

Kernel Hacking

Roopam Taneja

IIT Roorkee CSE

Some gyaan I had jotted down

- **libvirt:** It's an open-source API, daemon, and management tool for managing platform virtualization. It provides a toolkit to manage virtualization platforms. It supports various virtualization technologies such as KVM, QEMU etc.
- **QEMU CPU emulator:** QEMU is a free and open-source emulator that emulates a computer's processor through dynamic binary translation and provides a set of different hardware and device models for the machine, enabling it to run a variety of guest operating systems. It can run operating systems and programs made for one machine on a different machine
- **KVM virtual machine:** Kernel-based Virtual Machine (KVM) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). KVM lets you turn Linux into a hypervisor that allows a host machine to run multiple, isolated virtual environments called guests or virtual machines (VMs).
- **virt-manager:** The virt-manager application is a desktop user interface for managing virtual machines through libvirt. It primarily targets KVM VMs, but also manages Xen and LXC (linux containers). It presents a summary view of running domains, their live performance & resource utilization statistics.
- **virsh command:** The virsh command is the command line counterpart of virt-manager.

So, in simple terms, KVM allows you to run multiple operating systems at the same time on one machine. libvirt is a tool that helps you manage these systems, and libvirtd is the part of libvirt that's always running, waiting to carry out libvirt's instructions. They all work together to make managing virtual machines easier.

Boot Process Summary

BIOS (or UEFI) finds and runs MBR (Master Boot Record : 512 bytes generally)
=> Starts bootloader (eg GRUB2) .

(GRUB2 shows a screen to select from different kernel images or different OS (in case of dual boot))

At boot time, the boot loader loads the kernel and the initramfs image into memory and starts the kernel. The kernel checks for the presence of the initramfs

and, if found, mounts it as / and runs init or systemd. Kernel also loads various kernel modules and device drivers.

The only purpose of an initramfs is to mount the root filesystem. The initramfs is a complete set of directories that you would find on a normal root filesystem.

SystemD : Process with PID = 1

Real, User and Sys process time statistics

One of these things is not like the other. Real refers to actual elapsed time; User and Sys refer to CPU time used only by the process.

Real is wall clock time - time from start to finish of the call. This is all elapsed time including time slices used by other processes and time the process spends blocked (for example if it is waiting for I/O to complete).

User is the amount of CPU time spent in user-mode code (outside the kernel) within the process. This is only actual CPU time used in executing the process. Other processes and time the process spends blocked do not count towards this figure.

Sys is the amount of CPU time spent in the kernel within the process. This means executing CPU time spent in system calls within the kernel, as opposed to library code, which is still running in user-space. Like 'user', this is only CPU time used by the process. See below for a brief description of kernel mode (also known as 'supervisor' mode) and the system call mechanism.

User+Sys will tell you how much actual CPU time your process used. Note that this is across all CPUs, so if the process has multiple threads (and this process is running on a computer with more than one processor) it could potentially exceed the wall clock time reported by Real (which usually occurs). Note that in the output these figures include the User and Sys time of all child processes (and their descendants) as well when they could have been collected, e.g. by wait(2) or waitpid(2), although the underlying system calls return the statistics for the process and its children separately.

My Experiments with Kernel Developments

Started with the aim of completing atleast Hard Assignment-1 of this course : <https://www.cse.iitd.ac.in/~srsarangi/osbook/index.html>

Started with following the kernel-hacking-guide on the above link.

The host was an x86_64 system running Ubuntu 22.04.

Kernel compiling

Installed the packages and started with cloning the kernel code from the repo => v6.1.6.

The first day perhaps the network connection was poor since the estimated time was more than 5 hours.

I cancelled it for that day and tried it some other day, could complete the clone under 40 minutes (phew).

Time for compiling the kernel: Started compiling with the default config, ran into Makefile errors

Tried modifying a couple of system certificates still the same issue.

Tired, thought of plan-B : tarball

Downloaded latest linux kernel tarball => v6.7.8, and after modifying the system certificates started compiling it.

After 35-40 mins it compiled without Makefile errors this time (phew) since I was assigning it 8 cores (make -j 8)

Yocto Project

From here things got tricky.

Followed the guide, cloned yocto project and gave the command to build the image :

bitbake -k core-image-sato (ofc after fixing a couple of errors : Errors to me : I am inevitable!)

```
computer@computer-OptiPlex-3050: ~/Documents/live_again/yocto/build$ cd yocto
computer@computer-OptiPlex-3050: ~/Documents/live_again/yocto$ source oe-init-build-env
ash shell environment set up for builds. ##
You can now run 'bitbake <target>'

Common targets are:
  core-image-minimal
  core-image-full-cmdline
  core-image-sato
  core-image-weston
  meta-toolchain
  meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemu64-64'.

Other commonly useful commands are:
  'devtool' and 'recipetool' handle common recipe tasks
  'bitbake-layers' handles common layer tasks
  'oe-pkgdata-util' handles common target package tasks
computer@computer-OptiPlex-3050: ~/Documents/live_again/yocto/build$ bitbake -k core-image-sato -c clean
Loading cache: 100% ##### Time: 0:00:00
Loaded 1800 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "2.4.0"
BUILD_SYS            = "x86_64-linux"
NATIVE_SYS           = "universal"
TARGET_SYS           = "x86_64-poky-linux"
MACHINE              = "qemu64-64"
DISTRO               = "poky"
DISTRO_VERSION        = "4.2.4"
TUNE_FEATURES         = "m64 core2"
TARGET_FPU            = ""
meta
meta-poky
meta-yocto-bsp       = "mickledore:7235399a80b134e57d5eb78d77f1f57ca0439ae5"

Initialising tasks: 100% ##### Time: 0:00:02
State summary: Mitted 0 Local 0 Mirrors 0 Missed 0 Current 0 (0% match, 0% complete)
NOTE: No setscene tasks
NOTE: Executing tasks
NOTE: Task Summary: Attempted 1 tasks of which 0 didn't need to be rerun and all succeeded.
computer@computer-OptiPlex-3050: ~/Documents/live_again/yocto/build$ bitbake -k core-image-sato
Loading cache: 100% ##### Time: 0:00:00
Loaded 1800 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "2.4.0"
BUILD_SYS            = "x86_64-linux"
NATIVE_SYS           = "universal"
TARGET_SYS           = "x86_64-poky-linux"
MACHINE              = "qemu64-64"
```

And it took a lot of time => 3:30 hours and it gets stuck at 98 %

Only major task pending is do-fetch is fetch of linux-yocto and it is stuck at a measly 26%. The transfer rates are also not encouraging. After finding no good

solutions on stack overflow, I realised this can't proceed like this (An increment of 1% took more than 15-20 mins)

```
computer@computer-OptiPle...
Loaded 1800 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION      = "2.4.0"
BUILD_SYS       = "x86_64-linux"
NATIVELSBSTRING = "universal"
TARGET_SYS      = "x86_64-poky-linux"
MACHINE         = "qemux86-64"
DISTRO          = "poky"
DISTRO_VERSION  = "4.2.4"
TUNE_FEATURES   = "m64 core2"
TARGET_FPU      = ""
meta
meta-poky
meta-yocto-bsp  = "mickledore:7235399a86b134e57d5eb783d7f1f57ca0439ae5"

Initialising tasks: 100% |#####|
#####| Time: 0:00:04
Sstate summary: Wanted 514 Local 464 Mirrors 0 Missed 50
Current 2669 (90% match, 98% complete)
NOTE: Executing Tasks
Setscene tasks: 3183 of 3183
Setscene tasks: 3183 of 3183
Currently 1 running tasks (7854 of 8036) 97% |#####|
#####|
0: linux-yocto-6.1.57+gitAUTOINC+8aa5efbc5e_8a449d3428-r
0 do_fetch - 15m5s (pid 646022) 26% || 75.00 KiB/s
```

```
computer@computer-OptiPlex-3050: ~/Documents/live_again/poky/build
- 'bitbake-layers' handles common layer tasks
- 'oe-pkgdata-util' handles common target package tasks
computer@computer-OptiPlex-3050: ~/Documents/live_again/poky/build$ bitbake -k core-image-sato -c clean
Loading cache: 100% ##### Time: 0:00:00
Loaded 3100 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build configuration:
BB_VERSION           = "2.4.0"
BUILD_SYS            = "x86_64-linux"
HOST_LVSSTRING       = "universal"
TARGET_SYS           = "x86_64-poky-linux"
MACHINE              = "qemu68-64"
DISTRO               = "poky"
DISTRO_VERSION        = "4.2.4"
TUNE_FEATURES        = "m4d core2"
TARGET_FPU           = ""
meta
meta-poky
meta-yocto-bsp       = "mickledore:7235399a0b134e57d5eb783d7f1f57ca0439ae5"
Initialising tasks: 100% ##### Time: 0:00:02
State summary: Wanted 0 Local 0 Mirrors 0 Missed 0 Current 0 (0% match, 0% complete)
NOTE: Executing tasks
NOTE: Tasks Summary: Attempted 1 tasks of which 0 didn't need to be rerun and all succeeded.
computer@computer-OptiPlex-3050: ~/Documents/live_again/poky/build$ export LC_ALL="en_US.UTF-8"
computer@computer-OptiPlex-3050: ~/Documents/live_again/poky/build$ bitbake -k core-image-sato
Loading cache: 100% ##### Time: 0:00:00
Loaded 3100 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build configuration:
BB_VERSION           = "2.4.0"
BUILD_SYS            = "x86_64-linux"
HOST_LVSSTRING       = "universal"
TARGET_SYS           = "x86_64-poky-linux"
MACHINE              = "qemu68-64"
DISTRO               = "poky"
DISTRO_VERSION        = "4.2.4"
TUNE_FEATURES        = "m4d core2"
TARGET_FPU           = ""
meta
meta-poky
meta-yocto-bsp       = "mickledore:7235399a0b134e57d5eb783d7f1f57ca0439ae5"
Initialising tasks: 100% ##### Time: 0:00:04
State summary: Wanted 318 Local 404 Mirrors 0 Missed 50 Current 2609 (99% match, 99% complete)
Getscene task: 3183 of 3183
Currently 1 running task (7854 of 4036) 97% #####
0: linux-yocto-6.1.57gitTAUTOINC-Baa5e7b5e-Ba449d3420-r0 do_fetch - 15m15s (pid 646022) 26% ##### | 84.00 KiB/s
```

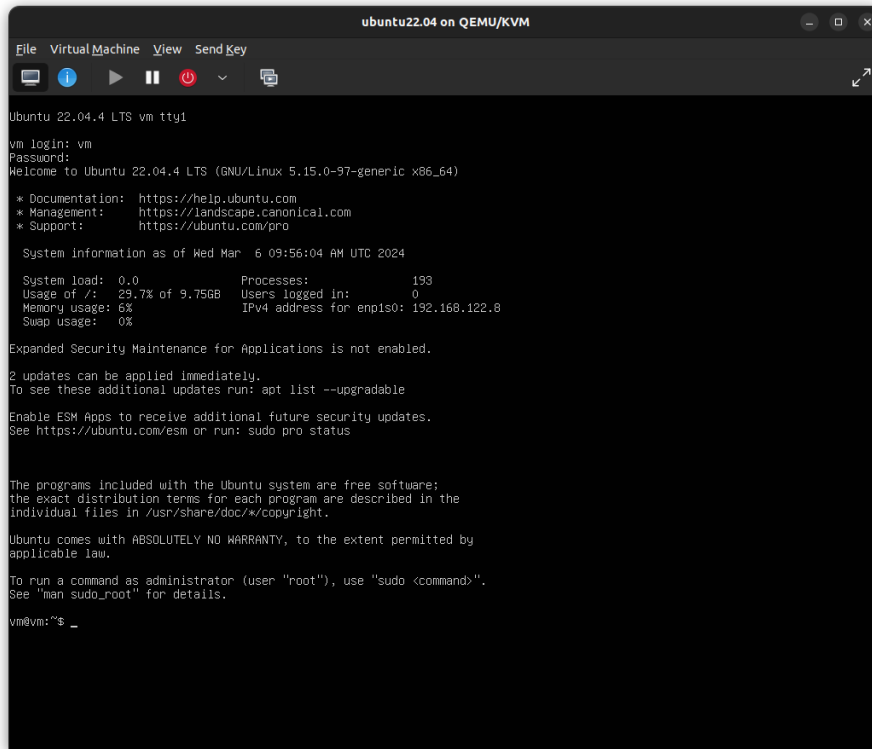
Feeling defeated? It's just getting started.

virt-manager

I thought of maybe trying the steps in the assignment themselves.

I saw the initial steps were already done, it was just about booting a new VM so I followed it but couldn't really get the configuring storage part.

Now I had a VM using virt-manager booting up Ubuntu server ISO downloaded from the net and now I had to boot it from my custom kernel.



```
ubuntu22.04 on QEMU/KVM
File VirtualMachine View Send Key
vm login: vm
Password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System Information as of Wed Mar 6 09:56:04 AM UTC 2024

System load: 0.0          Processes: 193
Usage of /: 29.7% of 9.75GB Users logged in: 0
Memory usage: 6%          IPv4 address for enp1s0: 192.168.122.8
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

vm@vm:~$
```

I had the kernel image so I felt all I had to do was :

```
sudo make modules_install -j sudo make install -j
```

Right...right??? Just that I thought I had to do it in my host terminal and not the VM terminal. (Now when I look back, it feels really foolish of me).

Ok now what needs to be done? No idea.

Looked back at the guide I began with. I thought of following the rest other than the Yocto part.

So I went ahead and set the kernel path to my kernel image.

I don't have an initrd (sad) regardless let's boot it.

Anddd. . .

```
uname -r gives out 6.7.8 (Yay!)
```

But it gave a whole bunch of modules not found errors in the terminal screen.

Now I still didn't have a fs, so I further tried following the guide for mounting rootfs steps and got a whole bunch of errors and failures.

Neither the partition was happening nor the make header_install was working. The storage of the system also started running out and was also the cause of a few errors ig.

So wait I wasn't getting a great feeling about the whole ordeal.

So I decided to power down the system and turn it on again.

The Scare

Now the system I was working on was a dual boot with Ubuntu 6.5. and Windows on separate partitions. It booted with a GRUB screen with options : Ubuntu, Windows Boot Manager, Advanced Ubuntu Options , etc in that order

And on pressing Ubuntu it tried booting the same kernel with module failures (yes 6.7.8)

You can imagine the stress levels right!

I had read somewhere else as well that make install replaced their default kernel but and they lamented that they didn't have a GRUB screen with them...wait a minute, they didn't have a GRUB screen but I do

I rebooted the system, in the GRUB screen selected Advanced Ubuntu Options and it showed me three different kernel versions including the older one which was running and the one I compiled.

And the old kernel booted successfully (imagine the relief I got, no leave it, you won't be able to imagine)

Now since it was booting the new one by default I modified the GRUB menu to rather choose the old one for default (stack overflow orz)

In short, there was no problem really

Scare averted - Back to work

With the scare I got, I simply stopped following these two guides I was following till now.

Removed the VM, the associated disk image, the ISO file everything.

Also removed the partial bitbake build since I had lost all hope on it.

Now it was just me and my kernel image.

QEMU

After almost losing hope, I searched for booting a custom kernel with maybe just qemu.

And I got a couple of great things to proceed :

<https://mgalgs.io/2015/05/16/how-to-build-a-custom-linux-kernel-for-qemu-2015-edition.html> (I must say this, if a single blog helped me the most, it was this one)

<https://blog.vmsplice.net/2011/02/near-instant-kernel-development-cycle.html>

They introduced me to busybox (meant for creating a custom userspace and giving a primitive set of commands). Initially building busybox was also giving Makefile errors but I could sort them by just using a newer version of busybox.

So basically here I was setting up my custom initramfs and tinkering with flags of qemu (which scared me a couple of weeks earlier). Honestly at this point I was less concerned about the end result but enjoying the process.

Created my init as per the first blog, but it gave me errors with connecting console tty0 so I took help from other blogs and stack overflow to fix it finally.

My workflow looked like this at that time:

- 1) Modify init
- 2) Compress initramfs : `find . -print0`
 | `cpio -null -ov ...`
- 3) cd to parent directory
- 4) Run qemu:

```
qemu-system-x86_64      -kernel linux-6.7.8/arch/x86_64/boot/bzImage \
-initrd obj/initramfs-busybox-x86.cpio.gz \
-nographic -append "console=ttyS0" -enable-kvm \
-drive file=disk_data.img,if=virtio,cache=none
```

(the last line came after I set up the qemu disk-img , more on that later)

Some images:

First time qemu custom kernel loaded, presented in full glory :

9

```
computer@computer-OptiPlex3050:~/Documents/live_again$
0.977836] PCI: Using host bridge windows from ACPI; if necessary, use "pci=
0.978099] PCI: Using 820 reservations for host bridge windows
0.980019] ACPI: Enabled 2 GPEs in block 00 to GPE
1.010960] ACPI: PCI Root Bridge (PCI0) (domain 0000 [bus 00-ff])
1.017951] acpi PMPM03:00: -GSC: OS supports (SMM Clocks Segments M1 ED)
1.018365] acpi PMPM03:00: -GSC: not requesting OS control, OS requires [E]
1.018355] acpi PMPM03:00: fail: to add MCMR2C information, can't access e
1.020030] acpiPhp: Slot [3] registered
1.020330] acpiPhp: Slot [4] registered
1.020600] acpiPhp: Slot [5] registered
1.020810] acpiPhp: Slot [6] registered
1.020903] acpiPhp: Slot [7] registered
1.020283] acpiPhp: Slot [8] registered
1.020636] acpiPhp: Slot [9] registered
1.020841] acpiPhp: Slot [10] registered
1.030646] acpiPhp: Slot [11] registered
1.030255] acpiPhp: Slot [12] registered
1.030445] acpiPhp: Slot [13] registered
1.030637] acpiPhp: Slot [14] registered
1.030820] acpiPhp: Slot [15] registered
1.031039] acpiPhp: Slot [16] registered
1.031228] acpiPhp: Slot [17] registered
1.031460] acpiPhp: Slot [18] registered
1.031651] acpiPhp: Slot [19] registered
1.031897] acpiPhp: Slot [20] registered
1.032124] acpiPhp: Slot [21] registered
1.032362] acpiPhp: Slot [22] registered
1.032607] acpiPhp: Slot [23] registered
1.032837] acpiPhp: Slot [24] registered
1.033051] acpiPhp: Slot [25] registered
1.033241] acpiPhp: Slot [26] registered
1.033430] acpiPhp: Slot [27] registered
1.033660] acpiPhp: Slot [28] registered
1.033866] acpiPhp: Slot [29] registered
1.034046] acpiPhp: Slot [30] registered
1.034234] acpiPhp: Slot [31] registered
1.034461] PCI host bridge to bus 0000:00
1.035047] pci_bus 0000:00: root bus resource [to 0x0000-0x0fff window]
1.035058] pci_bus 0000:00: root bus resource [to 0x0000-0xffff window]
1.035745] pci_bus 0000:00: root bus resource [mem 0x000a0000-0x000bffff w1]
1.036072] pci_bus 0000:00: root bus resource [mem 0x000c0000-0xfedfffff w1]
1.036328] pci_bus 0000:00: root bus resource [mem 0x10000000-0x17ffffff w1]
1.036639] pci_bus 0000:00: root bus resource [bus 00-ff]
1.038260] pci 0000:00:00.0: [0000:1237] type 00 class 0x060000
1.040332] pci 0000:00:01.0: [0000:7000] type 00 class 0x060100
1.040360] pci 0000:00:01.1: [0000:7000] type 00 class 0x060100
1.051475] pci 0000:00:01.1: reg 0x28: [io 0xc000-0xc0ff]
1.052438] pci 0000:00:01.1: legacy IDE quirk: reg 0x10: [io 0x01f0-0x01ff]
1.052664] pci 0000:00:01.1: legacy IDE quirk: reg 0x14: [io 0x0170]
1.052860] pci 0000:00:01.1: legacy IDE quirk: reg 0x18: [io 0x01f0-0x01ff]
1.053055] pci 0000:00:01.1: legacy IDE quirk: reg 0x1c: [io 0x0170]
1.053099] pci 0000:00:01.1: [0000:7113] type 00 class 0x060000
1.054444] pci 0000:00:01.3: quirk: [to 0x0000-0x003f] claimed by PIIX4 ACPI
1.054607] pci 0000:00:01.3: quirk: [to 0x0700-0x070f] claimed by PIIX4 SMB
1.055244] pci 0000:00:02.0: [1234:1111] type 00 class 0x030000
1.055197] pci 0000:00:02.0: rom 0x10: [mem 0xfed00000-0xfedfffff w64]

computer@computer-OptiPlex3050:~/Documents/live_again$
1.582786] platform eisa.0: Cannot allocate resource for EISA slot 4
1.583101] platform eisa.0: Cannot allocate resource for EISA slot 5
1.583270] platform eisa.0: Cannot allocate resource for EISA slot 6
1.583431] platform eisa.0: Cannot allocate resource for EISA slot 7
1.583593] platform eisa.0: Cannot allocate resource for EISA slot 8
1.583774] platform eisa.0: EISA: detected 0 cards
1.584053] amd-pstate: the CPU object is not present in SMIO5 or ACPI dsiod
1.584530] ledtrig-cpu: registered to indicate activity on CPUs
1.587660] irq: shorthand broadcast: enabled
1.522224] sched_clock: Marking stable (1084048270, 34058999)--(158159208,)
1.532101] Loading compiled-in module x.509 certificates
1.547191] Loaded x.509 cert 'Build time autogenerated kernel key: f814a0fd'
1.563683] Key type fscrypt registered
1.563844] Key type fscrypt-provisioning registered
1.569510] ata2: Found unknown device (class 0)
1.651189] ata2.00: ATAPI: QEMU DVD-ROM, 2.15+ max UDMA/100
1.674022] Key type encrypted registered
1.674028] Une: No TPM chip found, activating TPM-bypass!
1.676223] scsi 1:0:0:0: CD-ROM QEMU QEMU DVD-ROM 2.1+ PS
1.677629] Loading compiled-in module x.509 certificates
1.681070] Loaded x.509 cert 'Build time autogenerated kernel key: f814a0fd'
1.681087] Une: Allocated hash algorithm: sha1
1.687029] Une: No architecture policies found
1.688173] evm: Initialising EVM extended attributes:
1.688455] evm: security.selinux (disabled)
1.688637] evm: security.SMACK64 (disabled)
1.688784] evm: security.SMACK64EXEC (disabled)
1.688963] evm: security.SMACK64TRANSMUTE (disabled)
1.689130] evm: security.SMACK64MEMP (disabled)
1.689313] evm: security.apparmor (disabled)
1.689409] evm: security.ima
1.689573] evm: security.capability
1.689712] evm: HMAC attrs: 8x1
1.693239] Une: Magic number: 0:036:74
1.694647] poweroff.k8: Power state transitions not supported
1.720652] sr 1:0:0:0: (sr0) scsi3mm: drive: 4x/4x cd/rw xaif2rn2 tray
1.721383] cdrom: Uniform CD-ROM driver Revision: 3.20
1.722093] h2c_bsf: error while preloading H2B BSM dispatcher: -22
1.723233] NAG: Correctable Errors collector initialized.
1.729223] CLK: disabling unused clocks
1.742761] sr 1:0:0:0: Attached scsi generic sg0 type 5
1.752121] Freeing unused decrypted memory: 2028K
1.760454] Freeing unused kernel: kpage (kfinem) memory: 4272K
1.797310] Write protecting the kernel read-only data: 23528K
1.798073] Freeing unused kernel: kpage (redat/data gap) memory: 4K
1.971407] x86/mtrr: Checked mtrr mappings: passed, no MTRR pages found.
1.972007] Run /init as init process

root took 1.98 seconds

/bin/sh: can't access tty; job control turned off
# 2.193555] tsc: refined TSC clocksource calibration: 3599.738 MHz
2.196125] clocksource: tsc: mask: 0xfffffffffff max_cycles: 0x3e35bds
2.196471] clocksource: Switched to clocksource tsc
```

```
computer@computer-OptiPlex3050: ~/Documents/live_again
1.593270 platform eisa.0: Cannot allocate resource for EISA slot 6
1.593281 platform eisa.0: Cannot allocate resource for EISA slot 7
1.593593 platform eisa.0: Cannot allocate resource for EISA slot 8
1.593774 platform eisa.0: EISA: Detected 0 cards
1.594033 end state: the CPU object is not present in SMIOS or ACPI disad
1.594536 ledtrig-cpu: registered to indicate activity on CPUs
1.597860 IPI short-hand broadcast enabled
1.522224 sched_clock: Marking stable (148048270, 34058999)--(158119208,)
1.531101 Loading compiled-in X.509 certificates
1.547191 Loaded X.509 cert 'build time autogenerated kernel key: f814a5fd'
1.563603 Key type .fscrypt registered
1.563444 Key type fscrypt-preallocation registered
1.639510 ata2: Found unknown device (class 0)
1.651189 ata2.00: ATA: QEMU DVD-ROM, 1.5+ max UDMA/100
1.674022 Key type encrypted registered
1.674091 In: no TPM chip found, activating TPM-bypass!
1.676123 scsi 1:0:0:0: CD-ROM QEMU QEMU DVD-ROM 2.5+ PS
1.677769 Loading compiled-in module X.509 certificates
1.681070 Loaded X.509 cert 'build time autogenerated kernel key: f814a5fd'
1.681837 Ina: Allocated hash algorithm: sha1
1.682023 Ina: No architecture policies found
1.688173 evm: Initialising EVM extended attributes:
1.688435 evm: security.selinux (disabled)
1.688637 evm: security.SMACK64 (disabled)
1.688784 evm: security.SMACK64EX (disabled)
1.688963 evm: security.SMACK64TRANSMUTE (disabled)
1.689180 evm: security.SMACK64MAP (disabled)
1.689313 evm: security.aparmor (disabled)
1.689450 evm: security.tsm
1.689573 evm: security.capability
1.689712 evm: HMAC attr: 0x1
1.693230 PM: Magic number: 0:0:0:74
1.694647 poweroff.k8: Power state transitions not supported
1.720622 sr 1:0:0:0: (sr) scsi-mmc drive: 4x/4x cd/rw xa/forrn tray
1.721108 cdrom: Uniform CD-ROM driver Revision: 3.20
1.722009 hld-bf: error while preloading HLD BPF dispatcher: -22
1.732323 RAS: Correctable Errors collector initialized
1.732323 clk: Disabling unused clocks
1.742761 sr 1:0:0:0: Attached scsi generic sg0 type 5
1.752121 Freeing unused decrypted memory: 2030K
1.756141 Freeing unused kernel image (initmem) memory: 4272K
1.797310 Write protecting the kernel read-only data: 2232K
1.798771 Freeing unused kernel image (rodata/data gap) memory: 4K
1.971407 x86/mi: Checked mxi mappings: passed, no mxi pages found.
1.972007 Run /init as init process

Boot took 1.98 seconds

root/sh: can't access tty: job control turned off
# 1 3.95552 tsc: started TSC clocksource calibration: 3599.730 MHz
# 2 1.96125 clocksource: tsc: mask: 0xfffffffffff max_cycles: 0x3e35bds
# 3 2.196471 clocksource: Switched to clocksource tsc

uname
Linux
#
```

After fixing tty0 errors with modified init:

```
computer@computer-OptiPlex3050: ~/Documents/live_again
0.553957 platform eisa.0: Cannot allocate resource for EISA slot 4
0.554978 platform eisa.0: Cannot allocate resource for EISA slot 5
0.555973 platform eisa.0: Cannot allocate resource for EISA slot 6
0.557015 platform eisa.0: Cannot allocate resource for EISA slot 7
0.558016 platform eisa.0: Cannot allocate resource for EISA slot 8
0.559039 platform eisa.0: EISA: Detected 0 cards
0.559776 IPI short-hand broadcast enabled
0.560491 ledtrig-cpu: registered to indicate activity on CPUs
0.561554 IPI short-hand broadcast enabled
0.563003 sched_clock: Marking stable (640404238, 98558371)--(567135404, -)
0.564841 Loading compiled-in X.509 certificates
0.566491 Loaded X.509 cert 'build time autogenerated kernel key: f814a5fd'
0.566990 Key type .fscrypt registered
0.566644 Key type fscrypt-preallocation registered
0.572510 Key type encrypted registered
0.574140 In: no TPM chip found, activating TPM-bypass!
0.575022 Loading compiled-in module X.509 certificates
0.576220 Loaded X.509 cert 'build time autogenerated kernel key: f814a5fd'
0.577707 Ina: Allocated hash algorithm: sha1
0.578480 Ina: No architecture policies found
0.578123 evm: Initialising EVM extended attributes:
0.579097 evm: security.selinux (disabled)
0.580581 evm: security.SMACK64 (disabled)
0.581287 evm: security.SMACK64EX (disabled)
0.581899 evm: security.SMACK64TRANSMUTE (disabled)
0.582685 evm: security.SMACK64MAP (disabled)
0.583430 evm: security.aparmor (disabled)
0.584164 evm: security.tsm
0.584560 evm: security.capability
0.585168 evm: HMAC attr: 0x1
0.585833 PM: Magic number: 0:0:0:677
0.587237 hld-bf: error while preloading HLD BPF dispatcher: -22
0.587251 RAS: Correctable Errors collector initialized
0.589007 clk: Disabling unused clocks
0.601338 ata2: Found unknown device (class 0)
0.609331 ata2.00: ATA: QEMU DVD-ROM, 2.5+ max UDMA/100
0.609525 scsi 1:0:0:0: CD-ROM QEMU QEMU DVD-ROM 2.5+ PS
0.727564 sr 1:0:0:0: (sr) scsi-mmc drive: 4x/4x cd/rw xa/forrn tray
0.728141 cdrom: Uniform CD-ROM driver Revision: 3.20
0.743792 sr 1:0:0:0: Attached scsi generic sg0 type 5
0.747186 Freeing unused decrypted memory: 2030K
0.749935 Freeing unused kernel image (initmem) memory: 4272K
0.756101 Write protecting the kernel read-only data: 2232K
0.759960 Freeing unused kernel image (rodata/data gap) memory: 4K
0.799880 x86/mi: Checked mxi mappings: passed, no mxi pages found.
0.799739 x86/mi: Checking user space page tables
0.832810 x86/mi: Checked mxi mappings: passed, no mxi pages found.
0.836131 Run /init as init process

Mknod: /dev/ttyS0: File exists

Boot took 0.75 seconds

hello Hoopani!
#
```

A treat for eyes :

```
computer@computer-OptiPlex-3050: ~/Documents/live_again
- # uname -r
0: 2.6
- #

0.558816] platform etsa.0: Cannot allocate resource for EISA slot 8
0.558839] platform etsa.0: EISA: detected 0 cards
0.559776] intel_gstate: CPU model not supported
0.560491] ledtrig-cpu: registered to indicate activity on CPUs
0.561554] IPI shorthand broadcast: enabled
0.563603] sched_clock: Marking stable (4c4804239, 98558371)->(567135404, -)
0.564851] Loading compiled-in x.509 certificates
0.566049] Loaded x.509 cert 'Build time autogenerated kernel key: f81a4afd'
0.566901] Key type fscrypt registered
0.569061] Key type fscrypt-provisioning registered
0.572510] Key type encrypted registered
0.574140] tpm: No TPM chip found, activating TPM-bypass!
0.575822] Loading compiled-in module x.509 certificates
0.576220] Loaded x.509 cert 'Build time autogenerated kernel key: f81a4afd'
0.577707] tpm: Allocated hash algorithm: sha1
0.578400] tpm: No architecture policies found
0.579123] evm: Initialising EVM extended attributes:
0.579207] evm: security.selinux (disabled)
0.580558] evm: security.SMACK64 (disabled)
0.582307] evm: security.SMACK64EX (disabled)
0.583993] evm: security.SMACK64TRANSPARE (disabled)
0.584083] evm: security.SMACKERR (disabled)
0.584393] evm: security.aparmor (disabled)
0.584104] evm: security.tsm
0.584560] evm: security.capability
0.585103] evm: HMAC attr: 0x1
0.585833] tpm: Magic number: 0:428:677
0.587287] hid_bpf: error while preloading HID BPF dispatcher: -22
0.587293] hid: CoreSight Error's collector initialized.
0.589067] clk: Disabling unused clocks
0.591331] ata2: Found unknown device (class 0)
0.591931] ata2.00: ATAPI: QEMU DVD-ROM, 2.5+, max UDMA/100
0.597625] scsi 1:0:0:0: CD-ROM QEMU QEMU DVD-ROM 2.5+ PS
0.722564] sr 1:0:0:0: (sr) scsi3-mmc drive: 4x/4x cd/rw xa/form2 tray
0.723144] cdrom: Uniform CD-ROM driver Revision: 3.20
0.743792] sr 1:0:0:0: Attached scsi generic sg type 5
0.747186] Freeing unused kernel image (initmem) memory: 4372K
0.746935] Write protecting the kernel read-only data: 22528k
0.750903] Freeing unused kernel image (reada/data gap) memory: 4K
0.750986] x86/mm: Checked w/x mappings: passed, no w/x pages found.
0.759739] x86/mm: Checking user space page tables
0.833810] x86/mm: Checked w/x mappings: passed, no w/x pages found.
0.836611] Run /init as init process
initrd: /dev/loop0 file exists

Boot took 0.75 seconds

Hello Hoopani!

- # echo "Bye"
Bye
- # qemu 4.2.0 monitor - type 'help' for more information
(qemu) quit
computer@computer-OptiPlex-3050: ~/Documents/live_again$
```

Great, so can I move on to my context switch tracker, not yet. I can't create or store files, even if I add a syscall how will I create a C file to test it.

Now I need to set up a fs, there were various suggestions ranging from yocto project (on which i had given up) to buildroot etc

But then I found this:

<https://github.com/gurugio/linuxdeveloptip/blob/master/minikernelwithbusybox.md>

Tried using it, implementing the commands on my qemu terminal this time (didn't repeat earlier mistake)

fdisk : Utility for handling disk partitions. Referred to a couple of tutorials to break my disk image into two partitions

mkfs : For formatting the disk and setting up the requested file system on the partition

mount : For mounting the fs of the disk on a specified mount point

How it looked after that :

```
computer@computer-OptiPlex-3050: ~/Documents/live_again
- # fdisk /dev/vda
The number of cylinders for this disk is set to 33280.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
  1) software that runs at boot time (e.g., old versions of LILO)
  2) booting and partitioning software from other OSs
    (e.g., DOS fdisk, OS/2 fdisk)

Command (n for help): p
Disk /dev/vda: 16 GB, 17179869184 bytes, 33554432 sectors
33280 cylinders, 16 heads, 63 sectors/track
Units: sectors of 1 * 512 = 512 bytes

Device Boot StartCHS      EndCHS      StartLBA    EndLBA    Sectors  Size Id Type
/dev/vda1 *  0,1,1        1023,15,63        63    31457342    31457280  15.5M 83 Linux
/dev/vda2                1023,15,63    31457343    33554431    2097089 1023M 82 Linux swap

Command (n for help): q

- # mkfs.ext2 /dev/vda/
mkfs.ext2 can't open '/dev/vda/': Not a directory
- # mkfs.ext2 /dev/vda
mkfs.ext2: Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
1048576 (nodes), 4194304 blocks
209715 blocks (5%) reserved for the super user
First data block=0
Maximum filesystem blocks=4194304
128 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1685532, 2654208, 4096000
[ 395.998469] random: crng init done
- #
```

Great now it should work , right...right??

It didn't.

Issue was there were no permanent writes. If I turned off the system and booted the qemu VM again, the fdisk partitions would still be there but everything else would be gone, even the empty mount point directory I would create.

Tried a couple of things from what I could lookup online, created a /etc/fstab file and added this :

```
echo '/dev/vda /mnt ext4 defaults 0 2' >> /etc/fstab
```

and **mount -a** so that it would mount it automatically, but the fun part is no trace of this fstab file would exist after rebooting the system.

Finally after having invested considerable amount of time, I decided to give up

Reflections

Saw a couple of more things which suggested setting up buildroot etc but I thought it was a better idea to get done with it.

A couple of things which I think were the roadblock maybe were the limited configuration of the busybox setup I was using. It lacked a lot of things, there was no vim, not even gcc (even if I create a C file how will I compile it.). I was

editing my files by echo (sad face). I couldn't see a method of adding packages (no apt-get as well).

Maybe I was doing some blunder like earlier

Maybe I should have continued with configuring the yocto project.

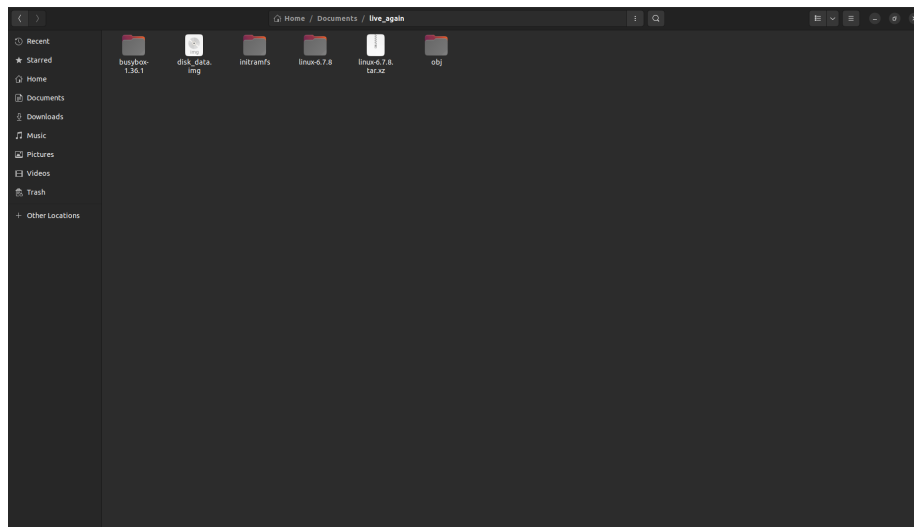
Maybe I was too hasty with my Ubuntu ISO VM (ig still compiling inside the VM could be an issue, plus configuring the fs still was a problem)

Maybe I could have continued with implementing my syscalls with my qemu setup itself. I won't be able to test them perhaps but ig successful boot could have been taken as sign of success.

Will I try to pick this particular activity again in future? Maybe not, ig the novelty of the challenge is gone and I may pickup newer projects. Or maybe I can still come back to this and start trying this again.

There are a lot of ifs and buts involved. Regardless it was a very fun exercise (tiring at some points as well) but something that I will cherish for a long time.

Directory Structure (in the end)



What I had wanted to do

Now I wanted to atleast complete the context switch tracker (bcoz the kernel module felt way tougher).

Studied a bit about how syscalls are added.

Compiled these references:

<https://www.kernel.org/doc/html/next/process/adding-syscalls.html>

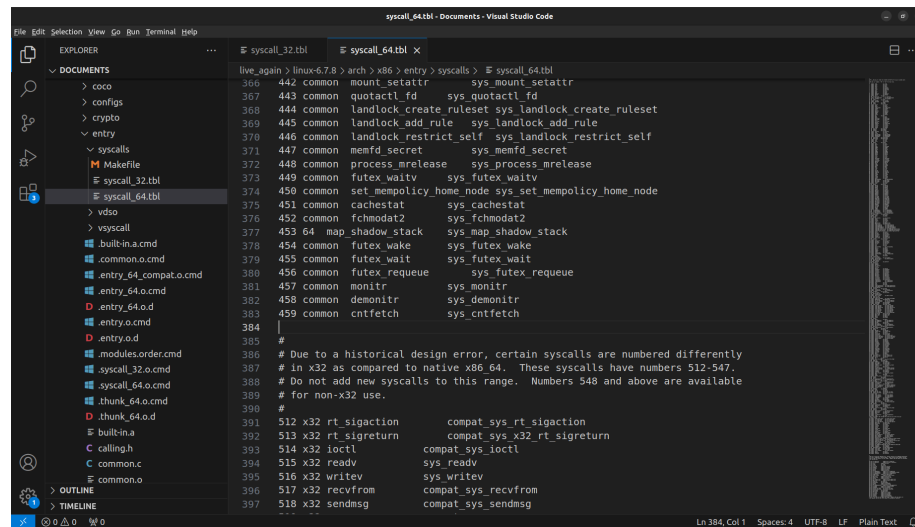
<https://brennan.io/2016/11/14/kernel-dev-ep3/>

<https://yulequan.github.io/Add-systemcall/addextraspace.html>

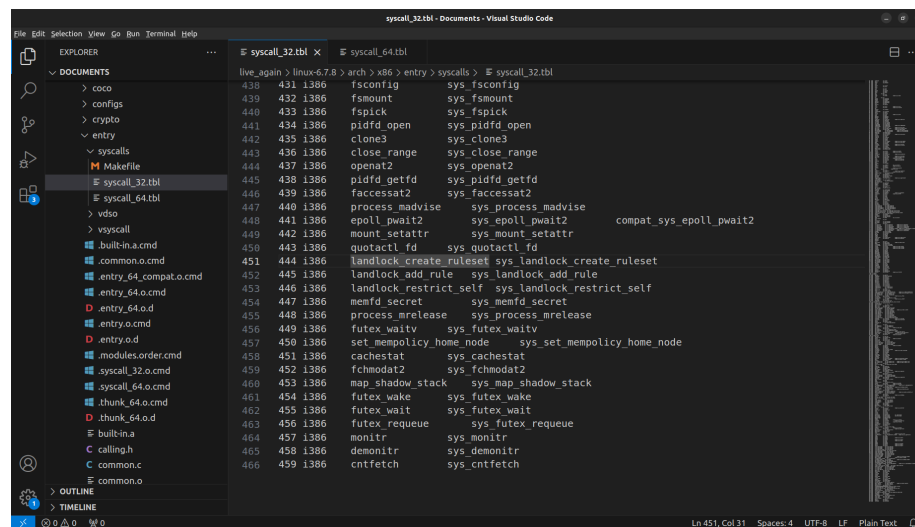
<https://www.stolaf.edu/people/rab/os/newsyscall.html>

I even studied the submitted report of a student who successfully did it (saved in folder).

I even had started modifying kernel code a bit (sadly this never saw the light of day):



```
live_again > linux-6.7.8 > arch > x86 > entry > syscalls > E syscall_64.tbl
366 442 common mount_setattr sys_mount_setattr
367 443 common quotactl_fd sys_quotactl_fd
368 444 common landlock_create_ruleset sys_landlock_create_ruleset
369 445 common landlock_add_rule sys_landlock_add_rule
370 446 common landlock_restrict_self sys_landlock_restrict_self
371 447 common memfd_secret sys_memfd_secret
372 448 common process_mrelease sys_process_mrelease
373 449 common futex_waitv sys_futex_waitv
374 450 common set_mempolicy_home_node sys_set_mempolicy_home_node
375 451 common cachestat sys_cachestat
376 452 common fchmodat2 sys_fchmodat2
377 453 64 map_shadow_stack sys_map_shadow_stack
378 454 common futex_wake sys_futex_wake
379 455 common futex_wait sys_futex_wait
380 456 common futex_requeue sys_futex_requeue
381 457 common monitor sys_monitor
382 458 common demonitr sys_demonitr
383 459 common cntfetch sys_cntfetch
384 |
385 #
386 # Due to a historical design error, certain syscalls are numbered differently
387 # in x32 as compared to native x86_64. These syscalls have numbers 512-547.
388 # Do not add new syscalls to this range. Numbers 548 and above are available
389 # for non-x32 use.
390 #
391 512 x32 rt_sigaction compat_sys_rt_sigaction
392 513 x32 rt_sigreturn compat_sys_x32_rt_sigreturn
393 514 x32 ioctl compat_sys_ioctl
394 515 x32 readv sys_readv
395 516 x32 writev sys_writev
396 517 x32 recvfrom compat_sys_recvfrom
397 518 x32 sendmsg compat_sys_sendmsg
```



```
live_again > linux-6.7.8 > arch > x86 > entry > syscalls > E syscall_32.tbl
438 431 1386 fsconfig sys_fsconfig
439 432 1386 fsmount sys_fsmount
440 433 1386 fspick sys_fspick
441 434 1386 pidfd_open sys_pidfd_open
442 435 1386 clone3 sys_clone3
443 436 1386 close_range sys_close_range
444 437 1386 openat2 sys_openat2
445 438 1386 pidfd_getfd sys_pidfd_getfd
446 439 1386 faccessat2 sys_faccessat2
447 440 1386 process_madvise sys_process_madvise
448 441 1386 epoll_pwait2 compat_sys_epoll_pwait2
449 442 1386 mount_setattr sys_mount_setattr
450 443 1386 quotactl_fd sys_quotactl_fd
451 444 1386 landlock_create_ruleset sys_landlock_create_ruleset
452 445 1386 landlock_add_rule sys_landlock_add_rule
453 446 1386 landlock_restrict_self sys_landlock_restrict_self
454 447 1386 memfd_secret sys_memfd_secret
455 448 1386 process_mrelease sys_process_mrelease
456 449 1386 futex_waitv sys_futex_waitv
457 450 1386 set_mempolicy_home_node sys_set_mempolicy_home_node
458 451 1386 cachestat sys_cachestat
459 452 1386 fchmodat2 sys_fchmodat2
460 453 1386 map_shadow_stack sys_map_shadow_stack
461 454 1386 futex_wake sys_futex_wake
462 455 1386 futex_wait sys_futex_wait
463 456 1386 futex_requeue sys_futex_requeue
464 457 1386 monitor sys_monitor
465 458 1386 demonitr sys_demonitr
466 459 1386 cntfetch sys_cntfetch
```

Ideas for custom syscalls (found online) :

- 1) Memory Statistics: Implement a syscall that returns memory usage statistics for a given process, such as the amount of virtual memory, resident

set size, etc.

- 2) File System Information: Create a syscall that returns detailed information about a specific file system, such as the type of file system, the total size, the used space, and the free space.
- 3) CPU Usage: Implement a syscall that returns the CPU usage of a given process. This could be calculated as the total CPU time used by the process divided by the total time the process has been running.

Existing syscalls for those purposes :

Memory Statistics: The `getrusage` system call can be used to get resource usage statistics for a process, which includes memory usage. Also, the `sbrk` system call can be used to increase the program's data space.

File System Information: The `stat`, `fstat`, and `lstat` system calls can be used to get information about a file, which includes the type of file system.

CPU Usage: There isn't a direct system call to get the CPU usage of a process. However, this information can be obtained by reading the `/proc/[pid]/stat` file in the `proc` filesystem

Some resources for kernel modules (maybe for me of an alternate universe) :

<https://devarea.com/linux-kernel-development-and-writing-a-simple-kernel-module/>

<https://devarea.com/linux-kernel-development-creating-a-proc-file-and-interfacing-with-user-space/>