Normi – OS

Introduction:

The NORMI-OS is GUI based Upgraded linux based Operating system which can Impress normal people to use linux based OS

Linux can serve as the basis for nearly any type of IT initiative, including containers, cloud-native applications, and security. It's at the core of some of the biggest industries and businesses in the world from knowledge-sharing websites like Wikipedia to the New York Stock Exchange to mobile devices running Android (which is a specific-use distribution of the Linux kernel with complimentary software). Linux has grown over the years to be the de facto standard for running highly available, reliable, and critical workloads in datacenters and cloud deployments. It has multiple use cases, distributions, target systems and devices, and capabilities all depending on your needs and workloads.

Very easy. Almost every Linux distribution comes with a graphic-user interface (GUI) that makes point-and-click actions very easy. These GUIs revolutionized computing during the OS wars in the late 1970s by humanizing software making it more visual. More real. But at some point, you might need to ask the computer to do something outside the prepackaged bucket of actions a GUI can perform. At that point, you need to open up the command line the strips of code you see tech-savvy geniuses in TV and movies feverishly typing away at. But that Hollywood treatment makes the command line (a staple of open source OSs) seem like something impossible to master.

The truth is, the command line is the most straightforward use of a computer. But there are some Linux distros like our latest release of Red Hat Enterprise Linux that make using open source OSs even easier.

Objectives:

To Develop a GUI based Upgraded linux based Operating system named as Normi-OS which can Impress normal people to use linux based OS.

Abstract:

Linux is a family of open-source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds. Linux is typically packaged as a Linux distribution, which includes the kernel and supporting system software and libraries, many of which are provided by the GNU Project. Many Linux distributions use the word "Linux" in their name, but the Free Software Foundation uses the name "GNU/Linux" to emphasize the importance of GNU software, causing some controversy. Popular Linux distributions include Debian, Fedora Linux, and Ubuntu, the latter of which itself consists of many different distributions and modifications, including Lubuntu and Xubuntu. Commercial distributions include Red Hat Enterprise Linux and SUSE Linux Enterprise. Desktop Linux distributions include a windowing system such as X11 or Wayland, and a desktop environment such as GNOME or KDE Plasma. Distributions intended for servers may omit graphics altogether, or include a solution stack such as LAMP. Because Linux is freely redistributable, anyone may create a distribution for any purpose. Linux is the most secure operating system, It may take time for people to adapt. GUI based Upgraded linux based Operating system named as Normi-OS which can Impress normal people to use linux based OS. This OS includes the following modules : Baseline Profile Files Module, Buliding personal ISO Module, Multiple profile Building Module, Buliding Normi-OS ISO Module. Each linux distro has its own purpose to fullfill the needs. For Normal people aka. Normi, Normi – OS come into play to fullfill all needs of all kind people.

Problem Description:

The Microsoft windows operating system has following disavantages like vulnerable to security threats. One such is the attack from hackers. Cybercriminals often target Windows OS due to its high popularity. Which is one of the reasons why Windows is poor when it comes to security.It requires very demanding system resources compared to other operating systems. You cannot run current version of windows on an older machine which was built 10 years ago.Its Majority of these applications include a fee (Either one time payment or subscription on a regular basis). Eventually, you will end up spending money on each and every software. To fix above mentioned swithing to linux was the only solution which is Linux is an open source operating system. A Linux distribution[a] (often abbreviated as distro) is an operating system made from a software collection that includes the Linux kernel, and often a package management system. Linux users usually obtain their operating system by downloading one of the Linux distributions, which are available for a wide variety of systems ranging from embedded devices (for example, OpenWrt) and personal computers (for example, Linux Mint) to powerful supercomputers (for example, Rocks Cluster Distribution). The linux distributions are quite complex by theory, In practical its not true. This project helps people switch to linux easily by making suitable OS for their need of use.

Literature Servey :

Existing System:

Microsoft Windows [4] :

Windows is a group of several proprietary graphical operating system families developed and marketed by Microsoft. Each family caters to a certain sector of the computing industry. Windows is the most popular desktop operating system in the world, with 75% market share as of April 2022, according to StatCounter. However, Windows is not the most used operating system when including both mobile and desktop OSes, due to Android's massive growth.

Advantages:

- Beginner-friendly.
- Pre-installed on many devices

Disadvantages:

- Less secure than Linux.
- Proprietary software.
- Can slow down over time.
- Can have bugs and reliability issues.
- Must be purchased.
- Subject to vulnerabilities.
- Does have data collection, though it can be turned off.
- Ads in Windows Search.

Cononical’s : Ubuntu [5] :

Ubuntu is a Linux distribution based on Debian and composed mostly of free and open-source software. Ubuntu is officially released in three editions: Desktop, Server, and Core for Internet of things devices and robots. All of the editions can run on a computer alone, or in a virtual machine. Ubuntu is a popular operating system for cloud computing, with support for OpenStack. Ubuntu's default desktop changed back from the in-house Unity to GNOME after nearly 6.5 years in 2017 upon the release of version 17.10. Ubuntu is released every six months, with long-term support (LTS) releases every two years. As of October 2022, the most-recent release is 22.10 ("Kinetic Kudu"), and the current long-term support release is 22.04 ("Jammy Jellyfish").

Ubuntu is developed by British company Canonical, and a community of other developers, under a meritocratic governance model. Canonical provides security updates and support for each Ubuntu release, starting from the release date and until the release reaches its designated end-of-life (EOL) date. Canonical generates revenue through the sale of premium services related to Ubuntu and donations from those who download the Ubuntu software.

Advantages:

- Free Operating System for Personal and Enterprise Computing.
- A Well-Rounded Operating System for Desktop Computing.

Disadvantages:

- Limited Functionality Due to Limited Applications.
- Designed for specific use case.
- Issues About Commercialization Versus Open Source.
- Few customizations.
- Not responsive community.

Garuda Linux [6] :

Garuda Linux is a Linux distribution based on the Arch Linux operating system. Garuda Linux is available in wide range of popular Linux desktop environments, including modified versions of the KDE Plasma 5 desktop environment. It features a rolling release update model using Pacman as its package manager. The term Garuda, which originates from Hinduism, is defined as a divine eagle-like sun bird and the king of birds. Garuda Linux installation process is done with Calamares, a graphical installer. The rolling release model means that the user does not need to upgrade/reinstall the whole operating system to keep it up-to-date inline with the latest release. Garuda Linux uses systemd as its init software. Package management is handled by Pacman via command line, and front-end UI package manager tools such as the pre-installed Pamac. It can be configured as either a stable system (default) or bleeding edge in line with Arch. Garuda Linux includes colorized UI which comes in various options, with the option to further customize the user preferences. Provides Garuda KDE Dragonized Gaming edition.

Advantages:

- Easy installation with Calamares installer.
- Automatic snapshots accessible from GRUB.

Disadvantages:

- Designed for specific use case.
- Need more resources to run.
- Need more Hard Disk space.

BlackArch[6]:

BlackArch is a penetration testing distribution based on Arch Linux that provides a large amount of cyber security tools. It is an open-source distro created specially for penetration testers and security researchers. The repository contains more than 2800 tools that can be installed individually or in groups. BlackArch Linux is compatible with existing Arch Linux installs. BlackArch is similar in usage to both Parrot OS and Kali Linux when fully installed, with a major difference being BlackArch is based on Arch Linux instead of Debian. BlackArch only provides the Xfce desktop environment in the "Slim ISO" but provides multiple preconfigured Window Managers in the "Full ISO". Similar to Kali Linux and Parrot OS, BlackArch can be burned to an ISO image and run as a live system. BlackArch can also be installed as an unofficial user repository on any current Arch Linux installation.

Advantages:

- Most scure linux distribution.
- Designed to audit security.
- Best for penetration testing.

Disadvantages:

- Designed for specific use case.
- Only for advanced user.

Requirement Specifications:

Hardware Requirements:

Processor	: Minimum 2Ghz Processor.
RAM	: Minimum 2.00GB & Recommended 4. GB
Hard Disk	: Minimum 15 GB .
Graphics	: HD Graphics Card.

Monitor.
Mouse.
Keyboard.

Modules in Details:

The Modules in Normi-OS are:

- Basic Profile, ISO Building and Testing Module.
- Personal profile and Personal ISO Building and Testing Module.
- Multiple profiles Module.
- Normi-OS ISO Building and Testing Module.

1. Basic Profile [7], ISO Build and Testing Module :

1.1 Basic Profile :

An Basic profile consists of several configuration files and a directory for files to be added to the resulting image.

1.1.1 Profile structure:

```
profile/
├── airootfs/
├── efiboot/
├── syslinux/
├── grub/
├── bootstrap_packages.arch
├── packages.arch
├── pacman.conf
└── profiledef.sh
```

The required files and directories are explained in the following sections.

1.1.1.1 profiledef.sh

This file describes several attributes of the resulting image and is a place for customization to the general behavior of the image.

The image file is constructed from some of the variables in **profiledef.sh**: **<iso_name>-<iso_version>-<ar ch>.iso**
(e.g. **archlinux-202010-x86_64.iso**).

* **iso_name** : The first part of the name of the resulting image (defaults to ``mkarchiso``)
* **iso_label** : The ISO's volume label (defaults to ``MKARCHISO``)
* **iso_publisher** : A free-form string that states the publisher of the resulting image (defaults to **mkarchiso**)
* **iso_application** : A free-form string that states the application (i.e. its use-case) of the resulting image (defaults
  to **mkarchiso iso**)
* **iso_version** : A string that states the version of the resulting image (defaults to **""**)
* **install_dir** : A string (maximum eight characters long, which **must** consist of **[a-z0-9]**) that states the
  directory on the resulting image into which all files will be installed (defaults to **mkarchiso**)
* **buildmodes** : An optional list of strings, that state the build modes that the profile uses. Only the following are understood:
  - **bootstrap** : Build a compressed file containing a minimal system to bootstrap from
  - **iso** : Build a bootable ISO image (implicit default, if no ``buildmodes`` are set)
  - **netboot** : Build artifacts required for netboot using iPXE
* **bootmodes** : A list of strings, that state the supported boot modes of the resulting image. Only the following are understood:
  - **bios.syslinux.mbr** : Syslinux for x86 BIOS booting from a disk
  - **bios.syslinux.eltorito** : Syslinux for x86 BIOS booting from an optical disc
  - **uefi-ia32.grub.esp** : GRUB for IA32 UEFI booting from a disk
  - **uefi-ia32.grub.eltorito** : GRUB for IA32 UEFI booting from an optical disc
  - **uefi-x64.grub.esp** : GRUB for x86_64 UEFI booting from a disk
  - **uefi-x64.grub.eltorito** : GRUB for x86_64 UEFI booting from an optical disc
  - **uefi-x64.systemd-boot.esp** : systemd-boot for x86_64 UEFI booting from a disk
  - **uefi-x64.systemd-boot.eltorito** : systemd-boot for x86_64 UEFI booting from an optical disc

Note that BIOS El Torito boot mode must always be listed before UEFI El Torito boot mode.
* **arch** : The architecture (e.g. **x86_64**) to build the image for. This is also used to resolve the name of the packages
  file (e.g. **packages.x86_64**)
* **pacman_conf** : The **pacman.conf** to use to install packages to the work directory when creating the image (defaults to
  the host's **/etc/pacman.conf**)
* **airootfs_image_type** : The image type to create. The following options are understood (defaults to **squashfs**):
  - **squashfs** : Create a squashfs image directly from the airootfs work directory
  - **ext4+squashfs** : Create an ext4 partition, copy the airootfs work directory to it and create a squashfs image from it
  - **erofs** : Create an EROFS image for the airootfs work directory
* **airootfs_image_tool_options** : An array of options to pass to the tool to create the airootfs image. **mksquashfs** and **mkfs.erofs** are supported. See **mksquashfs --help** or **mkfs.erofs --help** for all possible options
* **file_permissions** : An associative array that lists files and/or directories who need specific ownership or permissions. The array's keys contain the path and the value is a colon separated list of owner UID, owner GID and access mode. E.g. **file_permissions=(["/etc/shadow"]="0:0:400")**. When directories are listed with a trailing backslash (/) **all** files and directories contained within the listed directory will have the same owner UID, owner GID, and access mode applied recursively.

1.1.1.2 bootstrap_packages.arch

All packages to be installed into the environment of a bootstrap image have to be listed in an architecture specific
file (e.g. **bootstrap_packages.x86_64**), which resides top-level in the profile.
Packages have to be listed one per line. Lines starting with a # and blank lines are ignored.
This file is required when generating bootstrap images using the **bootstrap** build mode.

1.1.1.3 packages.arch
All packages to be installed into the environment of an ISO image have to be listed in an architecture specific file
(e.g. **packages.x86_64**), which resides top-level in the profile.

Packages have to be listed one per line. Lines starting with a # and blank lines are ignored.
This file is required when generating ISO images using the ``iso`` or ``netboot`` build modes.

1.1.1.4 pacman.conf
A configuration for pacman is required per profile.
Some configuration options will not be used or will be modified:
* **CacheDir** : the profile's option is **only** used if it is not the default (i.e. **/var/cache/pacman/pkg**) and if it is
  not the same as the system's option. In all other cases the system's pacman cache is used.
* **HookDir** : it is **always** set to the **/etc/pacman.d/hooks** directory in the work directory's airootfs to allow modification via the profile and ensure interoparability with hosts using dracut (see **#73 [8]**)
* **RootDir** : it is **always** removed, as setting it explicitely otherwise refers to the host's root filesystem (see **man 8 pacman** for further information on the **-r** option used by **pacstrap**)
* **LogFile** : it is **always** removed, as setting it explicitely otherwise refers to the host's pacman log file (see **man 8 pacman** for further information on the **-r** option used by **pacstrap**)
* **DBPath** : it is **always** removed, as setting it explicitely otherwise refers to the host's pacman database (see **man 8 pacman** for further information on the **-r** option used by **pacstrap**)

1.1.1.5 airootfs
This optional directory may contain files and directories that will be copied to the work directory of the resulting image's root filesystem. The files are copied before packages are being installed to work directory location. Ownership and permissions of files and directories from the profile's **airootfs** directory are not preserved. The mode will be **644**

for files and **755** for directories, all of them will be owned by root. To set custom ownership and/or permissions, use **file_permissions** in **profiledef.sh**.
With this overlay structure it is possible to e.g. create users and set passwords for them, by providing **airootfs/etc/passwd** , **airootfs/etc/shadow**, **airootfs/etc/gshadow** (see **man 5 passwd**, **man 5 shadow** and **man 5 gshadow** respectively). If user home directories exist in the profile's **airootfs**, their ownership and (and top-level) permissions will be altered according to the provided information in the password file.

1.1.1.6 Boot loader configuration

A profile may contain configuration for several boot loaders. These reside in specific top-level directories, which are
explained in the following subsections.
The following **custom template identifiers** are understood and will be replaced according to the assignments of the
respective variables in ``profiledef.sh``:
* **%ARCHISO_LABEL%** : Set this using the **iso_label** variable in **profiledef.sh**.
* **%INSTALL_DIR%** : Set this using the **install_dir** variable in **profiledef.sh**.
* **%ARCH%** : Set this using the **arch** variable in **profiledef.sh**.

1.1.1.7 efiboot

This directory is mandatory when the **uefi-x64.systemd-boot.esp** or **uefi-x64.systemd-boot.eltorito** bootmodes are selected in **profiledef.sh** . It contains configuration for **systemd-boot [9]**.
*The **custom template identifiers** are **only** understood in the boot loader entry `.conf` files (i.e. **not** in **loader.conf**).*

1.1.1.8 syslinux

This directory is mandatory when the **bios.syslinux.mbr** or the **bios.syslinux.eltorito** bootmodes are selected in **profiledef.sh**. It contains configuration files for **syslinux[10],** or **isolinux[11]**, or **pxelinux[12]** used in the resulting image.
*The **custom template identifiers** are understood in all `.cfg` files in this directory.*

1.1.1.9 grub

This directory is mandatory when any of the following bootmodes is used in **profiledef.sh** :
- **uefi-ia32.grub.esp** or

- **uefi-ia32.grub.eltorito** or
- **uefi-x64.grub.esp** or
- **uefi-x64.grub.eltorito**

It contains configuration files for **GRUB[13]** used in the resulting image.

1.2 Bulting and Testing :

1.2.1 Build the ISO:

# mkarchiso -v -w */path/to/work_dir* -o */path/to/out_dir /path/to/profile/*

-w specifies the working directory. If the option is not specified, it will default to work in the current directory.

-o specifies the directory where the built ISO image will be placed. If the option is not specified, it will default to out in the current directory.

It should be noted the profile file **profiledef.sh** cannot be specified when running mkarchiso, only the path to the file.

2. Personal profile, ISO Building Module :

2.1 Prepare a Personal profile:

Archiso comes with two profiles, **releng** and **baseline**.
* **releng** is used to create the official monthly installation ISO. It can be used as a starting point for creating a customized ISO image.
* **baseline** is a minimalistic configuration, that includes only the bare minimum packages required to boot the live environment from the medium.
Full Documentation are added in previous module **1.1** for reference (see **[9]**)

2.1.1 Profile structure

Full Documentation are added in previous module **1.1** for reference (see **[9]**)

2.1.2 Selecting packages

Edit **packages.x86_64** to select which packages are to be installed on the live system image, listing packages line by line.

2.1.2.1 Custom local repository

To add packages not located in standard Arch repositories (e.g. custom packages or packages from **AUR[14]**/**ABS[15]**), set up a **custom local repository[16]** and add your custom packages to it. Then add your repository to **pacman.conf** as follows: dl.dsp

```
personalprofile/pacman.conf
...
[customrepo]
SigLevel = Optional TrustAll
Server = file:///path/to/customrepo
…
```

2.1.2.2 Packages from multilib

To install packages from the **multilib[17]** repository, simply uncomment that repository in **pacman.conf.**

2.1.3 Adding files to image

The airootfs directory is used as the starting point for the **root directory[18]** (/) of the live system on the image. All its contents will be copied over to the working directory before packages are installed.

Place any custom files and/or directories in the desired location under **airootfs/**. For example, if you have a set of iptables scripts on your current system you want to be used on you live image, copy them over as such:

$ cp -r /etc/iptables archlive/airootfs/etc

Similarly, some care is required for special configuration files that reside somewhere down the hierarchy. Missing parts of the directory structure can be simply created with **mkdir(1)[19].**

By default, **permissions[20]** will be **644** for files and **755** for directories. All of them will be owned by the root user. To set different permissions or ownership for specific files and/or folders, use the file_permissions associative array in profiledef.sh. See **README.profile.rst[7]** for details.

2.1.4 Adding repositories to the image:

To add a repository that can be used in the live environment, create a **suitably modified[21]** pacman.conf and place it in **personalprofile/airootfs/etc/**.

If the repository also uses a key, place the key in **personalprofile/airootfs/usr/share/pacman/keyrings/**. The key file name must end with **.gpg**. Additionally, the key must be trusted. This can be accomplished by creating a **GnuPG** exported trust file in the same directory. The file name must end with **-trusted**.

The first field is the key fingerprint, and the second is the trust. You can reference **/usr/share/pacman/keyrings/archlinux-trusted** for an example.

2.1.4.1 archzfs example

The files in this example are:

```
airootfs
├── etc
│   ├── pacman.conf
│   └── pacman.d
│       └── archzfs_mirrorlist
└── usr
    └── share
        └── pacman
            └── keyrings
                ├── archzfs.gpg
                └── archzfs-trusted
```

airootfs/etc/pacman.conf

```
...
[archzfs]
Include = /etc/pacman.d/archzfs_mirrorlist
…
```

airootfs/etc/pacman.d/archzfs_mirrorlist

```
Server = https://archzfs.com/$repo/$arch
Server = https://mirror.sum7.eu/archlinux/archzfs/$repo/$arch
Server = https://mirror.biocrafting.net/archlinux/archzfs/$repo/$arch
Server = https://mirror.in.themindsmaze.com/archzfs/$repo/$arch
Server = https://zxcvfdsa.com/archzfs/$repo/$arch
```

airootfs/usr/share/pacman/keyrings/archzfs-trusted

```
DDF7DB817396A49B2A2723F7403BD972F75D9D76:4:
```

archzfs.gpg itself can be obtained directly from the repository site at **[22]**

2.1.5 Kernel

Although both archiso's included profiles only have **linux[23]**, ISOs can be made to include other or even multiple **kernels[24]**.

First, edit **packages.x86_64** to include kernel package names that you want. When ***mkarchiso*** runs, it will include all ***work_dir*/airootfs/boot/vmlinuz-*** and

work_dir/boot/initramfs-*.img files in the ISO (and additionally in the FAT image used for UEFI booting).

mkinitcpio[25] presets by default will build fallback initramfs images. For an ISO, the main initramfs image would not typically include the autodetect hook, thus making an additional fallback image unnecessary. To prevent the creation of an fallback initramfs image, so that it does not take up space or slow down the build process, place a custom preset in archlive/airootfs/etc/mkinitcpio.d/pkgbase.preset. For example, for **linux-lts[26]**:

Multiple profiles Module:

Full Documentation are added in previous module 1.1 for reference (see [9])

Normi Profile :

Ready to go version of iso which contains all the tools and software installed for users who wants use linux as daily driver.

Media Production Profile :

Ready to go version of iso which contains all the tools required for media production work.

Gaming Profile :

Ready to go version of iso which contains all the components needed for gaming with preconfigured for all types of hardwares.

Penetration testing:

Ready to go version of iso which contains all tool needed for penetration testing and preconfigured as which can run on all harwares.

Normi-OS ISO Building and Testing Module:

Full Documentation are added in previous module 1.1 for reference (see [9])

Reference:

1. https://en.wikipedia.org/wiki/Linux_distribution
2. https://www.redhat.com/en/topics/linux/what-is-linux
3. https://www.redhat.com/en/topics/linux
4. https://www.microsoft.com/en-us/windows
5. https://ubuntu.com/
6. https://www.blackarch.org/
7. https://gitlab.archlinux.org/archlinux/archiso/-/blob/master/docs/README.profile.rst
8. https://gitlab.archlinux.org/archlinux/archiso/-/issues/73
9. https://www.freedesktop.org/wiki/Software/systemd/systemd-boot
10. https://wiki.syslinux.org/wiki/index.php?title=SYSLINUX
11. https://wiki.syslinux.org/wiki/index.php?title=ISOLINUX
12. https://wiki.syslinux.org/wiki/index.php?title=PXELINUX
13. https://www.gnu.org/software/grub/
14. https://wiki.archlinux.org/title/AUR
15. https://wiki.archlinux.org/title/ABS
16. https://wiki.archlinux.org/title/Custom_local_repository
17. https://wiki.archlinux.org/title/Multilib
18. https://en.wikipedia.org/wiki/Root_directory
19. https://man.archlinux.org/man/mkdir.1
20. https://wiki.archlinux.org/title/Permissions
21. https://wiki.archlinux.org/title/Pacman#Repositories_and_mirrors
22. https://archzfs.com/archzfs.gpg
23. https://archlinux.org/packages/?name=linux
24. https://wiki.archlinux.org/title/Kernels
25. https://wiki.archlinux.org/title/Mkinitcpio
26. https://archlinux.org/packages/?name=linux-lts