Placement Empowerment Program

Cloud Computing and DevOps Centre

Day 07 – Directory Backup Script with Timestamp

Write a script that compresses a directory into a **.tar.gz** archive with a timestamp in the filename.

Name: Hema S

Department: ECE

Introduction

In Linux systems, regularly backing up important directories is a critical task for system reliability, disaster recovery, and data safety. Manual backups, however, are time-consuming and error-prone. In this Proof of Concept (POC), I developed a bash script that automates the process of compressing any directory into a timestamped **.tar.gz** archive.

This ensures:

✅ Easy identification of backup versions
✅ Consistent backup naming format
✅ Automation-ready behavior using cron scheduler

This POC combines core Linux concepts such as **shell scripting, date formatting, file compression using tar, and task scheduling —** all essential skills for system administration and DevOps workflows.

Overview

This POC focuses on creating a simple yet powerful bash script that automates the process of backing up a directory in a Linux environment. The script compresses the given directory into a **.tar.gz** archive and appends a timestamp to the filename, making each backup unique and easy to track.

This solution eliminates the need for manual naming and helps maintain a versioned history of backups. It's ideal for personal use, system maintenance, and DevOps tasks where regular backups are essential.
To further enhance automation, the script is scheduled using cron, ensuring that backups are performed at a fixed time daily without user intervention.

Key steps in this PoC:

✅ **Script Initialization:**
Created a bash script named backup.sh using a text editor like nano.

✅ **Input Handling:**
Configured the script to accept the directory name as a command-line argument.

✅ **Validation:**
Checked if the directory exists. If not, the script exits with an error message.

✅ **Timestamp Generation:**
Used the date command to generate a timestamp in the format YYYYMMDD_HHMMSS.

✅ **Backup Naming:**
Combined the directory name and timestamp to create a unique backup filename.

✅ **Compression:**
Used tar -czf to compress the directory into a **.tar.gz** archive.

✅ **Backup Verification:**
Tested the backup by listing and extracting the archive using **tar -xzf.**

✅ **Automation with Cron:**
Scheduled the script to run daily at a specific time using **crontab -e.**

Objectives :

✅ Automate directory backups using a simple and reusable bash script.
✅ Generate timestamped backup files to keep track of versions and changes over time.
✅ Ensure data safety and recovery by compressing important directories regularly.
✅ Utilize core Linux commands (tar, date, chmod, etc.) in a real-world use case.
✅ Schedule automated backups using the cron job scheduler to reduce manual effort.
✅ Enhance Linux scripting skills with practical system administration experience.

Importance :

✅**Data Protection:**
Regular backups are critical to prevent data loss due to accidental deletion, corruption, or system failure.

✅ **Version Control:**
Adding timestamps to backup filenames helps maintain a clear history of backups, making it easy to restore from a specific point in time.

✅**Automation Readiness:**
With cron integration, the script requires no manual intervention, reducing human error and saving time.

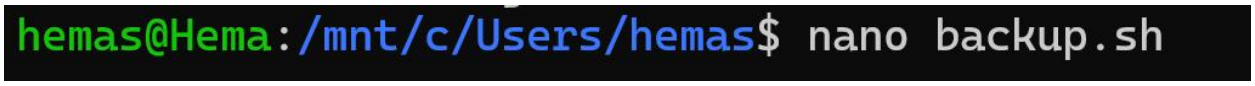
✅**Real-World Skill Building:**
This POC demonstrates practical knowledge of bash scripting, file compression, and Linux system operations — key skills for DevOps and sysadmins.

✅ **Professional Use Case:**
Such automated backup mechanisms are commonly used in production systems, making this POC valuable for IT job roles or project implementation.

Step-by-Step Overview

Step 1: Create the Script File

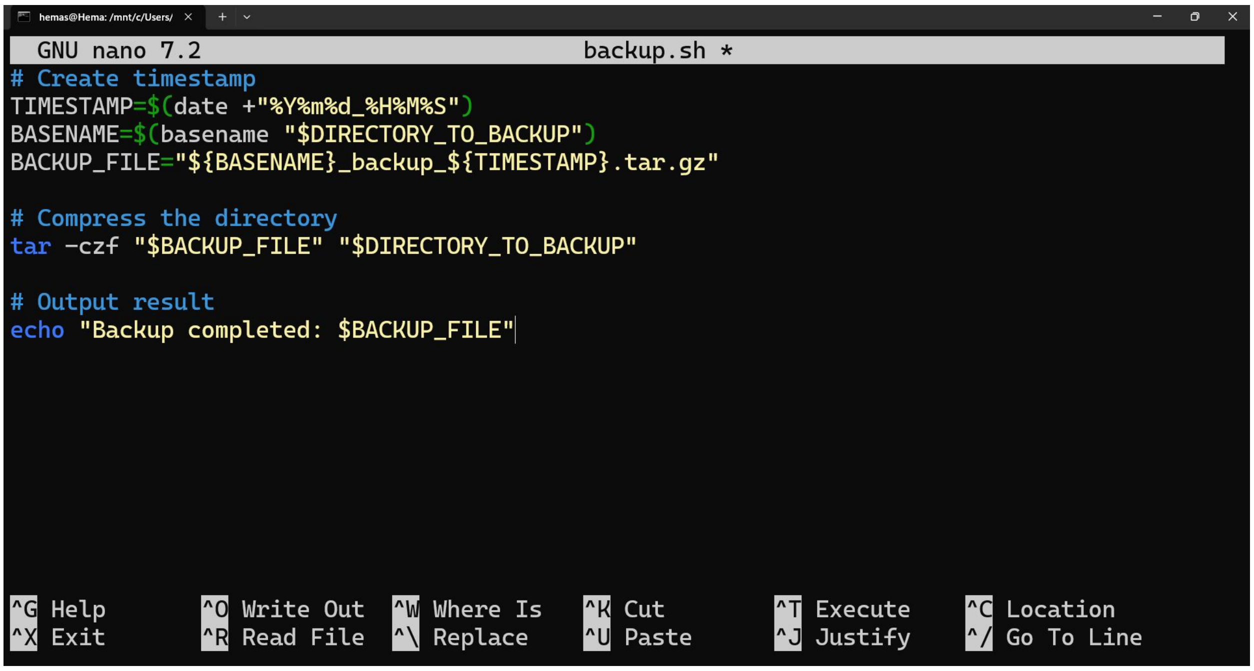
Use a terminal-based editor like nano to create the script:



```
hemas@Hema:/mnt/c/Users/hemas$ nano backup.sh
```

Step 2: Write the Backup Script

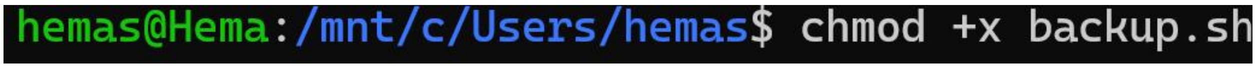
Paste the following bash code:



```
hemas@Hema: /mnt/c/Users/
  GNU nano 7.2                        backup.sh *
# Create timestamp
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")
BASENAME=$(basename "$DIRECTORY_TO_BACKUP")
BACKUP_FILE="${BASENAME}_backup_${TIMESTAMP}.tar.gz"

# Compress the directory
tar -czf "$BACKUP_FILE" "$DIRECTORY_TO_BACKUP"

# Output result
echo "Backup completed: $BACKUP_FILE"









^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line
```

Step 3: Make Script Executable

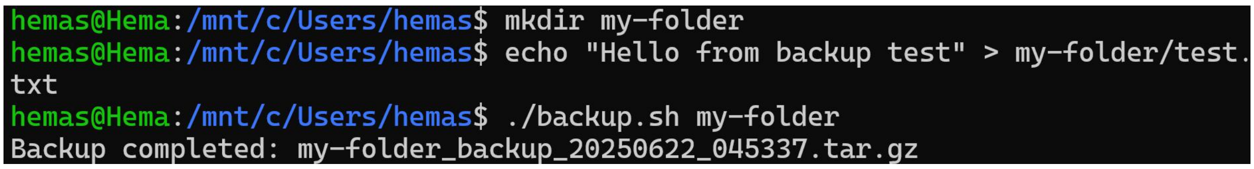
Give the script permission to run:



```
hemas@Hema:/mnt/c/Users/hemas$ chmod +x backup.sh
```

Step 4: Test with a Sample Folder

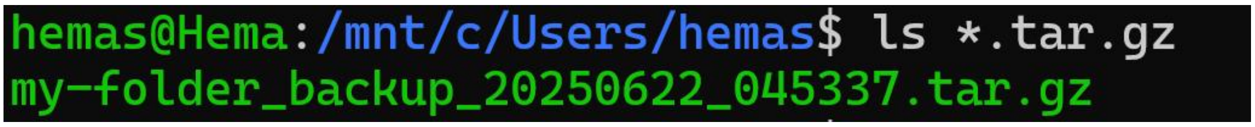
Create a folder and a test file:



```
hemas@Hema:/mnt/c/Users/hemas$ mkdir my-folder
hemas@Hema:/mnt/c/Users/hemas$ echo "Hello from backup test" > my-folder/test.
txt
hemas@Hema:/mnt/c/Users/hemas$ ./backup.sh my-folder
Backup completed: my-folder_backup_20250622_045337.tar.gz
```

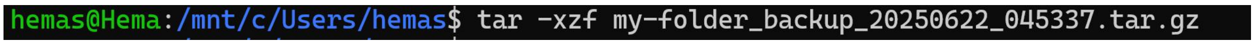
Step 5: Verify the Backup

List the archive:



```
hemas@Hema:/mnt/c/Users/hemas$ ls *.tar.gz
my-folder_backup_20250622_045337.tar.gz
```

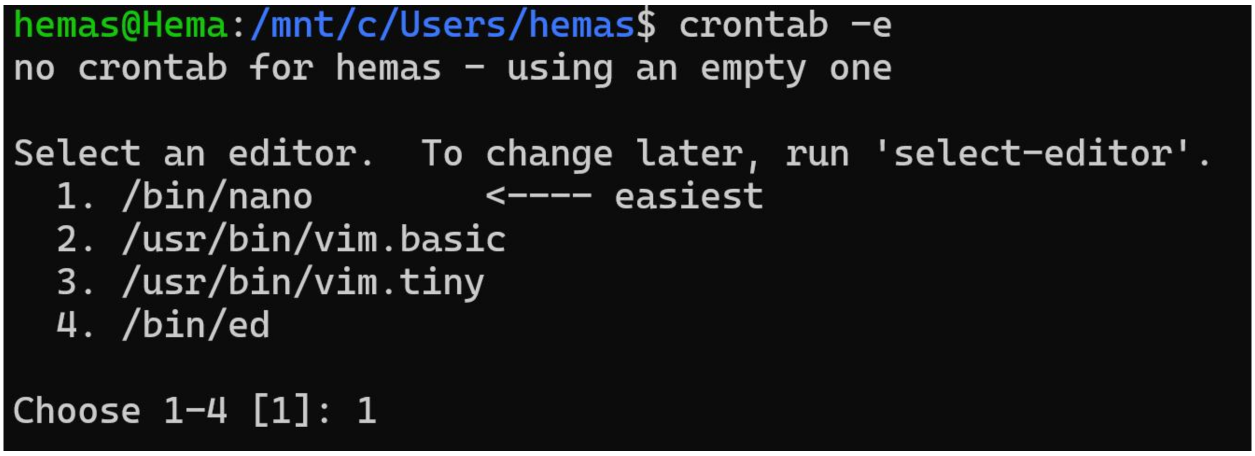
Extract it to test:



```
hemas@Hema:/mnt/c/Users/hemas$ tar -xzf my-folder_backup_20250622_045337.tar.gz
```

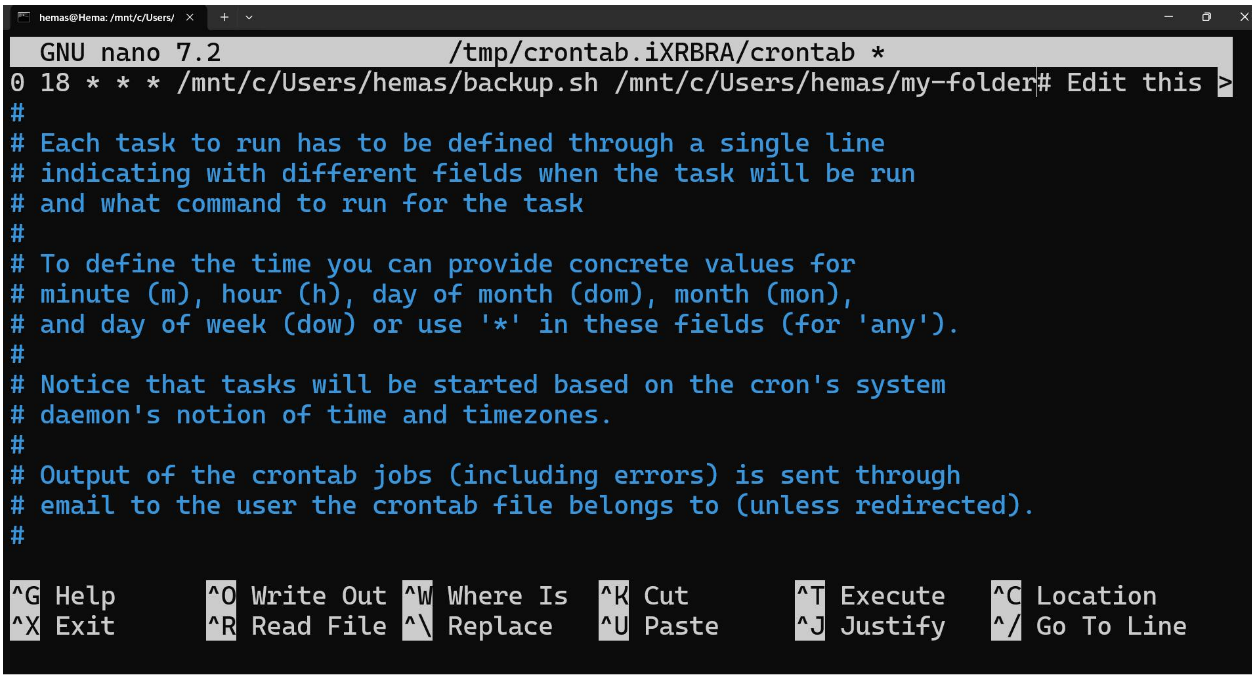
Step 6: Automate with Cron

Open your crontab:



```
hemas@Hema:/mnt/c/Users/hemas$ crontab -e
no crontab for hemas - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
  4. /bin/ed

Choose 1-4 [1]: 1
```

Add this line to schedule daily backups at 6:00 PM:



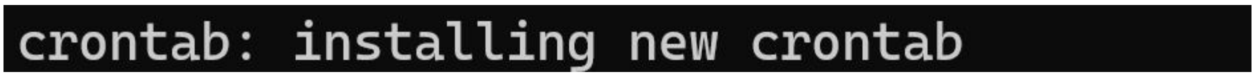
```
  GNU nano 7.2                /tmp/crontab.iXRBRA/crontab *
0 18 * * * /mnt/c/Users/hemas/backup.sh /mnt/c/Users/hemas/my-folder# Edit this >
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#

^G Help      ^O Write Out ^W Where Is ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace  ^U Paste     ^J Justify   ^/ Go To Line
```

Step 7:Save and Exit

Press Ctrl + O → Enter (to save)

Press Ctrl + X (to exit)

Step 8:Confirm It’s Scheduled



```
crontab: installing new crontab
```

Outcomes:

✅ Developed a functional bash script to back up any directory into a **.tar.gz** archive.
✅ Implemented timestamping in the filename to uniquely identify each backup version.

✅ Validated the backup process by testing archive creation and successful extraction.

✅Applied Linux fundamentals like **tar, date, basename, and chmod** in a real-world context.

✅ Scheduled daily backups using cron, demonstrating automation and hands-off operation.

✅Improved command-line proficiency and gained confidence in scripting and task automation.