



# A Case Study on Tiny Core Linux

**Nijiya Maharjan (078BCT052)**

**Rubika Bashyal (078BCT068)**

**Saksham Sapkota (078BCT072)**

**Submitted To:**

Department of Electronics and Computer Engineering  
Institute of Engineering, Pulchowk Campus

# 1 Acknowledgement

We would like to express our sincere gratitude to everyone who supported us in completing this project. First and foremost, we extend our heartfelt thanks to our teachers, Santosh Giri and Bikal Adhikari, for their invaluable guidance, support, and encouragement throughout the entire process. Their expertise and feedback have been crucial to the development of this work.

We would also like to acknowledge the Department of Electronics and Computer Engineering, Pulchowk Campus for providing the necessary resources and facilities to carry out this study.

A special thanks to our peers and colleagues for their continuous help, constructive suggestions, and sharing their insights, which have significantly contributed to the project's success.

Finally, we would like to extend our gratitude to our families for their understanding and constant support throughout the course of this work. Their patience and encouragement were instrumental in helping us complete this project successfully.

Thank you all for your assistance and contributions.

# Contents

<b>1</b>	<b>Acknowledgement</b>	<b>1</b>
<b>2</b>	<b>Abstract</b>	<b>5</b>
<b>3</b>	<b>Methodology</b>	<b>6</b>
3.1	Core Variants . . . . .	6
3.1.1	Core (17 MB) . . . . .	6
3.1.2	TinyCore (23 MB) . . . . .	6
3.1.3	CorePlus (248 MB) . . . . .	6
3.2	Installation . . . . .	7
3.2.1	Cloud Mode . . . . .	7
3.2.2	USB Stick Installation . . . . .	7
3.2.3	Frugal Install . . . . .	7
3.2.4	VirtualBox Installation . . . . .	8
<b>4</b>	<b>Architecture</b>	<b>9</b>
4.1	Core Components . . . . .	9
4.2	Modular Design . . . . .	10
4.3	Boot Process and Operation Modes . . . . .	10
4.4	Persistence Mechanism . . . . .	11
4.5	File System Layout . . . . .	12
4.5.1	Core System Files . . . . .	12
4.5.2	Persistent User Data (mydata.tgz) . . . . .	13
4.5.3	Extensions (TCZ Files) . . . . .	13
4.5.4	Symbolic Links and Read-Only Extensions . . . . .	13
<b>5</b>	<b>Package Management in Tiny Core Linux</b>	<b>14</b>
5.1	Package Management via GUI . . . . .	14
5.1.1	Installation Methods . . . . .	16
5.2	Package Management via CLI . . . . .	16
5.2.1	Using tce-ab . . . . .	16
5.2.2	Using tce-load . . . . .	17
5.3	Comparison with Other Package Managers . . . . .	17
<b>6</b>	<b>Process Scheduling and Memory Management</b>	<b>18</b>

<b>7</b>	<b>Process Scheduling in Tiny Core Linux</b>	<b>18</b>
7.1	Linux Kernel Scheduler . . . . .	18
7.2	Real-Time Scheduling . . . . .	18
7.3	Cooperative Scheduling . . . . .	18
<b>8</b>	<b>Memory Management in Tiny Core Linux</b>	<b>18</b>
8.1	Efficient Memory Usage . . . . .	18
8.2	Physical and Virtual Memory Management . . . . .	19
8.3	Shared Libraries and Extensions . . . . .	19
<b>9</b>	<b>Modifications to the OS</b>	<b>20</b>
9.1	Shared Folder Setup from Host Computer to Tiny Core Linux . . . .	20
9.1.1	Setting Up Shared Folders in VirtualBox . . . . .	20
9.2	Including Custom Applications . . . . .	21
9.3	Using <code>xbindkeys</code> for Custom Keyboard Shortcuts . . . . .	22
9.4	Changing Icons . . . . .	23
9.5	Changing Wallpaper . . . . .	24
9.6	Changing the position of the task bar . . . . .	25
9.7	Custom Aliases for Commands in Terminal . . . . .	26
9.8	Adding sticky notes . . . . .	28
9.9	Adding <code>htop</code> task manager . . . . .	29
9.10	Creating an ISO with Modifications Using <code>ezremaster</code> . . . . .	31
<b>10</b>	<b>Conclusion</b>	<b>33</b>

## List of Figures

1	Screenshot of Tinycore Desktop . . . . .	9
2	File System Architecture . . . . .	12
3	Package Management - Step 1 . . . . .	14
4	Package Management - Step 2 . . . . .	14
5	Package Management - Step 3 . . . . .	15
6	Setting up shared folder . . . . .	21
7	Custom Keyboard Shortcuts . . . . .	23
8	Changed Icons . . . . .	23
9	Changing Wallpaper - Step 1 . . . . .	24
10	Changing Wallpaper - Step 2 . . . . .	25
11	Changed Wallpaper . . . . .	25
12	Changing position of taskbar . . . . .	26
13	Changed position of taskbar . . . . .	26
14	Adding aliases . . . . .	27
15	Opening ashrc file . . . . .	27
16	Building ashrc file . . . . .	27
17	Custom Alias . . . . .	27
18	alias banau for touch . . . . .	28
19	developers command . . . . .	28
20	alias pls for sudo . . . . .	28
21	alias sara for mv . . . . .	28
22	Sticky Notes . . . . .	29
23	HTOP . . . . .	30
24	EZRemaster Steps . . . . .	31

## 2 Abstract

This document is a case study report prepared as part of the Department of Electronics and Computer Engineering, Pulchowk Campus. It provides an in-depth analysis of TinyCore Linux, a lightweight and modular operating system known for its minimalistic design and efficient resource utilization. The study explores TinyCore's architecture, including its core components and unique design principles, followed by a step-by-step guide on its installation and package management. Additionally, this report details the modifications made to enhance or customize the system, evaluating their impact on usability and performance. Through this analysis, we aim to understand TinyCore's advantages, limitations, and potential applications in lightweight computing environments.

## 3 Methodology

### 3.1 Core Variants

The Core Project is not a pre-configured desktop distribution but rather a minimal Linux base that allows users to customize their system according to their requirements. It provides three different x86 versions that serve as starting points: Core, TinyCore, and CorePlus.

#### 3.1.1 Core (17 MB)

Core is the most minimal version of the project, offering only a command-line interface. It is intended for advanced users who can manually install extensions to create a graphical desktop or a customized environment. This version is ideal for servers, embedded systems, and tailored desktop configurations.

#### 3.1.2 TinyCore (23 MB)

TinyCore is recommended for beginners who have a wired internet connection. It includes the base Core system along with X/GUI extensions, providing a lightweight FLTK/FLWM graphical desktop environment for an easier user experience.

#### 3.1.3 CorePlus (248 MB)

CorePlus is an installation image rather than a standalone distribution. It is designed for new users who require wireless network support or non-US keyboard layouts. This version includes the base Core system along with installation tools to provide the following features:

- A choice of seven window managers.
- Wireless support with various firmware options and ndiswrapper.
- Non-US keyboard compatibility.
- A remastering tool for further customization.

## 3.2 Installation

### 3.2.1 Cloud Mode

In cloud mode, Tiny Core Linux operates entirely in RAM, loading the base system and applications from a remote repository. This approach is beneficial for systems with limited storage or for users seeking a stateless environment. To initiate cloud mode:

1. Boot from the Tiny Core Linux media.
2. At the boot prompt, type `boot` and press Enter.
3. The system will load into RAM and fetch applications as needed from the internet.

### 3.2.2 USB Stick Installation

Installing Tiny Core Linux on a USB stick allows for a portable and persistent system. Follow these steps to create a bootable USB drive:

1. Download the latest Tiny Core Linux ISO from the official website.
2. Use a tool like UNetbootin to write the ISO to the USB stick.
3. Boot the target computer from the USB stick.
4. Follow the on-screen instructions to complete the installation.

**Note:** Ensure that the target computer supports booting from USB devices.

### 3.2.3 Frugal Install

A frugal install involves copying the essential files of Tiny Core Linux to a directory on an existing partition, allowing the system to coexist with other operating systems. To perform a frugal install:

1. Boot from the Tiny Core Linux media.
2. Create a directory on the target partition (e.g., `/mnt/sda1/tinycore`).
3. Copy the `vmlinuz` and `core.gz` files from the installation media to the created directory.



4. Configure the bootloader to include an entry for Tiny Core Linux, pointing to the copied files.

This method is efficient and conserves disk space.

### **3.2.4 VirtualBox Installation**

For the purposes of our case study, we chose to boot the Tiny Core Linux ISO image into VirtualBox. This approach provides an isolated environment for testing and development. The steps are as follows:

1. Download the Tiny Core Linux ISO image.
2. Open VirtualBox and create a new virtual machine:
  - Name: **TinyCore**
  - Type: **Linux**
  - Version: **Linux 2.6 / 3.x / 4.x**
3. Allocate at least 256 MB of RAM to the virtual machine.
4. Create a virtual hard disk (minimum 1 GB recommended).
5. In the virtual machine's settings, attach the downloaded ISO image to the optical drive.
6. Start the virtual machine; it will boot from the ISO image.
7. Once booted, you can proceed to install Tiny Core Linux onto the virtual hard disk or operate it in live mode.

This method allows for easy experimentation without affecting the host system.



Figure 1: Screenshot of Tinycore Desktop

## 4 Architecture

### 4.1 Core Components

The fundamental components of Tiny Core Linux include:

- **Linux Kernel:** The core of the operating system, responsible for managing hardware interactions and system processes. Tiny Core Linux utilizes a monolithic kernel, which integrates all essential services into a single binary, ensuring efficient performance and streamlined operations.
- **BusyBox:** A versatile utility that combines numerous Unix command-line tools into a single executable. BusyBox provides essential functionalities such

as file manipulation, process management, and system monitoring, all within a minimal footprint, contributing to the lightweight nature of Tiny Core Linux.

- **FLTK/FLWM:** The Fast Light Toolkit (FLTK) is a cross-platform graphical user interface toolkit that, in conjunction with the Fast Light Window Manager (FLWM), offers a minimalist and efficient desktop environment. This combination ensures a responsive user experience while maintaining a small resource footprint.

## 4.2 Modular Design

Tiny Core Linux employs a highly modular architecture, allowing users to customize their systems by adding or removing applications and extensions as needed. This design is facilitated through:

- **Extensions:** Additional software packages that can be dynamically loaded into the system. Extensions are stored separately from the core system, enabling users to maintain a minimal base installation. This approach allows for on-demand loading of applications, reducing unnecessary resource usage.
- **Mountable Filesystems:** Extensions are often packaged as mountable filesystems (e.g., `.tcz` files), which can be mounted on demand. This method minimizes the system's memory footprint, as applications are only loaded into RAM when required, ensuring efficient resource utilization.

## 4.3 Boot Process and Operation Modes

Upon booting, Tiny Core Linux loads the kernel and core files into RAM, creating a fast and responsive environment. It supports multiple operation modes:

- **Cloud or Internet Mode:** In this mode, the system operates entirely in RAM, fetching applications from a remote repository as needed. This stateless operation ensures that the system remains lean, with applications loaded temporarily for the current session without permanent installation.
- **TCE/Install Mode:** Extensions are stored on a persistent storage device and loaded into the system at boot time. This configuration allows for a consistent environment across reboots, as user-selected applications and settings are preserved.

- **TCE/CopyFS Mode:** Applications are installed onto a persistent storage device, similar to traditional Linux installations. This mode provides a balance between performance and persistence, enabling faster application access while maintaining system state across sessions.

## 4.4 Persistence Mechanism

While Tiny Core Linux is designed to run from RAM, it offers mechanisms for persisting user data and configurations:

- **Backup/Restore:** Users can create a backup of specified directories and files, which is stored on persistent storage and restored upon boot. This feature ensures that critical data and personalized settings are preserved, even though the system operates primarily in volatile memory.
- **Persistent /home and /opt:** By linking these directories to persistent storage, user data and configurations remain intact across reboots. This setup provides a seamless user experience, as personal files and application settings are consistently available, aligning with traditional operating system behavior.

## 4.5 File System Layout

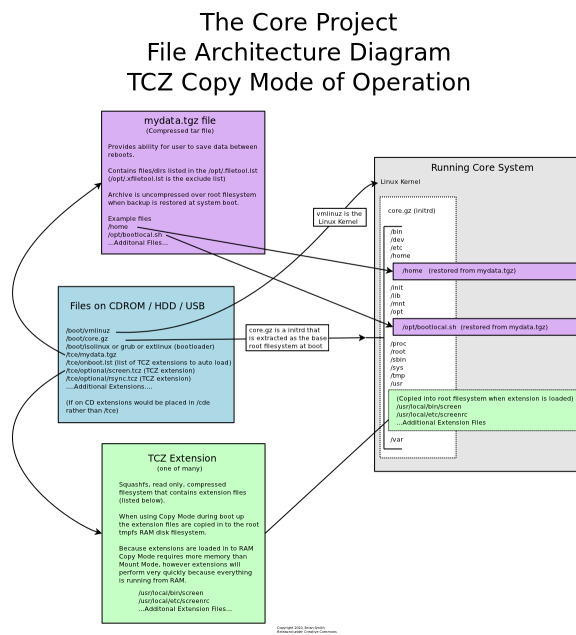


Figure 2: File System Architecture

The file system in Tiny Core Linux is structured to support its modular design:

### 4.5.1 Core System Files

The core of Tiny Core Linux is stored in the `/boot` directory on the storage medium (CDROM, HDD, USB). These files include:

- **/boot/vmlinuz** - The *Linux kernel*, responsible for booting the system and managing hardware interactions.
- **/boot/core.gz** - The *initial RAM disk (initrd)*, which contains the base root file system. It is extracted into RAM during boot, providing the minimal environment necessary for system operation.

Once booted, the system operates entirely from RAM, ensuring fast performance and stateless operation unless persistence is enabled.

### 4.5.2 Persistent User Data (mydata.tgz)

To retain user files and configurations across reboots, Tiny Core Linux uses a compressed archive named `mydata.tgz`, typically located in `/tce/mydata.tgz`. This archive contains:

- **User directories** such as `/home` and `/opt`, ensuring that personal files and configurations persist across sessions.
- **Startup scripts** (e.g., `/opt/bootlocal.sh`), allowing users to execute custom commands upon boot.

During startup, `mydata.tgz` is extracted over the root filesystem, restoring user files and configurations as they were in the previous session.

### 4.5.3 Extensions (TCZ Files)

Tiny Core Linux follows a modular approach where additional applications and utilities are loaded as **TCZ extensions**. These extensions are:

- **Stored in the `/tce/optional/` directory** as compressed SquashFS files.
- **Mounted dynamically** at `/tmp/tcloop/<extension_name>` when loaded.
- **Contain application binaries, libraries, and configuration files**, which are linked into the system.

Once an extension is mounted, symbolic links are created in `/usr/local/` to provide seamless integration with the system. This method reduces memory consumption by allowing on-demand loading of software.

### 4.5.4 Symbolic Links and Read-Only Extensions

When an extension is mounted at `/tmp/tcloop/<extension_name>`, its contents remain **read-only**. To make applications available to the system, symbolic links are created:

- **Example:** If the `screen.tcz` extension is loaded:
  - The actual binary is stored at `/tmp/tcloop/screen/usr/local/bin/screen`.
  - A symbolic link is created at `/usr/local/bin/screen`, pointing to the mounted extension.

This mechanism ensures that the base system remains unchanged while dynamically integrating additional software.

## 5 Package Management in Tiny Core Linux

(ADD IMAGES)

Tiny Core Linux provides a lightweight and modular package management system where software is installed as **extensions** in **.tcz** format. Package management can be performed using either a **Graphical User Interface (GUI)** or the **Command Line Interface (CLI)**.

### 5.1 Package Management via GUI

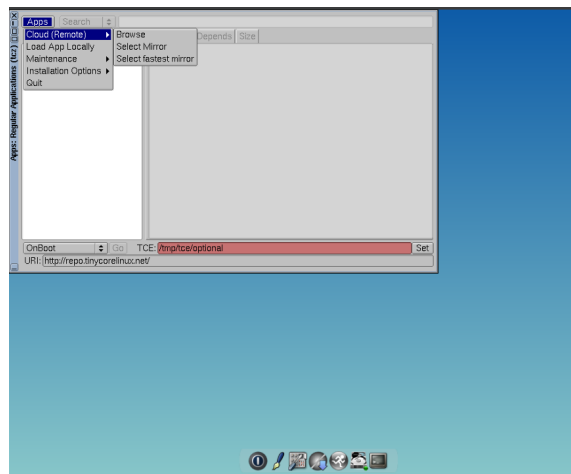


Figure 3: Package Management - Step 1

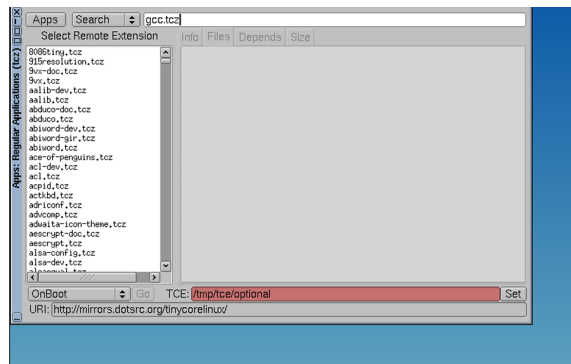


Figure 4: Package Management - Step 2

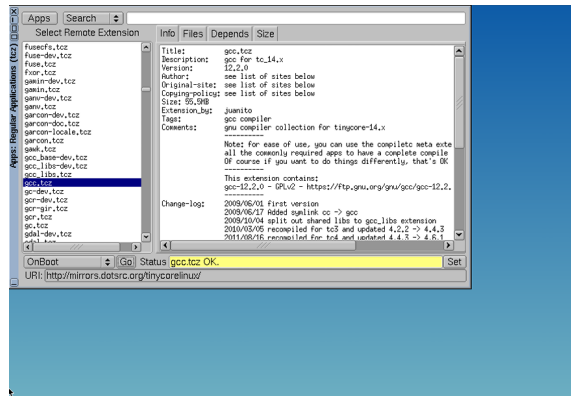


Figure 5: Package Management - Step 3

The **Apps** utility serves as Tiny Core Linux's graphical package manager. It can be launched from the desktop's **Apps** icon or via the menu in alternative window managers. The interface consists of:

- A package list (left panel) displaying available extensions.
- An information panel (right panel) with four tabs:
  - **Extension Info:** Details about the selected package.
  - **File List:** Displays files included in the package.
  - **Dependencies:** Lists required dependencies.
  - **Download Analysis:** Shows total download size.
- The **tce bar** indicating the current package directory (red for RAM, green for persistent storage).
- The **URI bar** displaying the selected package mirror.
- A search function to filter packages by name, tag, or provided files.

To browse packages, users can navigate to:

Apps → Remote → Browse

Once a package is selected, installation options include:



### 5.1.1 Installation Methods

- **OnBoot**: Installs the package and ensures it loads at boot.
- **OnDemand**: Creates a script to load the package when first used, reducing boot time but increasing first-use load time.
- **Download + Load**: Installs the package for the current session but does not persist after reboot.
- **Download Only**: Downloads the package without installing it.

Selecting **OnBoot** and clicking **Go** installs the package, displaying a progress window. If an error occurs (e.g., network failure, checksum error), an alert is shown.

## 5.2 Package Management via CLI

Tiny Core Linux provides the **tce-ab** (Application Browser) and **tce-load** commands for package management in the terminal.

### 5.2.1 Using tce-ab

```
$ tce-ab
tce-ab - Tiny Core Extension: Application Browser
S)earch P)rovides K)eywords Q)uit:
```

The search functions include:

- **Search**: Finds packages by name.
- **Provides**: Finds packages that provide specific files.
- **Keywords**: Searches packages by tag.

Once a package is selected, users can view its information file. After quitting the viewer, tce-ab offers various actions:

```
A)bout I)nstall O)nDemand D)epends T)ree F)iles siZ)e L)ist S)earch P)rovides K)eywor
```

### 5.2.2 Using tce-load

tce-load is a non-interactive command-line tool that automates package installation. The available options are:

```
$ tce-load -h
Usage: tce-load [-i -w -wi -wo -wil -ic -wic]{s} extensions
-i    Loads local extension
-w    Download extension only
-wi   Download and install extension
-wo   Download and create an OnDemand item
```

**Example usage:**

- Install and load a package:

```
$ tce-load -wi ace-of-penguins
```

- Load an already downloaded package:

```
$ tce-load -i ace-of-penguins
```

## 5.3 Comparison with Other Package Managers

Tiny Core Linux's package management system differs from traditional package managers like **apt** (Debian-based) and **yum** (RPM-based). The table below summarizes some common operations:

Operation	APT (Debian)	YUM (RPM)	TCE-Load (TCZ)
Install from repos	apt-get install pkg	yum install pkg	tce-load -wi pkg
Install from a local file	dpkg -i pkg	yum localinstall pkg	tce-load -i pkg
Search for a package	apt-cache search pattern	yum search pattern	tce-ab
List installed packages	dpkg -l	rpm -qa	ls /usr/local/tce.installed

Table 1: Comparison of Package Managers

## 6 Process Scheduling and Memory Management

## 7 Process Scheduling in Tiny Core Linux

### 7.1 Linux Kernel Scheduler

Tiny Core Linux utilizes the Linux kernel's process scheduling capabilities. The default scheduler in the Linux kernel is the Completely Fair Scheduler (CFS), which aims to allocate CPU time to processes in a fair manner. CFS maintains a virtual runtime (`vruntime`) for each process, representing the amount of CPU time a process has received, normalized by the system load. The scheduler selects the process with the smallest `vruntime` to run next, ensuring equitable distribution of CPU resources [?].

### 7.2 Real-Time Scheduling

For tasks requiring stringent timing constraints, the Linux kernel provides real-time scheduling policies such as `SCHED_FIFO` and `SCHED_RR`. These policies prioritize real-time processes over normal processes, ensuring that critical tasks are executed within specific time constraints.

### 7.3 Cooperative Scheduling

In addition to the preemptive multitasking provided by the Linux kernel, Tiny Core Linux can support cooperative scheduling, where processes voluntarily yield control, allowing other processes to run. This approach requires processes to indicate when they are idle or waiting for events, facilitating a single-threaded execution model where one process operates at a time. Developers must design processes to yield appropriately, ensuring fairness and responsiveness. While this simplifies the operating system, synchronization mechanisms like semaphores are crucial for coordination and avoiding deadlocks.

## 8 Memory Management in Tiny Core Linux

### 8.1 Efficient Memory Usage

Tiny Core Linux is engineered to operate with minimal memory overhead. The base system can run entirely in RAM, with the core component requiring as little as 11

MB of disk space. This design facilitates rapid boot times and efficient memory utilization, making it ideal for systems with constrained resources [?].

## **8.2 Physical and Virtual Memory Management**

The Linux kernel manages both physical RAM and virtual memory. It allocates memory pages for processes using functions like `kmalloc()` and enforces memory protection to prevent unauthorized access. Virtual memory techniques, such as paging, are employed to manage memory effectively. When physical RAM is low, the kernel swaps out less-used pages to swap space on disk, ensuring that active processes have the necessary memory resources.

## **8.3 Shared Libraries and Extensions**

Shared libraries allow multiple processes to use the same code, reducing overall memory usage. Tiny Core Linux utilizes extensions (TCEs) to provide additional functionality. These extensions allocate memory for their execution and can be loaded into RAM or mounted from persistent storage, depending on user preferences and system resources.

## 9 Modifications to the OS

### 9.1 Shared Folder Setup from Host Computer to Tiny Core Linux

To facilitate seamless file sharing between the host system and Tiny Core Linux, especially when running Tiny Core within a virtual machine, you can set up shared folders. This allows both the host and guest systems to access and modify files in a common directory, enhancing interoperability and efficiency.

#### 9.1.1 Setting Up Shared Folders in VirtualBox

Follow these steps to configure a shared folder between a host system and Tiny Core Linux running in VirtualBox:

**1. Install VirtualBox Guest Additions:**

- Start your Tiny Core Linux virtual machine.
- In the VirtualBox menu, navigate to **Devices** → **Insert Guest Additions CD Image**.
- Mount the CD-ROM:

```
mount /mnt/sr0
```

- Run the Guest Additions installer:

```
sudo /mnt/sr0/VBoxLinuxAdditions.run
```

- Reboot the virtual machine to apply changes.

**2. Create a Shared Folder on the Host:**

- On the host system, create a directory intended for sharing (e.g., `C:\SharedFolder`).
- In VirtualBox, right-click your Tiny Core VM and select **Settings**.
- Navigate to **Shared Folders** and click the **Add Folder** icon.
- Choose the folder path on the host, assign a folder name (e.g., `SharedFolder`), and enable **Auto-mount** and **Make Permanent** options.

**3. Mount the Shared Folder in Tiny Core Linux:**

- Boot into Tiny Core Linux.
- Load the vboxsf module:

```
sudo modprobe vboxsf
```

- Create a mount point:

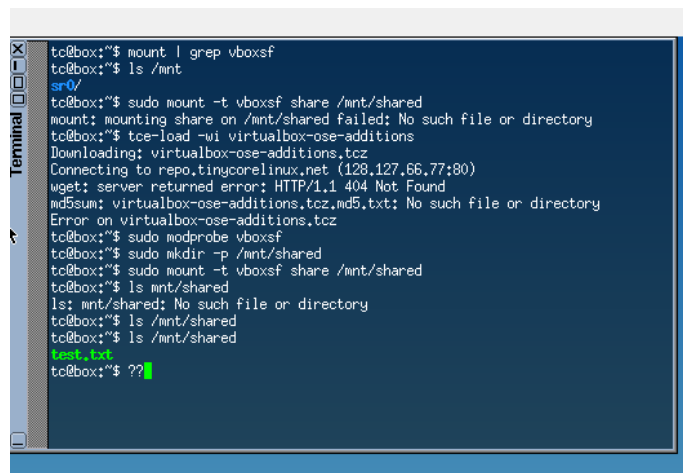
```
mkdir ~/shared
```

- Mount the shared folder:

```
sudo mount -t vboxsf SharedFolder ~/shared
```

- Verify access by listing the contents:

```
ls ~/shared
```



```

tc@box:~$ mount | grep vboxsf
tc@box:~$ ls /mnt
sr0/
tc@box:~$ sudo mount -t vboxsf share /mnt/shared
mount: mounting share on /mnt/shared failed: No such file or directory
tc@box:~$ tce-load -wi virtualbox-ose-additions
Downloading: virtualbox-ose-additions.tcz
Connecting to repo.tinycorelinux.net (128.127.66.77:80)
wget: server returned error: HTTP/1.1 404 Not Found
md5sum: virtualbox-ose-additions.tcz.md5.txt: No such file or directory
Error on virtualbox-ose-additions.tcz
tc@box:~$ sudo modprobe vboxsf
tc@box:~$ sudo mkdir -p /mnt/shared
tc@box:~$ sudo mount -t vboxsf share /mnt/shared
tc@box:~$ ls /mnt/shared
ls: /mnt/shared: No such file or directory
tc@box:~$ ls /mnt/shared
tc@box:~$ ls /mnt/shared
tc@box:~$ ls /mnt/shared
test.txt
tc@box:~$ ??

```

Figure 6: Setting up shared folder

## 9.2 Including Custom Applications

To enhance functionality, you can install additional applications:

1. Using the App Browser: - Open the App Browser:

`appbrowser`

- Search for the desired application. - Select and install the application.

2. Command Line Installation: - Update the repository:

`tce-update`

- Install the application:

`tce-load -wi <application_name>`

Replace `<application_name>` with the specific package name.

### 9.3 Using xbindkeys for Custom Keyboard Shortcuts

Customize keyboard shortcuts with `xbindkeys`:

1. Install `xbindkeys`:

`tce-load -wi xbindkeys`

2. Create Default Configuration:

`xbindkeys --defaults > ~/.xbindkeysrc`

3. Edit Configuration: - Open `/.xbindkeysrc` in a text editor. - Add key bindings in the format:

```
"command_to_run"  
key_combination
```

Example to set `Ctrl + Alt + t` to open a terminal:

```
"aterm"  
Control+Alt + t
```

4. Start `xbindkeys`:

xbindkeys

### Shotcuts Added

- Alt + t : Open New Terminal
- Alt + f : Open File Manager
- Alt + c : Open Chromium Browser
- Alt + q : Close current window

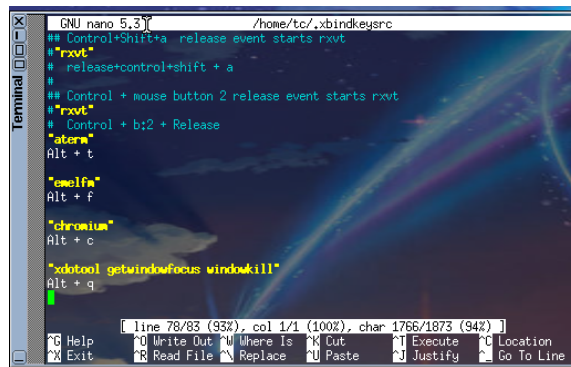


Figure 7: Custom Keyboard Shortcuts

## 9.4 Changing Icons

Customize desktop icons as follows:

1. Locate Icon Files: Default icons are typically stored in `/usr/local/share/icons` or `/usr/share/pixmaps`.
2. Replace Icons: - Prepare your custom icon images. - Replace the existing icon files with your custom images, ensuring they have the same filenames and formats.
3. Refresh Desktop: Restart the window manager or refresh the desktop to apply changes.



Figure 8: Changed Icons



## 9.5 Changing Wallpaper

To set a custom wallpaper:

### 1. Open the Control Panel

- Click on "*Control Panel*" from the wbar (dock) or open it via the terminal using:

```
controlpanel
```

### 2. Open Wallpaper Settings

- In the Control Panel, look for "*Wallpaper*" and click on it.

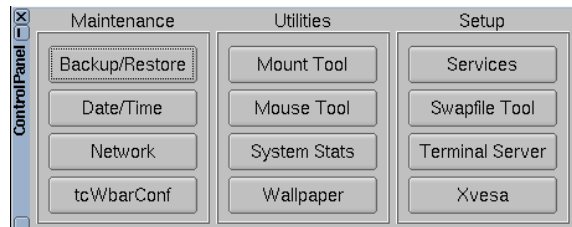


Figure 9: Changing Wallpaper - Step 1

### 3. Choose an Image

- Click "*Browse*" to navigate to the folder containing your wallpaper image.
- Select the desired image and choose from full, tile, corner and fill. We've chosen fill for now.

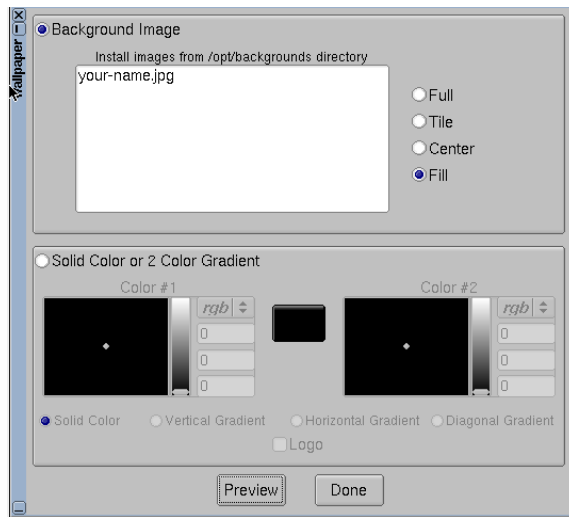


Figure 10: Changing Wallpaper - Step 2

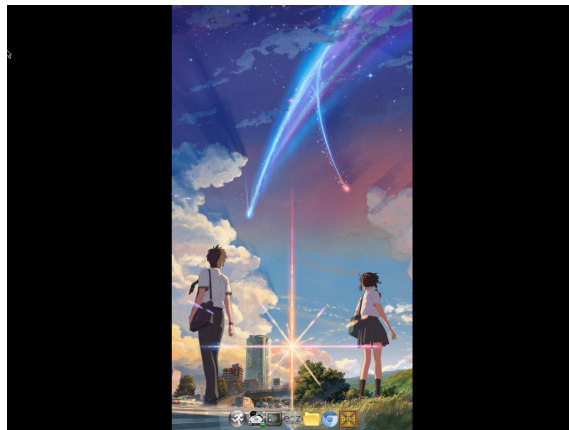


Figure 11: Changed Wallpaper

## 9.6 Changing the position of the task bar

To customize the application launcher (Wbar) in TinyCore OS, follow these steps:

### 1. Right-click on the Desktop

- On the TinyCore OS desktop, perform a right-click to open the context menu.

## 2. Navigate to System Tools

- In the context menu, locate and hover over the **System Tools** option.

## 3. Select tc-wbarconf

- From the **System Tools** submenu, click on **tc-wbarconf**.
- This will open the Wbar configuration tool, allowing you to customize the application launcher.

These steps will help modify the application launcher settings, such as adding or removing shortcuts, changing icons, and adjusting Wbar's appearance.

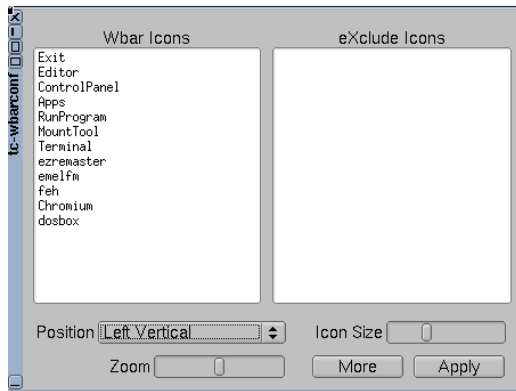


Figure 12: Changing position of taskbar

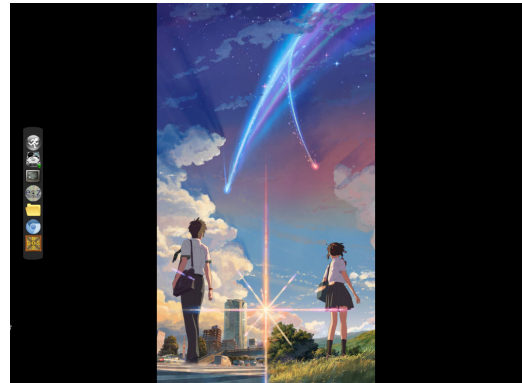


Figure 13: Changed position of taskbar

## 9.7 Custom Aliases for Commands in Terminal

Define command aliases to streamline terminal usage:

1. Edit `.ashrc`: - Open `/.ashrc` in a text editor. - Add aliases in the format:

```
alias shortname='full_command'
```

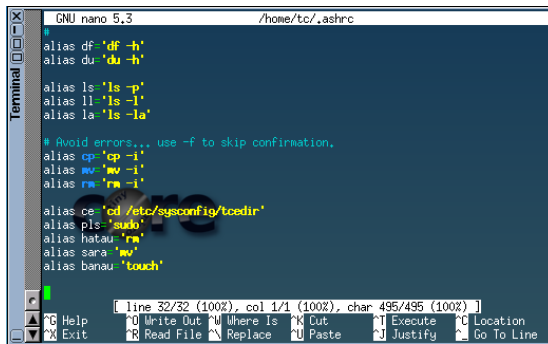


Figure 14: Adding aliases

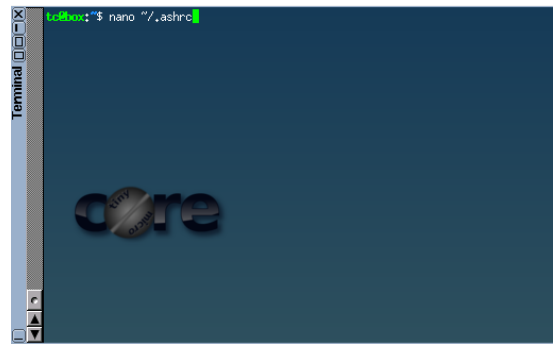


Figure 15: Opening ashrc file



Figure 16: Building ashrc file

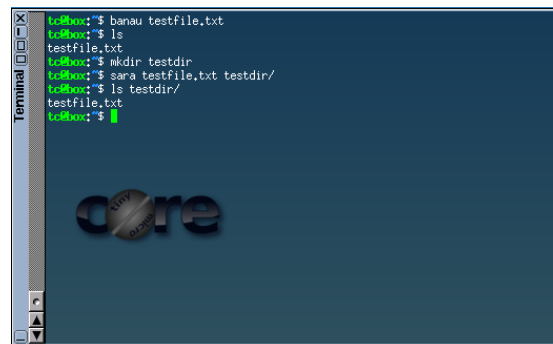
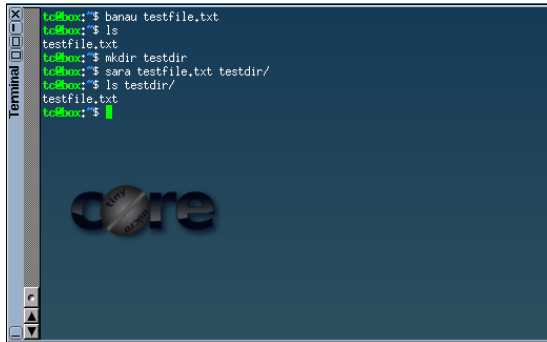


Figure 17: Custom Alias

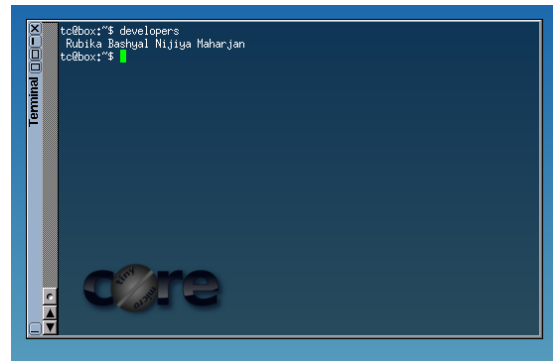
2. Apply Changes: - Reload the profile:

`~/.ashrc`



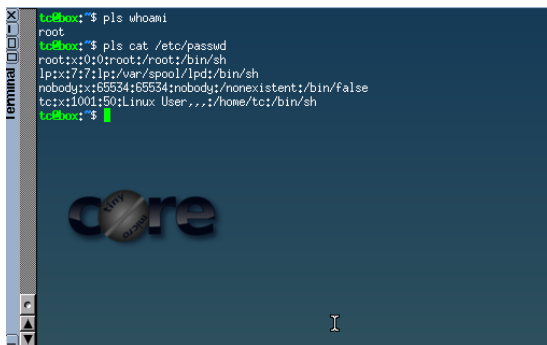
```
tce@box:~$ banau testfile.txt
tce@box:~$ ls
testfile.txt
tce@box:~$ mkdir testdir
tce@box:~$ sara testfile.txt testdir/
tce@box:~$ ls testdir/
testfile.txt
tce@box:~$
```

Figure 18: alias banau for touch



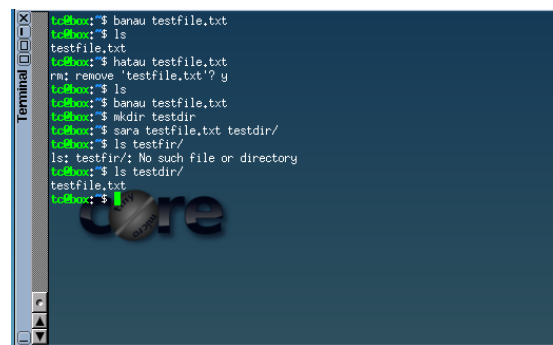
```
tce@box:~$ developers
Rubika Bashyal Nijiya Maharjan
tce@box:~$
```

Figure 19: developers command



```
tce@box:~$ pls whoami
root
tce@box:~$ pls cat /etc/passwd
root:x:0:0:root:/root:/bin/sh
lp:x:72:72:lp/var/spool/lpd:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/false
tce:x:1001:501:Linux User,,,:/home/tce:/bin/sh
tce@box:~$
```

Figure 20: alias pls for sudo



```
tce@box:~$ banau testfile.txt
tce@box:~$ ls
testfile.txt
tce@box:~$ hatau testfile.txt
rm: remove 'testfile.txt'? y
tce@box:~$ ls
tce@box:~$ banau testfile.txt
tce@box:~$ mkdir testdir
tce@box:~$ sara testfile.txt testdir/
tce@box:~$ ls testdir/
ls: testdir/: No such file or directory
tce@box:~$ ls testdir/
testfile.txt
tce@box:~$
```

Figure 21: alias sara for mv

## 9.8 Adding sticky notes

Xpad is a lightweight sticky notes application that allows easy note-taking on Tiny-Core OS.

Steps to Install and Use Xpad

### 1. Install Xpad

- Open a terminal and run the following command to install Xpad:

```
tce-load -wi Xpad
```

### 2. Launch Xpad

- Once installed, open Xpad by running:

xpad

- Alternatively, you can find it in the application menu.

### 3. Create a New Sticky Note

- Click on *"New"* to create a new note.
- Type your reminders or tasks in the sticky note window.

### 4. Customize Your Notes

- Change the background color, font size, or transparency from the settings.
- Right-click on a note to access more options.

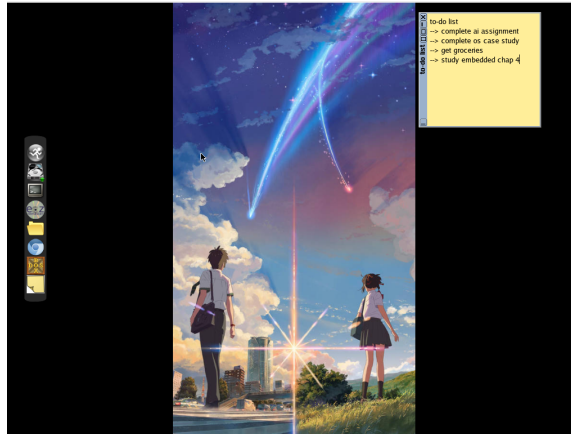


Figure 22: Sticky Notes

## 9.9 Adding htop task manager

**htop** is an interactive process viewer that provides a real-time overview of system resource usage, including CPU, memory, and running processes. It is more user-friendly than the default **top** command, allowing easier navigation and process management.

Steps to Install and Use htop

**htop** is an interactive process viewer that provides a real-time overview of system resource usage, including CPU, memory, and running processes.

It is more user-friendly than the default `top` command, allowing easier navigation and process management. **Install htop**

1. • Open a terminal and install `htop` using the following command:

```
tce-load -wi htop
```

## 2. Launch htop

- Once installed, run `htop` by entering:

```
htop
```

- This will open an interactive interface displaying system resources and running processes.

## 3. Navigate and Monitor Processes

- Use the arrow keys to navigate through the process list.
- Press F9 to kill a selected process.
- Press F10 to exit `htop`.

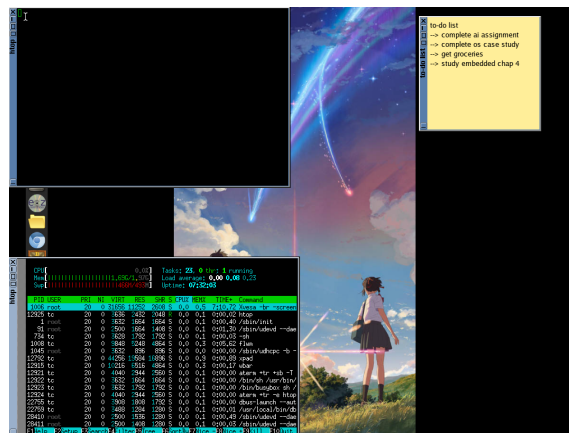


Figure 23: HTOP

On the other hand, PCManFM is a modern, single-pane file manager that provides a more user-friendly experience. It features drag-and-drop support, thumbnail previews, and a customizable graphical interface, making it more intuitive for users familiar with mainstream file managers. Although it requires additional GTK+ dependencies, it remains lightweight and efficient compared to more resource-intensive alternatives. PCManFM is ideal for those who prefer a visually appealing interface with enhanced usability. While it lacks the dual-pane functionality of `emelfm`, it excels in accessibility and ease of use, making it a great choice for general file management.

## 9.10 Creating an ISO with Modifications Using ezremaster

To create a custom ISO incorporating your modifications:

1. Install `ezremaster`:

```
tce-load -wi ezremaster
```

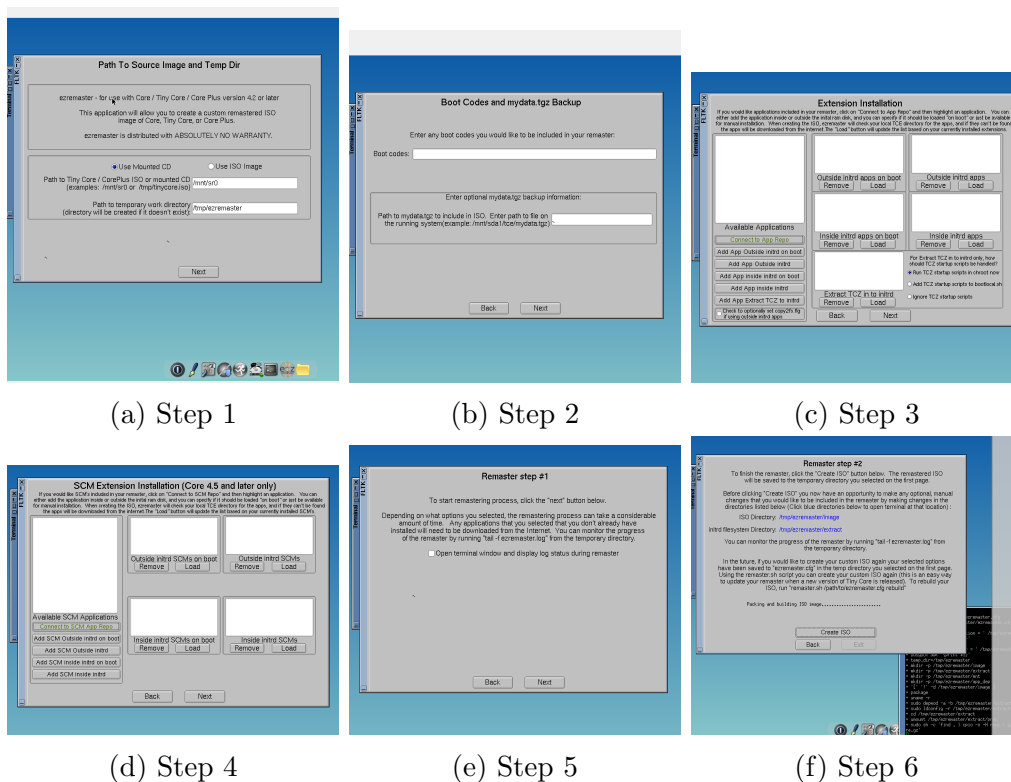


Figure 24: EZRemaster Steps



2. Launch **ezremaster**:

```
ezremaster
```

3. Configure Remastering:

- Source ISO: Select the original Tiny Core ISO.
- Custom Files: Add custom files or scripts as needed.
- Display Settings: On the third screen, specify display settings and include a path to a custom wallpaper image.

**ezremaster** will incorporate this image and set it as the default wallpaper.

4. Build ISO: Follow the prompts to create the remastered ISO.

## 10 Conclusion

In conclusion, this case study explored Tiny Core Linux (TCL), highlighting its minimalistic and efficient design, modular architecture, and various customization options.

We examined the installation process, which offers multiple methods to suit different user needs, from cloud mode to USB installation. We also discussed package management through both GUI and CLI methods, showcasing TCL's flexibility in managing software applications.

Customization and modifications were key aspects of this study, with detailed instructions on adding custom applications, setting up shared folders, creating keyboard shortcuts, changing icons and wallpapers, and more. These enhancements significantly improve usability and provide a personalized user experience.

Tiny Core Linux proves to be an excellent choice for users who require a lightweight, customizable operating system that performs well on resource-constrained devices. While it may present challenges due to its minimalistic approach and limited official documentation, users with a solid understanding of Linux systems can leverage its advantages for a variety of applications.