6.13.0, where CONFIG_NET_9P_USBG is defined as a tristate (tristate), allowing it to be set to y, m, or n (https://cateee.net/lkddb/web-lkddb/NET_9P_USBG.html)




I was able to fix it by adjusting the .config file and setting CONFIG_NET_9P_USBG to y before the **make olddefconfig** command.

2. After running make -j6 all “A screenshot of the successful build message: Kernel: arch/x86/boot/bzImage is
   Ready.”

```
Jan 14  04:08
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9
  BTF [M] net/vmw_vsock/vsock_diag.ko
  BTF [M] net/vmw_vsock/vsock.ko
  LD [M]  net/vmw_vsock/vmw_vsock_vmci_transport.ko
  BTF [M] net/openvswitch/openvswitch.ko
  LD [M]  net/vmw_vsock/vmw_vsock_virtio_transport.ko
  LD [M]  net/vmw_vsock/vmw_vsock_virtio_transport_common.ko
  LD [M]  net/vmw_vsock/hv_sock.ko
  BTF [M] net/vmw_vsock/vmw_vsock_vmci_transport.ko
  BTF [M] net/vmw_vsock/vmw_vsock_virtio_transport.ko
  BTF [M] net/vmw_vsock/hv_sock.ko
  BTF [M] net/vmw_vsock/vmw_vsock_virtio_transport_common.ko
  LD [M]  net/vmw_vsock/vsock_loopback.ko
  BTF [M] net/vmw_vsock/vsock_loopback.ko
  LD [M]  net/nsh/nsh.ko
  LD [M]  net/hsr/hsr.ko
  LD [M]  net/qrtr/qrtr.ko
  LD [M]  net/qrtr/qrtr-smd.ko
  LD [M]  net/qrtr/qrtr-tun.ko
  LD [M]  net/qrtr/qrtr-mhi.ko
  BTF [M] net/hsr/hsr.ko
  BTF [M] net/qrtr/qrtr.ko
  BTF [M] net/qrtr/qrtr-smd.ko
  BTF [M] net/nsh/nsh.ko
  BTF [M] net/qrtr/qrtr-tun.ko
  BTF [M] net/qrtr/qrtr-mhi.ko
(env) [uniqueusman@archlinux linux-6.12.9]$ make -j6 all
mkdir -p /home/uniqueusman/lfx_mentorship/linux-6.12.9/tools/objtool && make O=/home/uniqueusman/lfx_mentorship/linux-6.12.9 subdir=tools/objtool --no-print-directory -C obj
tool
mkdir -p /home/uniqueusman/lfx_mentorship/linux-6.12.9/tools/bpf/resolve_btfids && make O=/home/uniqueusman/lfx_mentorship/linux-6.12.9 subdir=tools/bpf/resolve_btfids --no-
print-directory -C bpf/resolve_btfids
  INSTALL libsubcmd_headers
  INSTALL libsubcmd_headers
  CALL    scripts/checksyscalls.sh
  CHK     kernel/kheaders_data.tar.xz
  TEST    posttest
arch/x86/tools/insn_decoder_test: success: Decoded and checked 7802951 instructions
  TEST    posttest
arch/x86/tools/insn_sanity: Success: decoded and checked 1000000 random instructions with 0 errors (seed:0x9406d38d)
Kernel: arch/x86/boot/bzImage is ready  (#1)
(env) [uniqueusman@archlinux linux-6.12.9]$
```

3. I tried to Install the Kernel I got this error cannot find LILO

```
Jan 14  04:25
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vsock_diag.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vsock_diag.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vsock_diag.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_vmci_transport.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_vmci_transport.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_vmci_transport.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_virtio_transport.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_virtio_transport.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_virtio_transport.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_virtio_transport_common.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_virtio_transport_common.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vmw_vsock_virtio_transport_common.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/hv_sock.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/hv_sock.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/hv_sock.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vsock_loopback.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vsock_loopback.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/vmw_vsock/vsock_loopback.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/nsh/nsh.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/nsh/nsh.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/nsh/nsh.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/hsr/hsr.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/hsr/hsr.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/hsr/hsr.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-smd.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-smd.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-smd.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-tun.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-tun.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-tun.ko.zst
  INSTALL /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-mhi.ko
  SIGN    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-mhi.ko
  ZSTD    /lib/modules/6.12.9-usmanakinyemi/kernel/net/qrtr/qrtr-mhi.ko.zst
  DEPMOD  /lib/modules/6.12.9-usmanakinyemi
  INSTALL /boot
Cannot find LILO.
(env) [uniqueusman@archlinux linux-6.12.9]$
```

Basically, LILO is a boot loader like GRUB. I think this particular error is specific to Arch Linux Users. So, I will have to include some other steps.

sudo install -Dm644 "$(make -s image_name)" /boot/vmlinuz-6.12.9-usmanakinyemi

```
Cannot find LILO.
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo install -Dm644 "$(make -s image_name)" /boot/vmlinuz-6.12.9-usmanakinyemi
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo cp -vf System.map /boot/System.map-6.12.9-usmanakinyemi
'System.map' -> '/boot/System.map-6.12.9-usmanakinyemi'
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo cp -vf .config /boot/config-6.12.9-usmanakinyemi
'.config' -> '/boot/config-6.12.9-usmanakinyemi'
(env) [uniqueusman@archlinux linux-6.12.9]$ ~
```

4. I will have to generate the initial ramdisk, I think this is not needed on other distros.

```
Jan 14 04:44
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9
(env) [uniqueusman@archlinux linux-6.12.9]$ lvim /etc/mkinitcpio.d/linux-usmanakinyemi.preset
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo lvim /etc/mkinitcpio.d/linux-usmanakinyemi.preset
sudo: lvim: command not found
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo vim /etc/mkinitcpio.d/linux-usmanakinyemi.preset
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo mkinitcpio -p linux-usmanakinyemi
/etc/mkinitcpio.d/linux-usmanakinyemi.preset: line 7: unexpected EOF while looking for matching `"'
==> ERROR: Failed to load preset: '/etc/mkinitcpio.d/linux-usmanakinyemi.preset'
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo vim /etc/mkinitcpio.d/linux-usmanakinyemi.preset
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo vim /etc/mkinitcpio.d/linux-usmanakinyemi.preset
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo vim /etc/mkinitcpio.d/linux-usmanakinyemi.preset
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo mkinitcpio -p linux-usmanakinyemi
/etc/mkinitcpio.d/linux-usmanakinyemi.preset: line 7: unexpected EOF while looking for matching `"'
==> ERROR: Failed to load preset: '/etc/mkinitcpio.d/linux-usmanakinyemi.preset'
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo vim /etc/mkinitcpio.d/linux-usmanakinyemi.preset
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo mkinitcpio -p linux-usmanakinyemi
==> Building image from preset: /etc/mkinitcpio.d/linux-usmanakinyemi.preset: 'default'
==> Using configuration file: '/etc/mkinitcpio.conf'
  -> -k /boot/vmlinuz-6.12.9-usmanakinyemi -c /etc/mkinitcpio.conf -g /boot/initramfs-6.12.9-usmanakinyemi.img
==> Starting build: '6.12.9-usmanakinyemi'
  -> Running build hook: [base]
  -> Running build hook: [udev]
  -> Running build hook: [autodetect]
  -> Running build hook: [microcode]
  -> Running build hook: [modconf]
  -> Running build hook: [kms]
  -> Running build hook: [keyboard]
  -> Running build hook: [keymap]
  -> Running build hook: [consolefont]
==> WARNING: consolefont: no font found in configuration
  -> Running build hook: [block]
  -> Running build hook: [encrypt]
==> WARNING: Possibly missing firmware for module: 'qat_420xx'
  -> Running build hook: [lvm2]
  -> Running build hook: [filesystems]
  -> Running build hook: [fsck]
==> Generating module dependencies
==> Creating zstd-compressed initcpio image: '/boot/initramfs-6.12.9-usmanakinyemi.img'
  -> Early uncompressed CPIO image generation successful
==> Initcpio image generation successful
==> Building image from preset: /etc/mkinitcpio.d/linux-usmanakinyemi.preset: 'fallback'
```

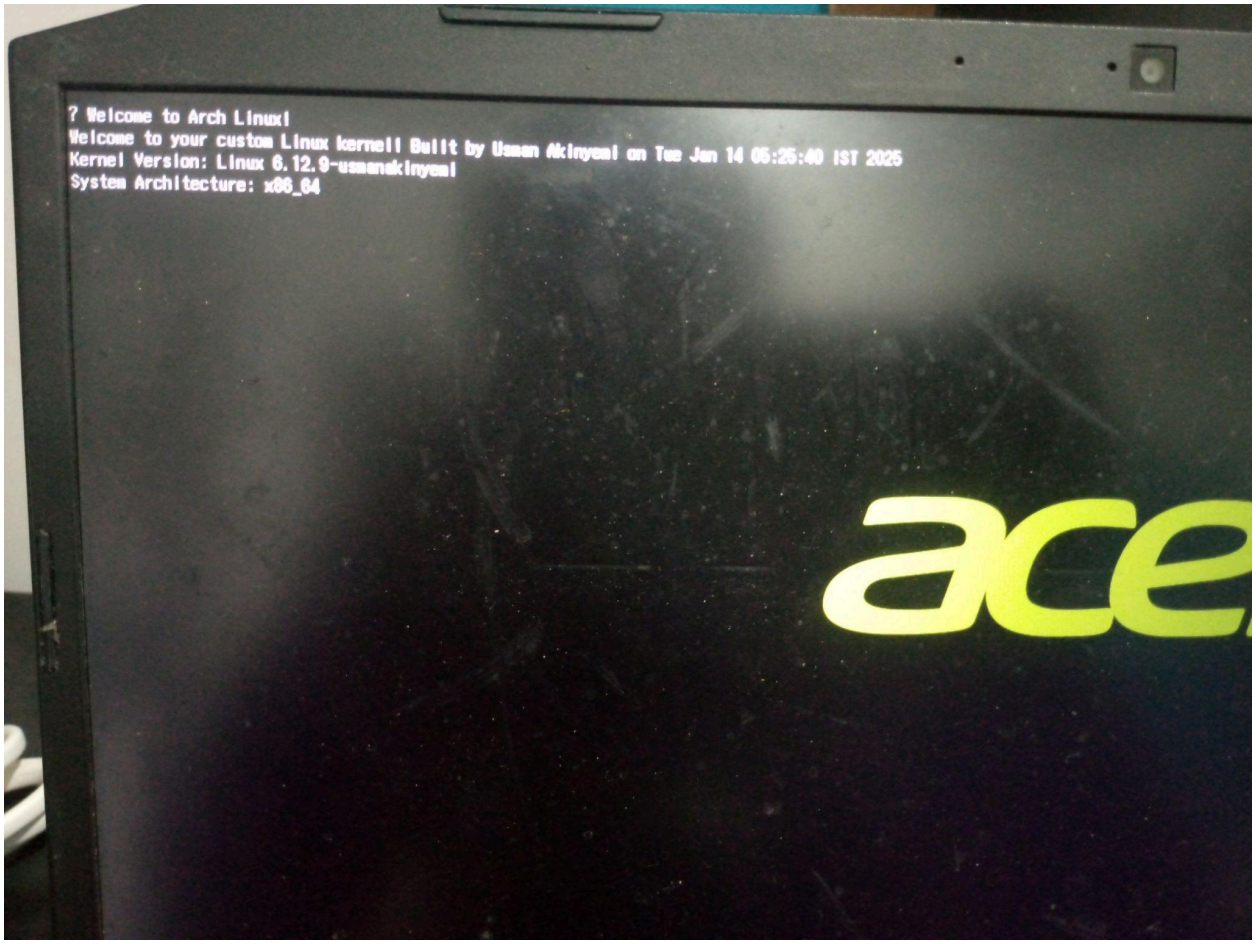
5. Updating the grub

```
Jan 14 04
uniqueusman@archlinux:~/lfx
(env) [uniqueusman@archlinux linux-6.12.9]$ sudo grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux-lts
Found initrd image: /boot/initramfs-linux-lts.img
Found fallback initrd image(s) in /boot:  initramfs-linux-lts-fallback.img
Found linux image: /boot/vmlinuz-linux
Found initrd image: /boot/initramfs-linux.img
Found fallback initrd image(s) in /boot:  initramfs-linux-fallback.img
Found linux image: /boot/vmlinuz-6.13.0-rc6-geea6e4b4dfb8
Found initrd image: /boot/initramfs-6.13.0-rc6-geea6e4b4dfb8.img
Found linux image: /boot/vmlinuz-6.12.9-usmanakinyemi
Found initrd image: /boot/initramfs-6.12.9-usmanakinyemi.img
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
```

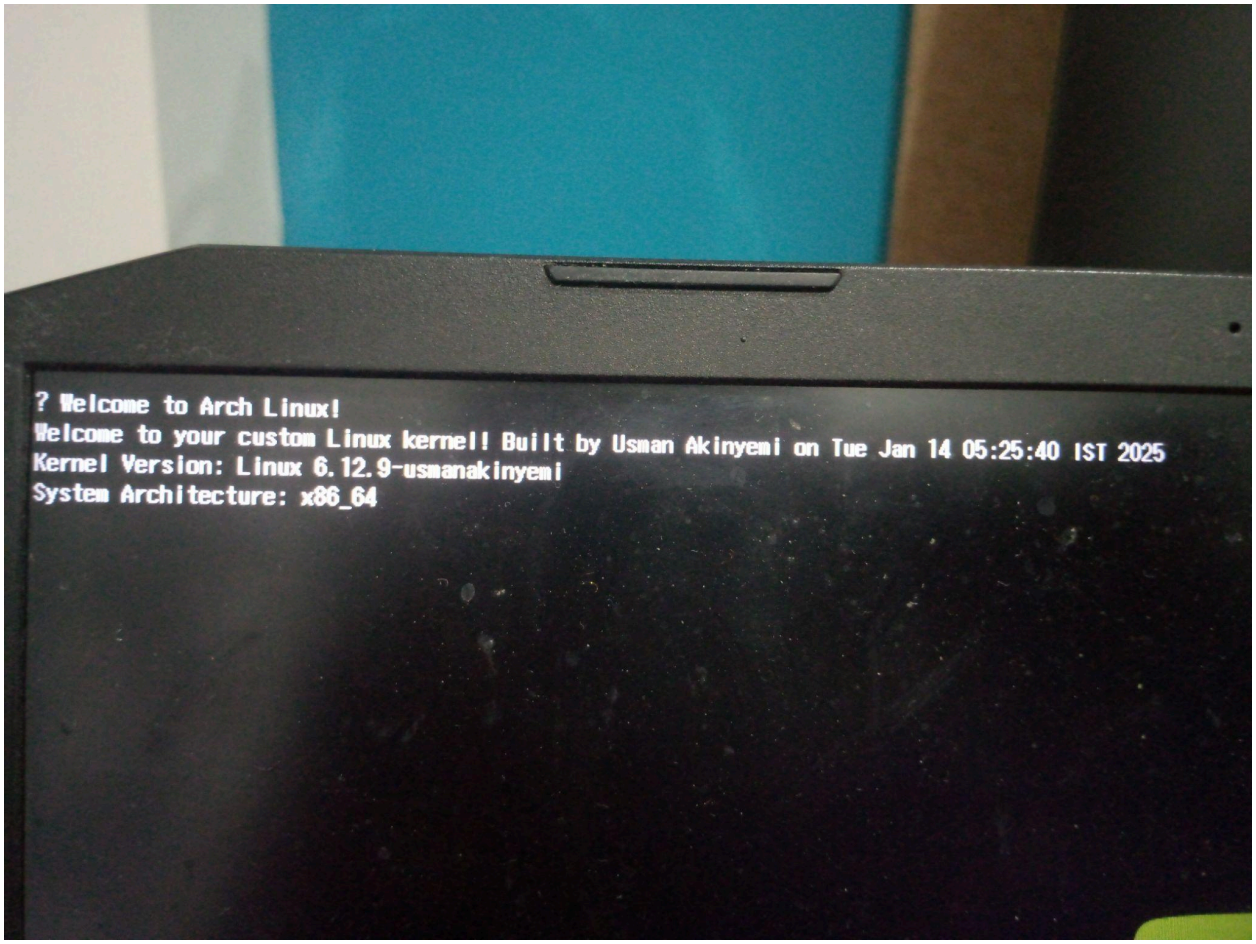
6. Customizing the Boot Message.

```
uniqueusman@archlinux:~
[uniqueusman@archlinux ~]$ lvim /tftpboot/linux-install/msgs/boot.msg
[uniqueusman@archlinux ~]$ sudo rm -rf /tftpboot/linux-install/msgs/boot.msg
[sudo] password for uniqueusman:
[uniqueusman@archlinux ~]$ ls
ALXExpert  Downloads  SuperVaani  a.py                   alx-interview  kb.c                linux_work      share    temp-repo
Android    Music      Templates   a.txt                  c.txt          leetcode_king_maker pdfai           sphere   test
Desktop    Pictures   Videos      alx-backend-javascript git            lfx_mentorship      ping.txt        systemd  website-redesign
Documents  Public     a.css       alx-backend-user-data  grep           libreoffice         recipe-backend  temp
[uniqueusman@archlinux ~]$ sudo vim /etc/grub.d/40_custom
[uniqueusman@archlinux ~]$ sudo grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux-lts
Found initrd image: /boot/initramfs-linux-lts.img
Found fallback initrd image(s) in /boot:  initramfs-linux-lts-fallback.img
Found linux image: /boot/vmlinuz-linux
Found initrd image: /boot/initramfs-linux.img
Found fallback initrd image(s) in /boot:  initramfs-linux-fallback.img
Found linux image: /boot/vmlinuz-6.13.0-rc6-geea6e4b4dfb8
Found initrd image: /boot/initramfs-6.13.0-rc6-geea6e4b4dfb8.img
Found linux image: /boot/vmlinuz-6.12.9-usmanakinyemi
Found initrd image: /boot/initramfs-6.12.9-usmanakinyemi.img
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
[uniqueusman@archlinux ~]$
```

For the customization, it really took me a lot of learning and trying different things to figure it out.

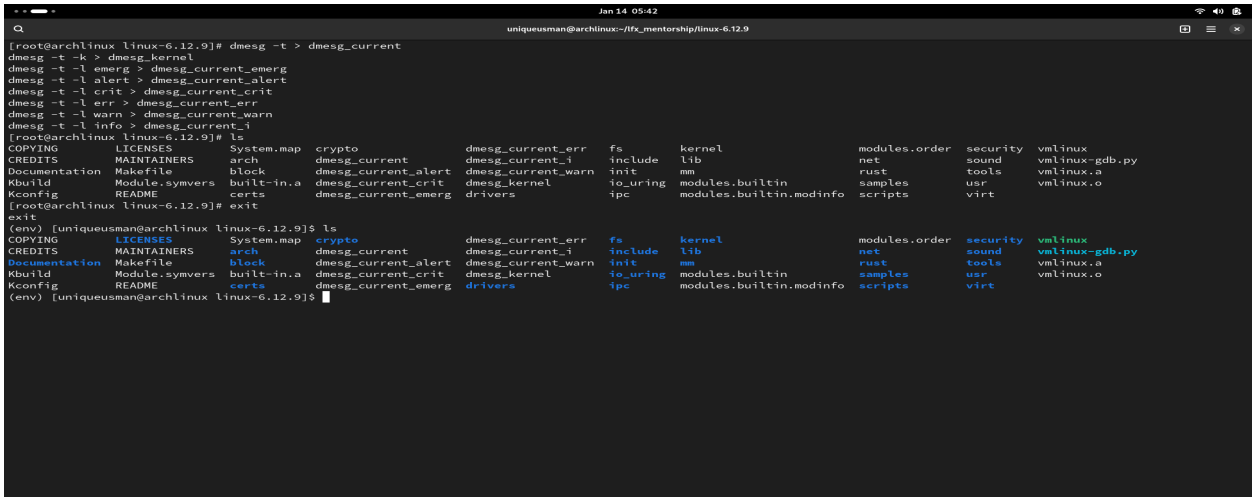



Since I am on Arch Linux PC, the instruction on the challenge pdf provided does not really work. I tried using a systemd service for it. I tried using plymouth also though it works but, I do not like the result. But, I was later able to configure it using the **/etc/grub.d/40_custom** which is provided by Grub and can be configured to show boot messages.

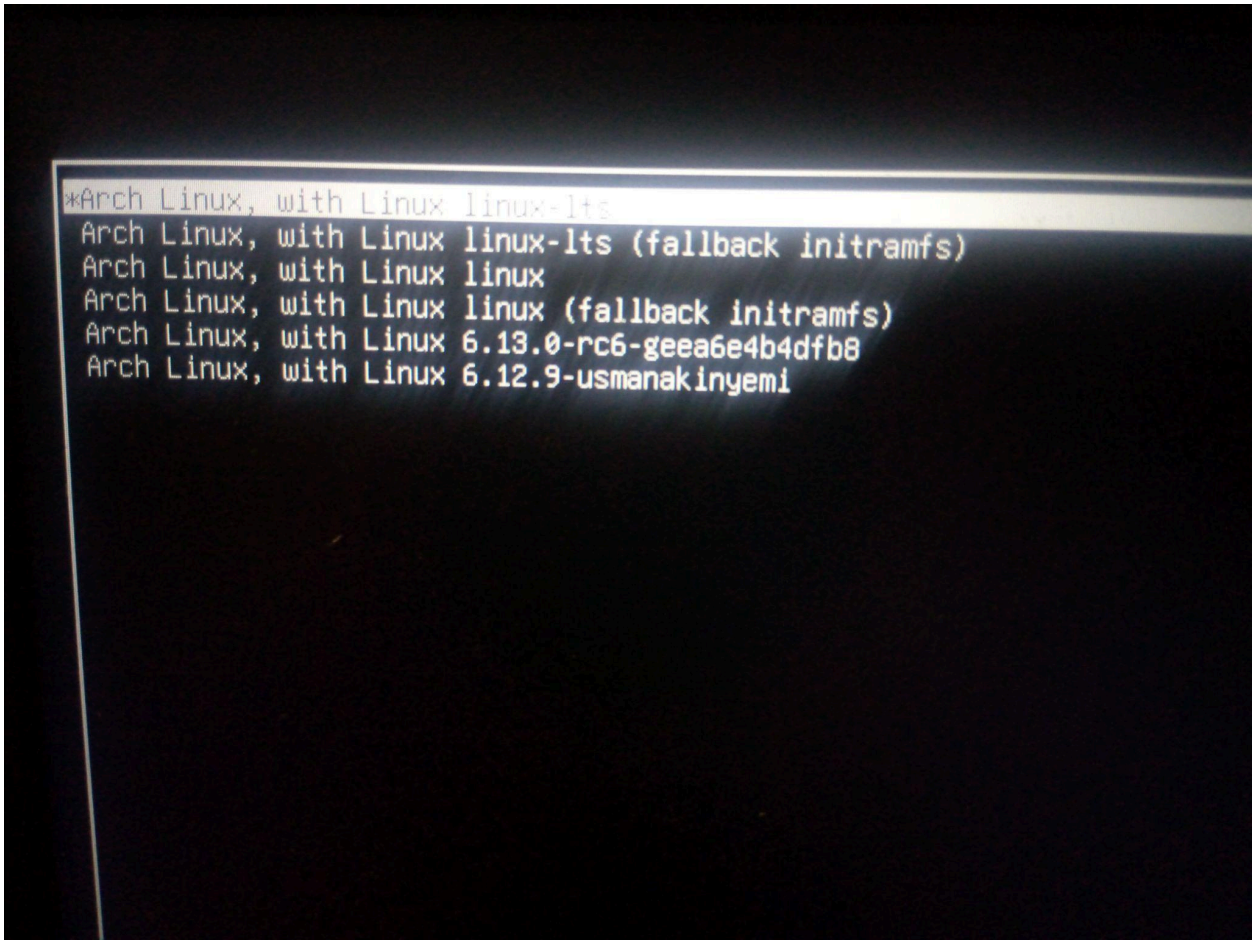



I added extra-information about the kernel and also the architecture.

7. Before rebooting, there is something that could help in ensuring that the new kernel was installed properly. And it is saving the logs from the current kernel to compare with the new kernel, look for regressions and new errors using dmesg.




8. After rebooting, the kernel got installed perfectly




Since I have multiple Kernel, I have to choose the new one I just built - 6.12.9-usmanakinyemi


9. Before running the KSelfTest, remember we saved the dmesg of the previous kernel, let's generate for the new kernel and compare the logs to ensure no new errors and that the kernel indeed builds well.

```
Jan 15 02:11
uniqueusman@archlinux:/home/uniqueusman/lfx_mentorship/linux-6.12.9
uniqueusman@archlinux:/home/uniqueusman/lfx_mentorship/linux-6.12.9    uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9/tools/testing/selftests
[uniqueusman@archlinux linux-6.12.9]$ dmesg -t > dmesg_new
dmesg -t -k > dmesg_new_kernel
dmesg -t -l emerg > dmesg_new_emerg
dmesg -t -l alert > dmesg_new_alert
dmesg -t -l crit > dmesg_new_crit
dmesg -t -l err > dmesg_new_err
dmesg -t -l warn > dmesg_new_warn
dmesg -t -l info > dmesg_new_info
dmesg: read kernel buffer failed: Operation not permitted
dmesg: read kernel buffer failed: Operation not permitted
dmesg: read kernel buffer failed: Operation not permitted
dmesg: read kernel buffer failed: Operation not permitted
dmesg: read kernel buffer failed: Operation not permitted
dmesg: read kernel buffer failed: Operation not permitted
dmesg: read kernel buffer failed: Operation not permitted
dmesg: read kernel buffer failed: Operation not permitted
[uniqueusman@archlinux linux-6.12.9]$ su
Password:
[root@archlinux linux-6.12.9]# dmesg -t > dmesg_new
dmesg -t -k > dmesg_new_kernel
dmesg -t -l emerg > dmesg_new_emerg
dmesg -t -l alert > dmesg_new_alert
dmesg -t -l crit > dmesg_new_crit
dmesg -t -l err > dmesg_new_err
dmesg -t -l warn > dmesg_new_warn
dmesg -t -l info > dmesg_new_info
[root@archlinux linux-6.12.9]#
```

```
[root@archlinux linux-6.12.9]# diff dmesg_current dmesg_new
diff dmesg_current_err dmesg_new_err
diff dmesg_current_warn dmesg_new_warn
1,2022c1,1900
< usb 3-5: device not accepting address 100, error -71
< usb 3-5: WARN: invalid context state for evaluate context command.
< usb usb3-port5: unable to enumerate USB device
< usb 3-5: new full-speed USB device number 101 using xhci_hcd
< usb 3-5: device descriptor read/64, error -71
< usb 3-5: device descriptor read/64, error -71
< usb 3-5: new full-speed USB device number 102 using xhci_hcd
< usb 3-5: device descriptor read/64, error -71
< usb 3-5: device descriptor read/64, error -71
< usb usb3-port5: attempt power cycle
< usb 3-5: new full-speed USB device number 103 using xhci_hcd
< usb 3-5: Device not responding to setup address.
```

While there are some differences between the logs, they are mostly due to the fact that the logs are from two different Kernel versions, remember the old log was linux-6.13.0 and the new one was linux-6.12.9. But, in many cases, this can be very useful for debugging the newly installed Kernel and also ensuring they work well.

10. Running Kernel Test (KSelfTest)

```
[uniqueusman@archlinux selftests]$ sudo make run_tests > test_results.txt 2>&1
[sudo] password for uniqueusman:
zhello, world!
[uniqueusman@archlinux selftests]$
```

```
Jan 14 23:17
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9/tools/testing/selftests
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9/tools/testing/selftests
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9/tools/testing/selftests
[uniqueusman@archlinux selftests]$ tail -36 test_results.txt
# supported algs: lzo-rle lzo lz4 lz4hc [zstd] deflate 842
# /sys/block/zram0/comp_algorithm = 'lzo'
# zram set compression algorithm: OK
# set disk size to zram device(s)
# /sys/block/zram0/disksize = '2097152'
# zram set disksizes: OK
# set memory limit to zram device(s)
# /sys/block/zram0/mem_limit = '2M'
# zram set memory limit: OK
# make ext4 filesystem on /dev/zram0
# zram mkfs.ext4: OK
# mount /dev/zram0
# zram mount of zram device(s): OK
# fill zram0...
# zram0 can be filled with '1900' KB
# zram compression ratio: 18.26:1: OK
# zram cleanup
# zram01 : [PASS]
#
# create '1' zram device(s)
# all zram devices (/dev/zram0~0) successfully created
# set max_comp_streams to zram device(s)
# The device attribute max_comp_streams was deprecated in 4.7
# set disk size to zram device(s)
# /sys/block/zram0/disksize = '1048576'
# zram set disksizes: OK
# set memory limit to zram device(s)
# /sys/block/zram0/mem_limit = '1M'
# zram set memory limit: OK
# make swap with zram device(s)
# done with /dev/zram0
# zram making zram mkswap and swapon: OK
# zram swapoff: OK
# zram cleanup
# zram02 : [PASS]
ok 1 selftests: zram: zram.sh
[uniqueusman@archlinux selftests]$
```

The kernel selftests completed successfully, confirming that the system's key features and security mechanisms are functioning as expected. Detailed results were provided for each test, highlighting both successful checks and skipped tests due to hardware Limitations.

I successfully ran Kunit tests on the kernel source tree. First, I set up the testing environment and executed the Kunit test suite using the kunit.py tool. After resolving any necessary dependencies, I ensured that all tests were executed without issues. The tests were designed to verify various kernel functionalities, and I received detailed results for each test, confirming that all tests passed successfully. This process validated the stability and correctness of the kernel's features covered by the Kunit test suite.

```
Jan 15 02:29
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9
uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9    uniqueusman@archlinux:~/lfx_mentorship/linux-6.12.9

^C[uniqueusman@archlinux linux-6.12.9]$ sudo ./tools/testing/kunit/kunit.py run
[02:28:14] Configuring KUnit Kernel ...
[02:28:14] Building KUnit Kernel ...
Populating config with:
$ make ARCH=um O=.kunit olddefconfig
Building with:
$ make all compile_commands.json ARCH=um O=.kunit --jobs=16
[02:28:21] Starting KUnit Kernel (1/1)...
[02:28:21] ============================================================
Running tests with:
$ .kunit/linux kunit.enable=1 mem=1G console=tty kunit_shutdown=halt
[02:28:21] ================ example_init (1 subtest) ================
[02:28:21] [PASSED] example_init_test
[02:28:21] ================= [PASSED] example_init ==================
[02:28:21] ============== time_test_cases (1 subtest) ===============
[02:28:22] [PASSED] time64_to_tm_test_date_range
[02:28:22] ================ [PASSED] time_test_cases ================
[02:28:22] ================= resource (3 subtests) ==================
[02:28:22] [PASSED] resource_test_union
[02:28:22] [PASSED] resource_test_intersection
[02:28:22] [PASSED] resource_test_region_intersects
[02:28:22] ================== [PASSED] resource ====================
[02:28:22] =============== sysctl_test (11 subtests) ===============
[02:28:22] [PASSED] sysctl_test_api_dointvec_null_tbl_data
[02:28:22] [PASSED] sysctl_test_api_dointvec_table_maxlen_unset
[02:28:22] [PASSED] sysctl_test_api_dointvec_table_len_is_zero
[02:28:22] [PASSED] sysctl_test_api_dointvec_table_read_but_position_set
[02:28:22] [PASSED] sysctl_test_dointvec_read_happy_single_positive
[02:28:22] [PASSED] sysctl_test_dointvec_read_happy_single_negative
[02:28:22] [PASSED] sysctl_test_dointvec_write_happy_single_positive
[02:28:22] [PASSED] sysctl_test_dointvec_write_happy_single_negative
[02:28:22] [PASSED] sysctl_test_api_dointvec_write_single_less_int_min
[02:28:22] [PASSED] sysctl_test_api_dointvec_write_single_greater_int_max
[02:28:22] [PASSED] sysctl_test_register_sysctl_sz_invalid_extra_value
[02:28:22] ================== [PASSED] sysctl_test ==================
[02:28:22] ==================== exec (1 subtest) ====================
[02:28:22] [PASSED] exec_test_bprm_stack_limits
[02:28:22] ===================== [PASSED] exec =====================
```

The logs of Kunit test.

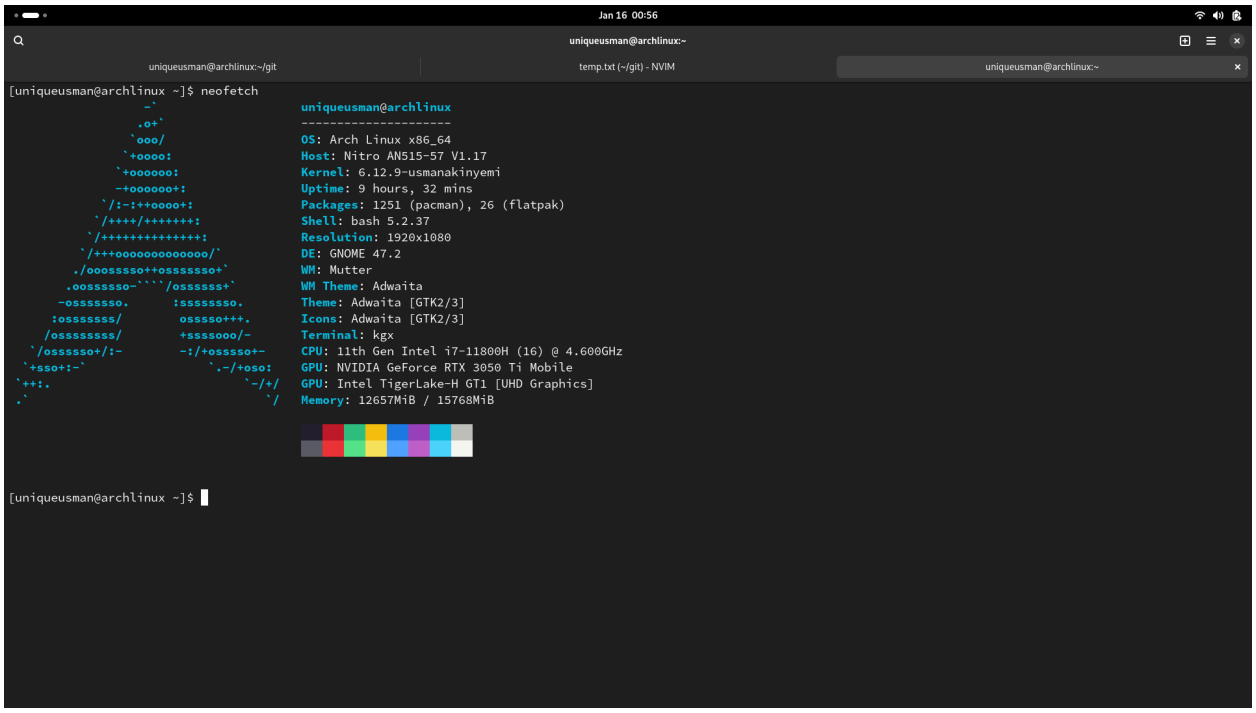
```
Jan 16 00:04
uniqueusman@archlinux:~/kerneci_coding_challenge
uniqueusman@archlinux:~/git    temp.txt (~/git) - NVIM    uniqueusman@archlinux:~/kerneci_coding_challenge

[02:28:40] [PASSED] fortify_test_memchr_inv
[02:28:40] [PASSED] fortify_test_memcmp
[02:28:40] [PASSED] fortify_test_kmemdup
[02:28:40] ==================== [PASSED] fortify ====================
[02:28:40] ================== siphash (1 subtest) ===================
[02:28:40] [PASSED] siphash_test
[02:28:40] ==================== [PASSED] siphash ====================
[02:28:40] ================= usercopy (4 subtests) ==================
[02:28:40] [PASSED] usercopy_test_valid
[02:28:40] [PASSED] usercopy_test_invalid
[02:28:40] [PASSED] usercopy_test_check_nonzero_user
[02:28:40] [PASSED] usercopy_test_copy_struct_from_user
[02:28:40] ==================== [PASSED] usercopy ===================
[02:28:40] ============== qos-kunit-test (3 subtests) ===============
[02:28:40] [PASSED] freq_qos_test_min
[02:28:40] [PASSED] freq_qos_test_maxdef
[02:28:40] [PASSED] freq_qos_test_readd
[02:28:40] ================ [PASSED] qos-kunit-test =================
[02:28:40] ============== property-entry (7 subtests) ===============
[02:28:40] [PASSED] pe_test_uints
[02:28:40] [PASSED] pe_test_uint_arrays
[02:28:40] [PASSED] pe_test_strings
[02:28:40] [PASSED] pe_test_bool
[02:28:40] [PASSED] pe_test_move_inline_u8
[02:28:40] [PASSED] pe_test_move_inline_str
[02:28:40] [PASSED] pe_test_reference
[02:28:40] ================= [PASSED] property-entry =================
[02:28:40] ================ input_core (4 subtests) =================
[02:28:40] [PASSED] input_test_polling
[02:28:40] [PASSED] input_test_timestamp
[02:28:40] [PASSED] input_test_match_device_id
[02:28:40] [PASSED] input_test_grab
[02:28:40] ================== [PASSED] input_core ===================
[02:28:40] ============================================================
[02:28:40] Testing complete. Ran 381 tests: passed: 370, skipped: 11
[02:28:40] Elapsed time: 6.736s total, 0.001s configuring, 3.049s building, 3.594s running

[uniqueusman@archlinux kerneci_coding_challenge]$
```

The two logs from the Kunit and Kselftest are also in the repository ([https://github.com/Unique-Usman/KernelCI_task_submission_logs](https://github.com/Unique-Usman/KernelCI_task_submission_logs))

Also, one way to actually test the Kernel is to use it, since I installed the Kernel, I have been using it to daily drive my pc so, it is also a good test that the Kernel was installed properly. Below is my uptime since I on my PC with the same Kernel since morning.



Also, while working on this task, I quickly took “A Beginner’s Guide to Linux Kernel Development (LFD103) ”