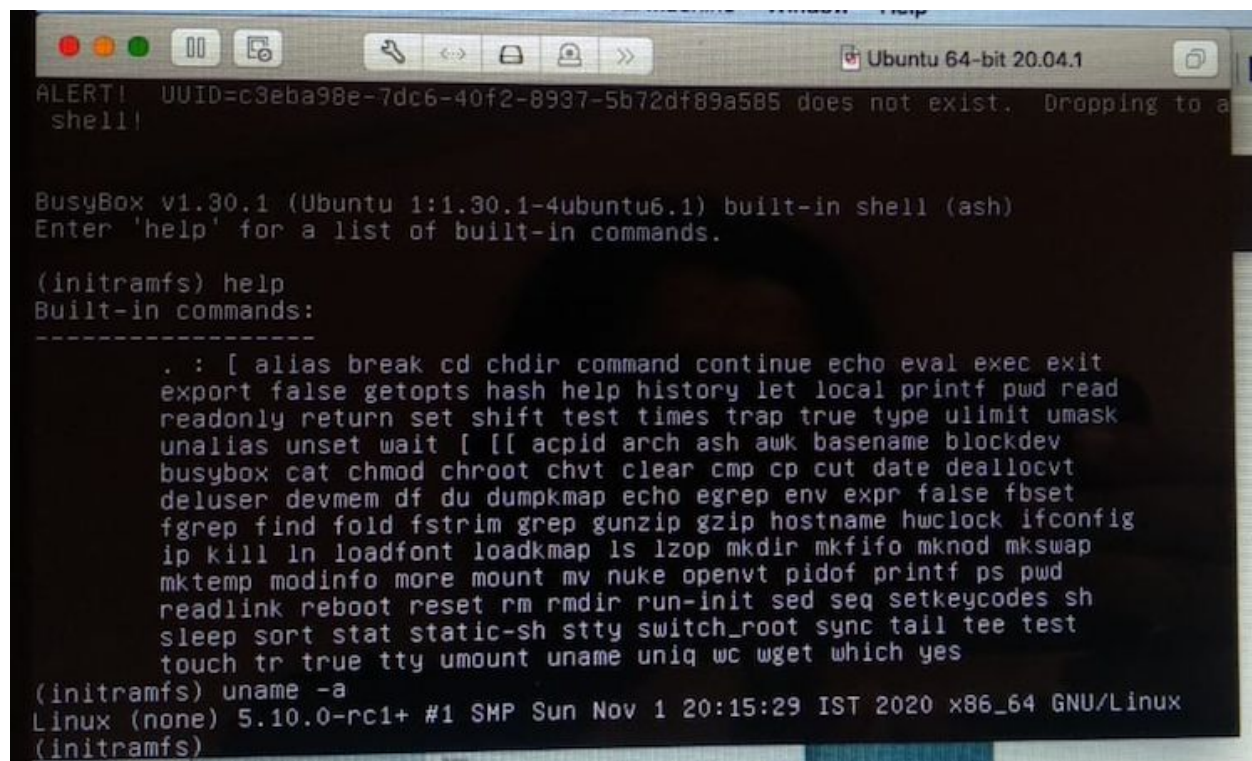


We have followed the below steps to setup the kernel and get the system ready for Assignment 2:

Compile the new kernel code with below commands:

1. We are using Ubuntu as our virtual machine which is mounted over VMware fusion.
2. We forked and cloned the git repository <https://github.com/torvalds/linux> over OS.
3. This OS(ubuntu) has KVM capabilities enabled (verified with KVM-ok command as well) and VMX\SVM has exposed
4. We built the Linux kernel source code(assignment requirement) with below commands under the Linux Folder:
 - a. `sudo bash`
 - b. `apt-get install build-essential kernel-package fakeroot libncurses5-dev libssl-dev ccache bison flex libelf-dev`
 - c. `uname -a`
 - d. `cp /boot/config-4.15.0-112-generic ./config`
 - e. `make oldconfig`
 - f. `make && make modules && make install && make modules-install`
 - g. Reboot

After Reboot, we faced below error and unable to see the upgraded kernel version (5.10)



```
ALERT! UID=c3eba98e-7dc6-40f2-8937-5b72df89a585 does not exist. Dropping to a
shell!

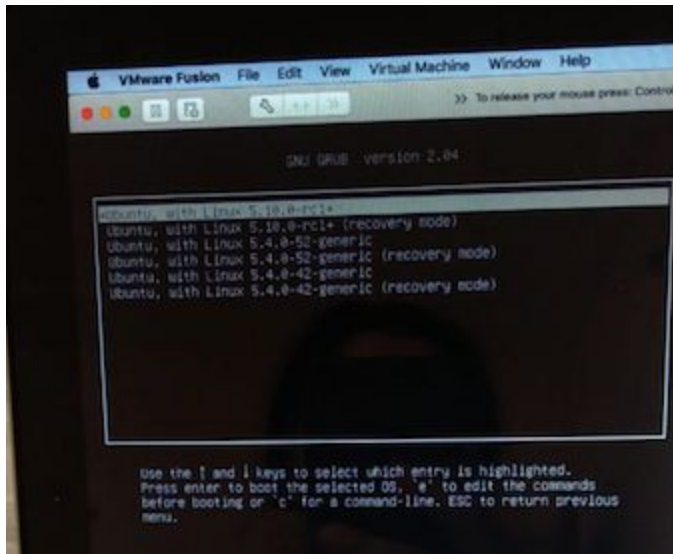
BusyBox v1.30.1 (Ubuntu 1:1.30.1-4ubuntu6.1) built-in shell (ash)
Enter 'help' for a list of built-in commands.

(initramfs) help
Built-in commands:
-----
. : [ alias break cd chdir command continue echo eval exec exit
export false getopt hash help history let local printf pwd read
readonly return set shift test times trap true type ulimit umask
unalias unset wait [ [[ acpid arch ash awk basename blockdev
busybox cat chmod chroot chvt clear cmp cp cut date deallocvt
deluser devmem df du dumpkmap echo egrep env expr false fbset
fgrep find fold fstrim grep gunzip gzip hostname hwclock ifconfig
ip kill ln loadfont loadkmap ls lzop mkdir mkfifo mknod mkswap
mktemp modinfo more mount mv nuke openvt pidof printf ps pwd
readlink reboot reset rm rmdir run-init sed seq setkeycodes sh
sleep sort stat static-sh stty switch_root sync tail tee test
touch tr true tty umount uname uniq wc wget which yes

(initramfs) uname -a
Linux (none) 5.10.0-rc1+ #1 SMP Sun Nov 1 20:15:29 IST 2020 x86_64 GNU/Linux
(initramfs)
```

To fix the above error, we ran below commands and were able to build the kernel to 5.10 version.

1. Reboot -f
2. We selected the Generic code version from the available list, which was before reboot with the generic one select and hit enter.



- 3.
4. We went linux source folder and followed below commands
5. Sudo bash
6. `sudo make -j 2` && `sudo make modules_install -j 2` && `sudo make install -j 2`
(nproc command provided the CPU number output 2)
7. `update-grub`
8. Make
9. Reboot
10. Verified the version with command "`uname -r`" and Kernel is ready.

```
kunjan@ubuntu:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
kunjan@ubuntu:~$ free -g
              total          used          free      shared  buff/cache   available
Mem:           3             0             1           0           0           2
Swap:          1             0             1           0           0           0
kunjan@ubuntu:~$ uname -mrs
Linux 5.10.0-rc1+ x86_64
kunjan@ubuntu:~$
```

Now install KVM QEMU

As Kernel(5.10) is ready, now we installed the KVM(hypervisor) over host OS and referred below link:

<https://www.tecmint.com/install-kvm-on-ubuntu/>

1. `$ egrep -c '(vmx|svm)' /proc/cpuinfo` (will verify virtualization support if value is more then 0)
2. `$ sudo kvm-ok`
3. `$ sudo apt install cpu-checker`
4. `$ sudo apt install -y qemu qemu-kvm libvirt-daemon libvirt-clients bridge-utils virt-manager`
5. `$ sudo systemctl status libvirtd`
6. `$ sudo systemctl enable --now libvirtd`
7. `$ lsmod | grep -i kvm`

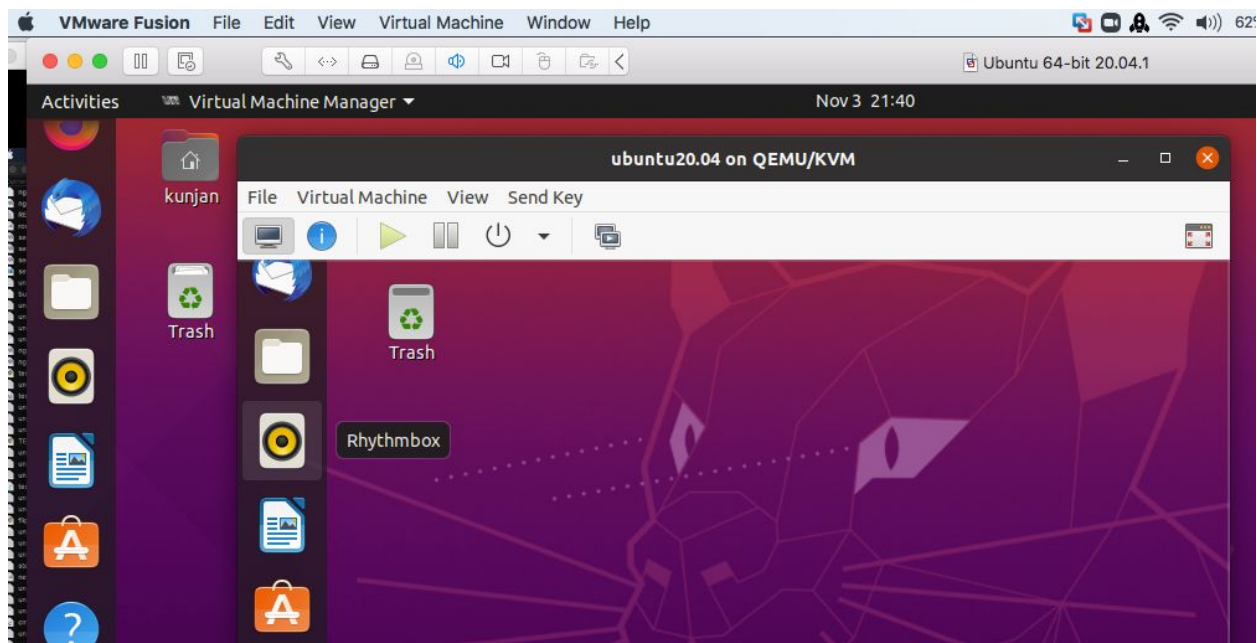
While, we were executing above commands we got error “openc1-amdgpu-pro-orca” to fix this we referred below link:

<https://unix.stackexchange.com/questions/546572/virt-manager-kvm-is-not-available>

Now hypervisor is installed and we are now able to launch virt-manager with command (`sudo virt-manager`) or can be opened directly from the application section.

We need to install the Linux(ubuntu) OS over the hypervisor through virt manager.

1. Upload the iso image file under new VM
2. Install the ubuntu and reboot.
3. We are able to login to our inner VM (ubuntu) as below:



Our assignment requirement is to modify the CPUID emulation code in KVM to report back additional information when a special CPUID “leaf function” is called.

1. Modify the kernel code into two files over host(Ubuntu) machine at below location:
~linux/arch/x86/kvm/cupid.c
~linux/arch/x86/kvm/vmx/vmx.c
2. We ran the following command in host machine under Linux directory:
-sudo rmmod kvm_intel
-sudo rmmod kvm
-sudo make SUBDIRS= arch/x86/kvm/
-sudo insmod arch/x86/kvm/kvm.ko
-sudo insmod arch/x86/kvm/kvm-intel.ko
3. We have start the guest or inner VM from virt manager
4. Create a shell test script file(test_script.sh) as below inside inner VM (once inner VM is up)

```
#!/bin/bash
```

```
cpuid="0x4FFFFFFF"
```

```
output=$(cpuid -l $cpuid-1)
```

```
eax=$(echo $output | grep 0x4ffffff | awk '{print toupper($4)}' | cut -c7-14)
```

```
ebx=$(echo $output | grep 0x4ffffff | awk '{print toupper($5)}' | cut -c7-14)
```

```
ecx=$(echo $output | grep 0x4ffffff | awk '{print toupper($6)}' | cut -c7-14)
```

```
exits=$(echo "ibase=16; $eax" | bc)
```

```
cycles=$(echo "ibase=16; $ebx$ecx" | bc)
```

```
echo "CPUID($cpuid), exits=$exits, cycles spent in exit=$cycles"
```

5. We have checked the output of number of VM exit and CPU cycles over inner VM as below:

***kunjan@kunjan-Standard-PC-Q35-ICH9-2009:~/Documents\$
./test_script.sh CPUID(0x4FFFFFFF), exits=2325981, cycles spent in
exit=5452092626***

6. We checked at Host machine with “dmesg” command

***30282.107579] Exit Count : 2325981
[30282.107581] EBX : 12
[30282.107581] ECX : 2981318754
[30282.107894] Exit Count : 2332268
[30282.107896] EBX : 12
[30282.107896] ECX : 3021617040***

1. For each member in your team, provide 1 paragraph detailing what parts of the lab that member implemented / researched. (You may skip this question if you are doing the lab by yourself).

We worked in a team of 2.

Shivam Shrivastav and Kunjan Malik

We both have researched together about how to install and run installed VMFusion, Ubuntu Host VM and Inner VM in both of our systems.

First we built the solution separately without the atomic read to check if we are at least able to achieve the main requirement.

We shared the load, where Kunjan worked on the total exits part in vmx.c and cpuid.

Shivam worked on total cycles in cpuid.c and vmx.c. We then integrated our parts and worked together to build a complete solution.

After the successful results, we worked on handling concurrent exits in multi-VCPU VM .

This was achieved using atomic read and atomic increment functionality. With this, we built the complete solution and tested our output using the above mentioned test script.

2. Steps described in the readme.md file.

You can refer to the delta parts(code changes to achieve the requirement) in the github respective files.

3. Comment on the frequency of exits – does the number of exits increase at a stable rate? Or are there more exits performed during certain VM operations? Approximately how many exits does a full VM boot entail?

It does not seem to be increasing at a stable rate. When we run a variety of VM instructions in our inner VM, say unauthorized file access, playing video on the VM or internet or other VM operations., more number of VM exits occur due to Page Fault, exceptions etc accordingly.

In our setup, approximately, 2325981 exits occur within 54520926306 cpu cycles after VM boot.