



UNIVERSITY INSTITUTE *of*
COMPUTING
Asia's Fastest Growing University

NAAC
GRADE **A+**
ACCREDITED UNIVERSITY

REPORT OF THE PROJECT

Digital Clock in Linux using Shell Script

Submitted by

Ankita Kumari(24MCA20142)

In partial fulfilment for the award of the degree of

MASTER OF COMPUTER APPLICATION

IN

University Institute of Computing (UIC)



CHANDIGARH UNIVERSITY

SESSION 2024 -26

1. Introduction

The goal of this project is to develop a lightweight digital clock application that runs directly within the Linux terminal, continuously displaying the current system time. This project is an excellent introduction to shell scripting and the use of Linux command-line tools. It provides practical learning in real-time data display and helps beginners get familiar with shell script automation and formatting commands in Linux. The digital clock is relevant as a fundamental tool for system monitoring, and it illustrates how basic scripting can be used to create simple utilities that enhance user productivity and system interaction.

2. Technologies Used

The main technology for this project is the **Bash shell script**, which is widely used due to its flexibility and integration with Linux systems. The project makes use of the following commands:

- **date:** This command is used to display and format date and time. It supports custom formatting options to tailor the output as needed.
- **watch:** This command runs a given program periodically and shows the output in a loop, refreshing at specified intervals.
- **clear:** Clears the terminal screen for a clean display before starting the clock.

3. Project Requirements

The project is simple and requires only a basic Linux environment, making it highly accessible. The requirements include:

- **System Requirements:**
 - Any distribution of a Linux-based operating system (e.g., Ubuntu, Debian, Fedora, CentOS).
- **Software Requirements:**
 - A shell environment such as Bash, which is typically pre-installed on most Linux distributions.

4. Design and Implementation

The digital clock's design revolves around using shell scripting and basic Linux commands to create a dynamic display in the terminal. The date command is used to format and show the current system time in HH:MM:SS format. The watch command ensures the date command is executed every second to update the display continuously.

Code Structure

The shell script for this project is simple, yet effective in achieving its purpose. Below is the code, along with a detailed breakdown:

```
date
```

```
date +%T
```

```
date +%T; sleep 1s;
```

```
date +T; sleep 1s; date +%T; sleep 1s
```

```
clear; date +%T; sleep 1s; clear; date +%T; sleep 1s
```

```
vi digi.sh
```

```
chmod 777 digi.sh
```

```
./digi.sh
```

```
vi digi.sh
```

```
./digi.sh
```

Explanation:

Here's an explanation for each step and command:

· date:

This command displays the current date and time in the default format. Running this alone in the terminal will output something like:

· date +%T:

- This variation of the date command formats the output to show only the current time in HH:MM:SS format. For example:

· date +%T; sleep 1s;:

- This command first displays the current time in HH:MM:SS format, pauses the execution for 1 second (sleep 1s), and then completes.
- The sleep command is used to delay the next command execution, allowing for a short pause.

· date +%T; sleep 1s; date +%T; sleep 1s:

- This series of commands displays the current time, waits 1 second, displays the updated time again, waits another second, and repeats this process.

· clear; date +%T; sleep 1s; clear; date +%T; sleep 1s:

- This sequence clears the terminal (clear), displays the current time, waits for 1 second, clears the terminal again, and repeats the process. The clear command ensures that only the updated time is shown on the screen each second, simulating a real-time ticking clock.

• **vi digi.sh:**

- This command opens the vi editor to create or edit a shell script named digi.sh. You can enter the code for the digital clock inside this file.

• **chmod 777 digi.sh:**

- This command changes the permissions of digi.sh to make it fully executable and accessible by the user, group, and others. The permission 777 means:
 - 7 (owner): read, write, and execute.
 - 7 (group): read, write, and execute.
 - 7 (others): read, write, and execute.

• **./digi.sh:**

- This runs the digi.sh script in the current directory. The ./ indicates that the script should be executed from the current directory.

• **vi digi.sh (again):**

- If you open the file again with vi digi.sh, you are likely modifying or reviewing the script.

Explanation of ./digi.sh:

- This command runs the script digi.sh you created earlier. If your script includes code like

while true

do

clear

date +%T

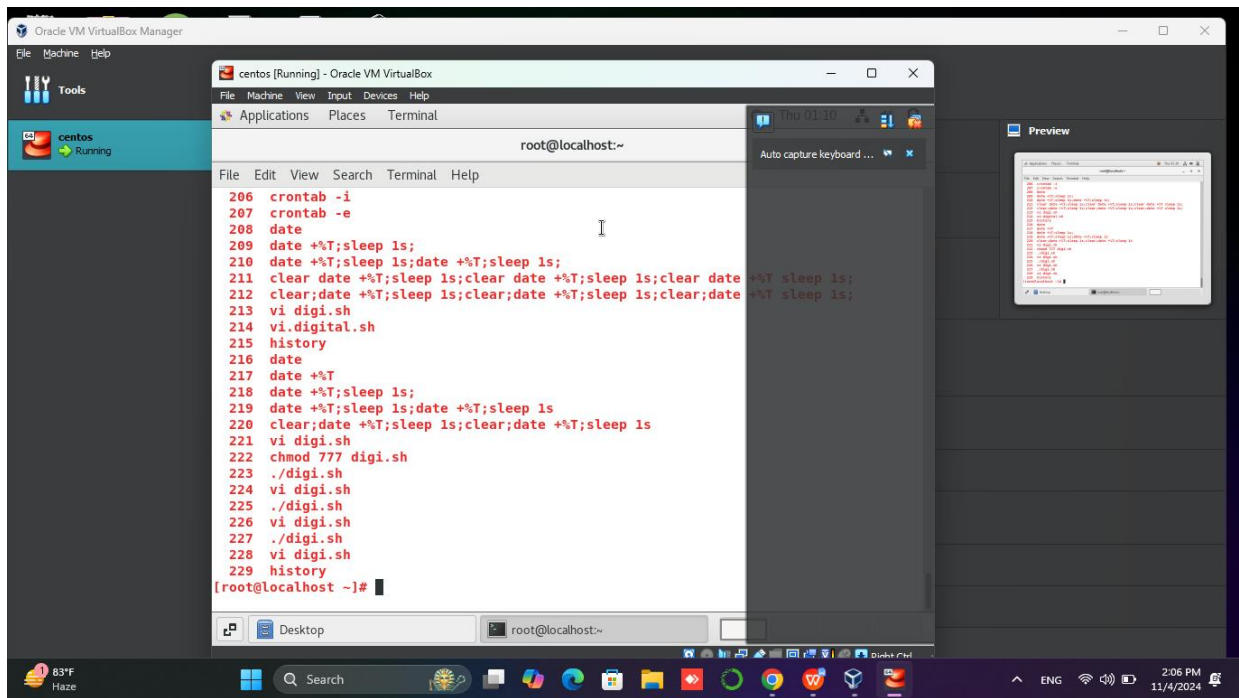
sleep 1

done

It would display the current time in HH:MM:SS format, clear the screen, and refresh the time every second indefinitely until you stop the script (using Ctrl+C).

5. Execution Steps

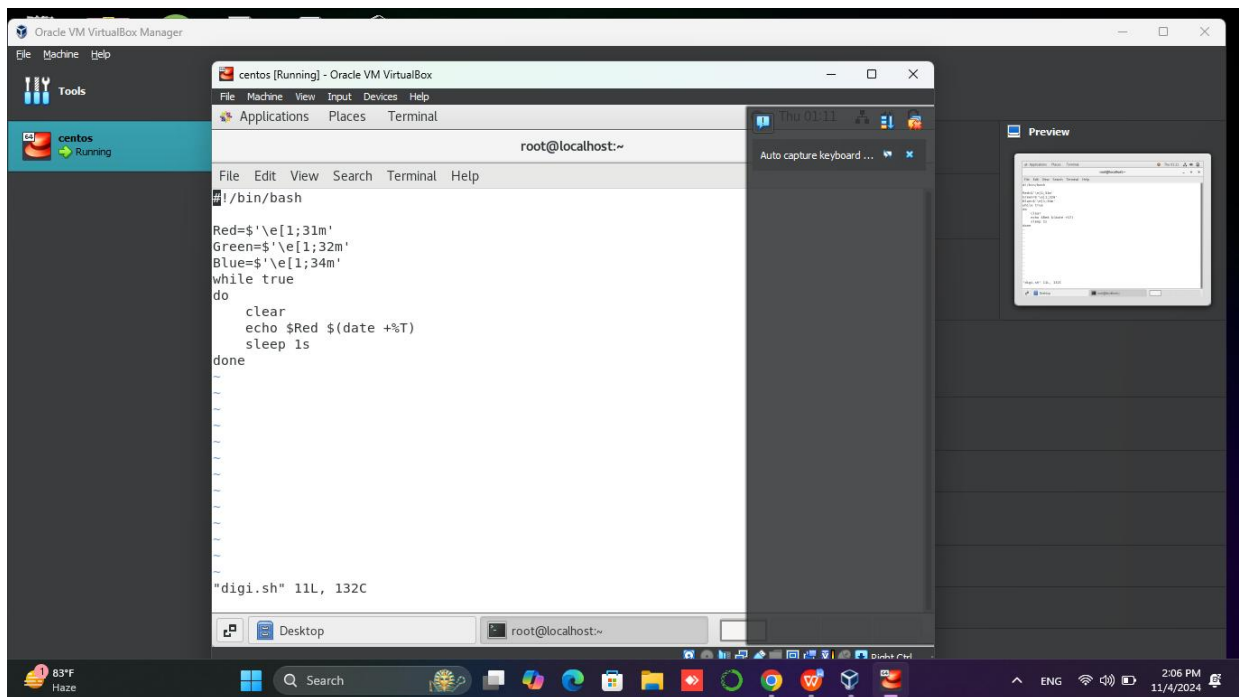
Follow these steps to create and execute the digital clock script:



```

root@localhost:~# crontab -i
root@localhost:~# crontab -e
root@localhost:~# date
root@localhost:~# date +%T;sleep 1s;
root@localhost:~# date +%T;sleep 1s;date +%T;sleep 1s;
root@localhost:~# clear;date +%T;sleep 1s;clear;date +%T;sleep 1s;clear;date +%T;sleep 1s;
root@localhost:~# vi digi.sh
root@localhost:~# vi .digital.sh
root@localhost:~# history
root@localhost:~# date
root@localhost:~# date +%T
root@localhost:~# date +%T;sleep 1s;
root@localhost:~# date +%T;sleep 1s;date +%T;sleep 1s
root@localhost:~# clear;date +%T;sleep 1s;clear;date +%T;sleep 1s
root@localhost:~# vi digi.sh
root@localhost:~# chmod 777 digi.sh
root@localhost:~# ./digi.sh
root@localhost:~# vi digi.sh
root@localhost:~# ./digi.sh
root@localhost:~# vi digi.sh
root@localhost:~# ./digi.sh
root@localhost:~# vi digi.sh
root@localhost:~# ./digi.sh
root@localhost:~# vi digi.sh
root@localhost:~# history
root@localhost:~#

```



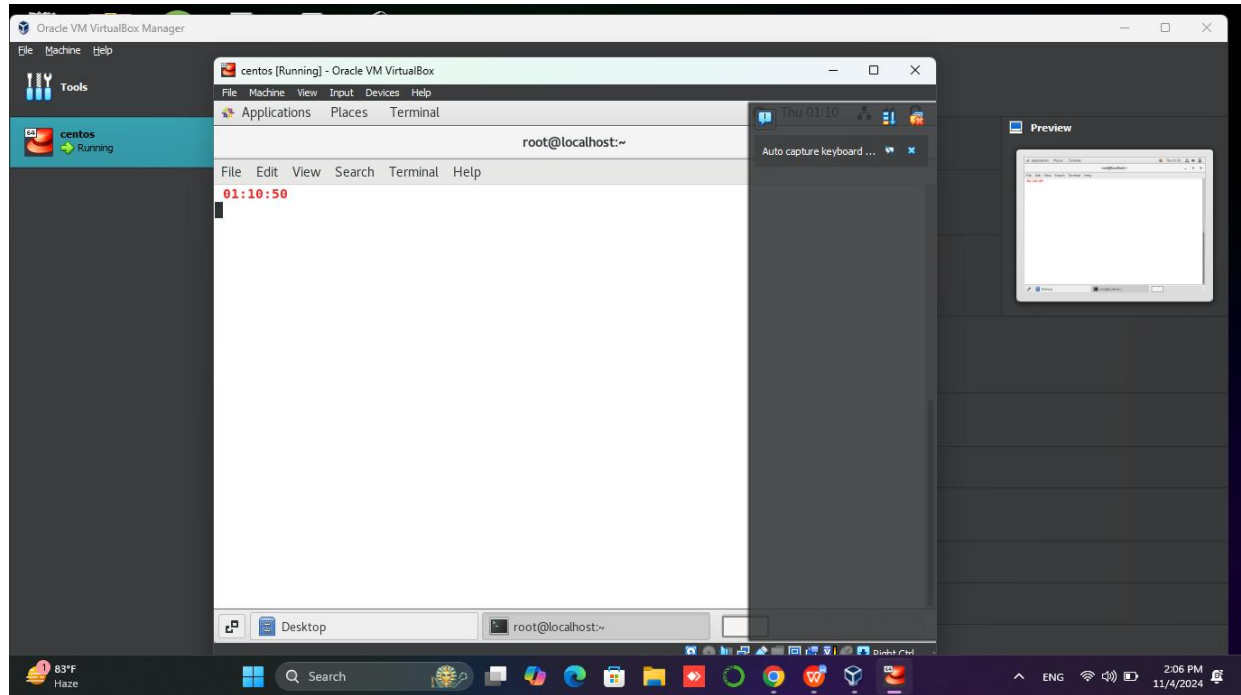
```

root@localhost:~# ./bin/bash
root@localhost:~# Red='\e[1;31m'
root@localhost:~# Green='\e[1;32m'
root@localhost:~# Blue='\e[1;34m'
root@localhost:~# while true
root@localhost:~# do
root@localhost:~#     clear
root@localhost:~#     echo $Red $(date +%T)
root@localhost:~#     sleep 1s
root@localhost:~# done
root@localhost:~#
root@localhost:~# "digi.sh" 11L, 132C

```

6. Output

The output of the script is a continuously updating display of the current system time in HH:MM:SS format. The watch command ensures that the output refreshes every second, creating a live digital clock effect in the terminal. The screen clears and shows only the clock, providing a minimalistic and focused interface.



7. Challenges and Solutions

Screen Refresh: One potential challenge when displaying real-time information in the terminal is ensuring that the screen updates smoothly without flickering. The watch command handles this efficiently by clearing and updating the screen for each cycle, ensuring a seamless viewing experience.

Customization: Another challenge is the need for customization. The date command supports numerous format specifiers that can be added to display additional information such as the date, day of the week, or time zone. This flexibility allows users to modify the script to fit their specific needs. For example, displaying both date and time can be done by changing the command to:

```
bash
Copy code
watch -n 1 "date +%Y-%m-%d %H:%M:%S"
```

8 Conclusion

The completion of this project has resulted in a fully functional and efficient digital clock that operates within the Linux terminal, displaying real-time updates. This project exemplifies how powerful and practical shell scripting can be for creating straightforward yet effective utilities. By using a concise shell script, we have demonstrated the power of built-in Linux commands to automate tasks and present dynamic information seamlessly. This project is especially useful for beginners as it bridges the gap between basic command-line usage and the development of simple automated tools. Moreover, it showcases how easily shell scripts can be tailored and adapted to a wide range of practical purposes without requiring additional software installations.

The project's implementation is straightforward and provides an excellent introduction to the fundamental concepts of Bash scripting, command-line commands, and real-time data processing. Its ease of understanding and implementation makes it a great learning tool for anyone new to shell scripting or looking to expand their command-line expertise.

Future Enhancements There are multiple directions in which this digital clock project could be enhanced and expanded. These improvements would make the tool more versatile, user-friendly, and aesthetically appealing. Here are some ideas for further development:

Adding Colors:

1. Implement the `tput` command to introduce color to the text display, which would improve the visual appeal and user experience. For example, using `tput setaf <color-code>` can change the text color dynamically in the script.

Display Customization:

1. Enhance the script by adding options to display time in different formats, including support for various time zones. This would make the tool more useful for users who work across multiple regions. Additionally, including AM/PM markers would allow users to toggle between 12-hour and 24-hour time formats.

Additional Features:

1. Extend the functionality of the script to include countdown timers, alarms, and stopwatch capabilities. This would convert the digital clock into a comprehensive time-management utility. Users could customize timers and alarms with personalized notifications or sound alerts using Linux's built-in `beep` command or audio scripts.

Date Display:

1. Modify the script to show both the date and time simultaneously. This can be done by updating the date command with a format string that includes the date, such as `date +%Y-%m-%d %H:%M:%S`. This feature would make the clock more informative and better suited for daily use.

Enhanced User Interface:

1. Consider incorporating ASCII art or symbols to create an attractive clock border or separator. This could make the script visually distinctive and fun to use.

User Input for Customization:

1. Include user input prompts or options that let users customize how the clock displays. For instance, users could specify the refresh rate, choose different date and time formats, or toggle between showing and hiding the date.

Script Optimization:

1. Optimize the script to handle more complex tasks efficiently, such as displaying multiple clocks with different time zones on the same terminal window or providing a multi-functional interface where users can switch between clock modes (digital clock, timer, alarm).

Integration with System Services:

1. Integrate the script as a background service that runs on startup and displays the clock at the user's request. This could be achieved by adding the script to the system's startup services using systemd or cron jobs.

9. References

1. Linux Man Pages for date Command:

1. Detailed documentation on the usage, options, and formatting of the date command to display and customize date and time output.

2. Linux Man Pages for watch Command:

1. Comprehensive guidance on the watch command, including options for running commands repeatedly at specified intervals.

3. Bash Scripting Tutorials:

1. Online tutorials provided foundational insights into basic and advanced shell scripting techniques.

4. Community Discussions:

1. Forums and Linux user groups that shared examples and experiences in building lightweight command-line tools.

5. Linux Documentation:

1. Official and user-contributed documentation that helped understand shell scripting practices and methods to automate tasks.