PROJECT REPORT ON

"FILE BACKUP SCRIPT"

Submitted By:

Ankur Bibhuti, UID- 24MCA20354

Under The Guidance of:

MS. Geetanjali Sharma

April, 2024

University Institute of

ComputingChandigarh

University,

Mohali, Punjab

CERTIFICATE

This is to certify that Ankur Bibhuti (UID- 24MCA20354) have successfully completed the project title **"File Backup Script"** at University Institute of Computing under my supervision and guidance in the fulfilment of requirements of fourth semester, **Master of Computer Application** Of Chandigarh University, Mohali, Punjab.

Dr. Abdullah
Head of the Department
University Institute of Computing

Ms Geetanjali Sharma
Project Guide Supervisor
University Institute of Computing

ACKNOWLEDGEMENT

We deem it a pleasure to acknowledge our sense of gratitude to our project guide Ms. Geetanjali Sharma under whom we have carried out the project work. Her incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish to reciprocate in full measure the kindness shown by Dr. Abdullah (H.O.D, University Institute of Computing) who inspired us with his valuable suggestions in successfully completing the project work.

We shall remain grateful to Dr. Manisha Malhotra, Additional Director, University Institute of Technology, for providing us a strong academic atmosphere by enforcing strict discipline to do the project work with utmost concentration and dedication.

Finally, we must say that no height is ever achieved without some sacrifices made at some end and it is here where we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period of time.

Date: 25.10.2024

Place: Chandigarh University, Mohali, Punjab

Ankur Bibhuti, UID- 24MCA20354

ABSTRACT

In an era where digital data has become essential to the operations of individuals, businesses, and organisations, safeguarding this data against loss is a critical requirement. Data loss can occur for various reasons, including accidental deletion, system failures, hardware malfunctions, or even cyberattacks. Regular file backups are one of the most effective strategies for mitigating such risks. This project focuses on the development of an automated file backup script, designed to simplify, automate, and streamline the process of creating backups on Linux-based systems.

The primary objective of the file backup script is to provide a reliable and user-friendly method for creating backups of important files and directories, ensuring that valuable data is protected and can be easily restored in case of any unforeseen events. The script is highly customisable and provides various backup options, including full backups, which copy all the files from the source directory, and incremental backups, which only copy files that have changed or been added since the last backup. This incremental backup feature significantly reduces the time and storage required for subsequent backups, making it a more efficient solution for users with large datasets.

One of the standout features of the script is versioning, which automatically appends a timestamp to each backup, allowing users to maintain multiple versions of backups. This feature is particularly useful if they need to revert to a previous backup version at a later date. The timestamped backups ensure that older copies of data are retained, reducing the risk of overwriting valuable files.

To further optimise storage, the script supports compression, using the `.tar.gz` format, which compresses the backup data into a smaller file size. This is especially beneficial for users with limited disk space or those backing up large directories, as it minimises storage requirements while ensuring that backup data remains easily accessible and manageable.

The backup process is designed to be fully automated, and users can schedule the script to run at predefined intervals using cron jobs, a built-in Linux utility that automates tasks based on a time schedule. This means that backups can be executed at regular intervals (e.g., daily, weekly, or monthly) without requiring manual intervention, ensuring that the data is always up to date without any extra effort from the user.

Additionally, the script includes robust error handling mechanisms that ensure the backup process runs smoothly, even in the event of issues such as invalid file paths, permission problems, or system failures. If any errors occur, the script logs them and provides useful error messages to the user. This logging functionality is essential for troubleshooting, as it gives the user a clear understanding of what went wrong and helps in quickly resolving any problems.

INTRODUCTION

In today's digital landscape, data is not only an essential asset for individuals, but also for businesses and organisations of all sizes. From personal photos, documents, and emails to business-critical data, such as financial records, databases, and system configurations, data forms the foundation of our personal and professional lives. However, data is inherently fragile. Accidental deletion, system failures, corruption, malware, cyberattacks, and natural disasters are all risks that can lead to the permanent loss of important information. To mitigate these risks, it is crucial to maintain regular backups of data, ensuring that copies are safely stored and can be restored in the event of a disaster.

A data backup is essentially a duplicate copy of critical files, stored separately from the original data. In case the original data becomes compromised, users can restore it from a backup. While backing up data is a good practice, doing so manually can be time-consuming, error-prone, and inefficient—especially when dealing with large amounts of data. Therefore, automating the backup process is one of the most effective ways to streamline and ensure the reliability of data protection efforts.

This project aims to address this need by developing an automated file backup script for Linux systems. The script is designed to automate the process of backing up files or directories from a source location to a destination location, offering flexibility and customisation to meet diverse backup requirements. The primary goal is to make it easier for users to secure their data, ensuring regular backups without requiring continuous manual intervention.

OBJECTIVE

The primary objective of this project is to develop an automated file backup script for Linux-based systems that simplifies the process of backing up files and directories. The script aims to improve the reliability, efficiency, and convenience of data backup by automating critical tasks and offering customisable options for users. Below are the specific objectives that the project aims to achieve:

1. Automate the Backup Process

- **Objective**: Automate the entire backup process to eliminate the need for manual intervention.
- **Details**: The script will allow users to create backups at scheduled intervals without needing to run the backup task manually each time. This will ensure that backups are conducted consistently and reliably, reducing the chances of data loss due to missed backups.

2. Support Full and Incremental Backups

- **Objective**: Provide flexibility in backup strategies by supporting both full and incremental backups.
- **Details**:
  - **Full Backup**: The script will create a complete backup by copying all files from the source directory to the destination, ensuring that the entire set of files is backed up.
  - **Incremental Backup**: The script will only back up files that have changed since the last backup. This feature will be especially useful for users with large amounts of data, as it will minimise the time and storage required for subsequent backups.
  - The script will intelligently decide whether to perform a full or incremental backup based on the user's choice and the backup history.

3. Provide Version Control for Backups

- **Objective**: Maintain multiple versions of backups using timestamp-based versioning.
- **Details**: The script will automatically append a timestamp to each backup, allowing users to retain multiple versions of backups over time. This will enable users to restore a previous version of their data in case they need to recover files from a specific point in the past.

4. Implement Compression to Save Storage Space

- **Objective**: Reduce the storage space required for backups by providing an option to **compress backup files**.
- **Details**: The script will use common compression formats (e.g., `.tar.gz` or `.zip`) to compress backup data. This will make it easier to store large backups while minimising the amount of disk space consumed by backup files. Compression will also help with data transfer if backups need to be moved across the network.

TOOLS AND TECHNOLOGIES

The development of the **Automated File Backup Script** for Linux-based systems leverages a variety of **tools** and **technologies** that provide the necessary functionality for efficient, reliable, and flexible backup operations. Below is a detailed overview of the tools and technologies used in this project:

1. Linux Shell Scripting (Bash)

- **Role**: The backbone of the file backup script is the **Bash shell**, which is the default shell on most Linux distributions. Bash scripting allows for the automation of various tasks through command-line instructions, making it ideal for system administration tasks like file backups.
- **Why it is Used**:
  - **Automation**: Bash scripting enables automation of repetitive tasks, reducing the need for user intervention.
  - **Simplicity**: Bash is lightweight, highly efficient, and provides easy integration with other Linux utilities.
  - **System Integration**: Linux shell scripts are well-suited for interacting with the file system, performing file manipulations (copy, move, compress), and scheduling tasks via `cron`.

2. `rsync`

- **Role**: `rsync` is a powerful file synchronisation and transfer tool in Linux, often used for efficient backups.
- **Why It is Used**:
  - **Incremental Backups**: `rsync` is perfect for performing **incremental backups**. It only copies files that have changed since the last backup, saving time and storage.
  - **Reliability**: `rsync` is known for its efficiency in copying large files and directories over local or remote connections, while ensuring data integrity.
  - **Options**: `rsync` supports various options for copying files, excluding certain files or directories, preserving file permissions, and performing checksums for verification.

3. `tar` (Tape Archive)

- **Role**: `tar` is a widely used Linux utility for archiving files, and it also supports compression.

- **Why it is Used**:
  - **Compression**: The `tar` command can combine multiple files and directories into a single compressed archive (`.tar.gz`, `.tar.bz2`, etc.), reducing the storage required for backups.
  - **Backup Packaging**: It allows the backup to be packaged into a single file, making it easier to transfer or store.
  - **Versatility**: The `tar` command can work with local files as well as remote directories over SSH.

4. `cron` (Cron Jobs)

- **Role**: `cron` is the time-based job scheduler in Unix-like operating systems, used to schedule tasks to run at specific intervals.
- **Why it is Used**:
  - **Scheduling Automatic Backups**: `cron` allows for the **automatic execution** of the backup script at predefined times (e.g., daily, weekly, monthly).
  - **Reliability**: It ensures that backups are performed consistently, even when the user is not actively managing the system.
  - **Flexibility**: Cron jobs can be customised to run the backup script at various times, ensuring that the data is backed up regularly and reliably.

5. `gzip` / `bzip2` / `xz` (Compression Tools)

- **Role**: Compression utilities like `gzip`, `bzip2`, and `xz` are used to reduce the size of backup files, making them easier to store and transfer.
- **Why it is Used**:
  - **Storage Efficiency**: These tools reduce the size of backup archives, saving storage space, particularly when backing up large amounts of data.
  - **Different Compression Levels**: `gzip` offers faster compression, while `bzip2` and `xz` provide higher compression ratios, which may be preferable for extremely large backups.

COMMAND/STEPS

To accomplish the **File Backup Script** project in Redhat Linux, we will need to follow thesesteps:

1. **Create a Bash Script for Backups:** We'll use rsync and tar for the backups. rsync is great for syncing files between directories, and tar can be used to compress the backup

2. **Use Cron for Scheduling:** Cron is a job scheduling system that allows you to run scriptsat regular intervals.

- **Step 1:** Create the Backup Script:

  1. Open a terminal and create a new Bash script file:

     " nano backup_script.sh "

  2. Add the following script to handle the file backup. This script uses rsync to sync filesfrom the source directory to the backup directory and tar to compress the backup:

```
# Source and destination directories
SOURCE_DIR="/home/ankurbibhuti/Documents
"BACKUP_DIR="/home/ankurbibhuti/
Downloads "

# Get current date for backup naming
DATE=$(date +"%Y-%m-%d")

# Rsync command to sync files
rsync -av --delete $SOURCE_DIR/ $BACKUP_DIR/

# Compress the backup directory using tar
tar -czvf $BACKUP_DIR/backup_$DATE.tar.gz -C
$BACKUP_DIR
```

echo " Backup completed for $DATE
"

3. Save and close the script (Ctrl + X, then Y, then Enter).

4. Make the script executable: -

“ chmod +x backup_script.sh ”

- **Step 2**: Test the Script: “ ./backup_script.sh ”

If everything works, you should see a message saying "Backup completed" and a .tar.gz file inthe backup directory.

- **Step 3:** Schedule the Script Using Cron

1. Open the crontab file for the current user:

“ crontab -e ”

2. Add a cron job to run the script at a specific time. For example, to run the backup everyday at 2 AM:

“ 0 2 * * * /path/to/backup_script.sh “

- **Step 4:** Verify Cron Job:

“ crontab -l “

LEARNING OUTCOMES

By completing this project on creating an Automated File Backup Script for Linux systems, several important learning outcomes were achieved, which can be applied to real-world scenarios in system administration, data management, and software development. These outcomes focus on practical skills, understanding core concepts, and gaining hands-on experience with various Linux tools and technologies.

1. Understanding and Mastering Bash Scripting

- **Outcome**: One of the key learning outcomes is the ability to write effective Bash shell scripts to automate repetitive tasks in Linux environments.
- **Details**: By developing the backup script, I learned how to utilise various Bash constructs such as conditionals, loops, functions, and error handling. This has enhanced my ability to automate system administration tasks in Linux environments.

2. Proficiency with Backup Tools (rsync, tar, gzip, etc.)

- **Outcome**: I gained hands-on experience with important Linux backup tools such as `rsync`, `tar`, `gzip`, and `bzip2`, each of which serves a distinct role in the backup process.
- **Details**: I learned the strengths and limitations of these tools in terms of their use in incremental backups, file compression, and archiving. Understanding when to use each tool based on the requirements (e.g., compression vs. speed vs. network efficiency) is a crucial skill in real-world data management scenarios.

3. Practical Knowledge of Backup Strategies

- **Outcome**: I developed a solid understanding of various backup strategies (full, incremental, differential) and the trade-offs involved in each.
- **Details**: Through the development of the script, I learned how to balance the need for complete backups with the efficiency of incremental backups, and how to maintain different versions of backups over time. This knowledge is essential for any IT professional involved in data management and protection.

4. Mastering Scheduling and Automation with Cron

- **Outcome**: By integrating the backup script with cron jobs, I learned how to schedule regular tasks to run at specified times, thus automating the backup process.
- **Details**: This skill is highly relevant for system administrators who need to ensure routine tasks (like backups, system updates, or log rotations) are performed consistently without

manual intervention. I learned how to set up cron jobs to automate these processes on a Linux system.

5. Implementing Error Handling and Logging

- **Outcome**: I learned how to implement error handling and logging in scripts, which are crucial for diagnosing issues, ensuring that the backup process runs smoothly, and providing accountability for system operations.
- **Details**: This experience taught me how to use Linux tools like `echo`, `logger`, and custom log files to capture backup status, errors, and other relevant information. Logging is a fundamental aspect of managing automated processes in production environments.

CONCLUSION

The Automated File Backup Script project has been a highly educational and practical exercise in understanding the importance of data protection, automation, and system administration within a Linux-based environment. By developing and implementing this script, I not only gained a deeper understanding of core Linux tools and utilities but also acquired the skills necessary to create a functional, scalable, and reliable backup solution. This project also highlighted the significance of automating essential tasks, such as regular backups, and demonstrated how system administrators can streamline processes to ensure data integrity and minimise human error.

The primary goal of this project was to create an automated solution that could efficiently and reliably back up important files, with minimal user intervention. By using a combination of tools like **rsync**, **tar**, **gzip**, and **cron**, the script is able to meet this goal by providing an incremental backup solution that is both resource-efficient and simple to deploy. It achieves a good balance between performance, storage usage, and automation —ensuring backups run on a schedule, without requiring constant monitoring. This not only saves time but also ensures data is protected regularly, which is crucial in an increasingly data-driven world.

REFERENCE

The Linux Documentation Project.
Bash Guide for Beginners.
URL: https://tldp.org/LDP/Bash-Beginners-Guide/html/

- This guide provided foundational knowledge on Bash scripting, including how to write and structure shell scripts, as well as handling variables, loops, and functions within the script.

GNU `rsync` Manual.
rsync(1) — a fast, versatile, remote (and local) file-copying tool.
URL: https://www.gnu.org/software/rsync/manual/

- The official documentation for `rsync`, a critical tool used in the script for efficient file synchronisation, incremental backups, and remote copying.

GNU `tar` Manual.
tar — Archiving utility.
URL: https://www.gnu.org/software/tar/manual/

- This manual provided an in-depth understanding of how to use `tar` for creating compressed backup archives, including common options for file exclusion and compression.

GNU `gzip` Manual.
gzip — A utility for file compression.
URL: https://www.gnu.org/software/gzip/manual/

- Information on using `gzip` for compressing backup archives, with tips on optimizing compression for faster transfer speeds and lower storage requirements.

The Linux Command Line.
William E. Shotts, Jr.
No Starch Press, 2012.

- A comprehensive resource for learning Linux command-line usage, Bash scripting, and advanced topics in Linux system administration.