Here’s the content for a `README.txt` file that explains the usage and the code:

Backup with Rotation Script

This script creates timestamped backups of a specified directory and implements a rotation mechanism to retain only the last 3 backups. Older backups are automatically deleted to save space.

Features

1. Creates a timestamped backup folder inside the specified directory.
2. Copies all files from the directory into the backup folder (excluding existing backup folders).
3. Ensures only the last 3 backups are retained by removing older ones.
4. Provides clear feedback during execution.

How to Use

Prerequisites

- Ensure you have Bash installed (most Linux systems have it by default).
- The script requires `rsync` for efficient file copying.

Steps to Use

1. Save the script as `backup_with_rotation.sh`.
2. Make it executable:

   ```
   chmod +x backup_with_rotation.sh
   ```

3. Run the script with the path to the directory you want to back up:

   ```
   ./backup_with_rotation.sh /path/to/directory
   ```

Example Usage

First Execution:

```
$ ./backup_with_rotation.sh /home/user/documents
Backup created: /home/user/documents/backup_2023-07-30_12-30-45
```

Second Execution:

```
$ ./backup_with_rotation.sh /home/user/documents
Backup created: /home/user/documents/backup_2023-08-01_09-15-30
Rotating backups...
Removing old backup: /home/user/documents/backup_2023-07-30_12-30-45
```

Explanation of Code

1. Input Validation

```
if [[ -z "$1" ]]; then
    echo "Usage: $0 <target_directory>"
    exit 1
fi

if [[ ! -d "$TARGET_DIR" ]]; then
    echo "Error: Directory '$TARGET_DIR' does not exist"
    exit 1
fi
```

- The script checks if a directory path is provided as an argument.
- If no argument is provided or if the specified directory does not exist, it exits with an error message.

2. Creating a Timestamped Backup

```
TIMESTAMP=$(date +"%Y-%m-%d_%H-%M-%S")
BACKUP_DIR="${TARGET_DIR}/backup_${TIMESTAMP}"
mkdir -p "$BACKUP_DIR" || exit 1

rsync -a --exclude='backup_*' "$TARGET_DIR/" "$BACKUP_DIR/" || {
    echo "Backup failed - removing incomplete backup"
    rm -rf "$BACKUP_DIR"
    exit 1
}
```

- A timestamp is generated using `date` to create a unique name for the backup folder.
- The `mkdir` command creates the backup folder.
- Files are copied using `rsync`, excluding existing backup folders (`backup_*`) to avoid recursive copying.
- If copying fails, the incomplete backup folder is removed, and the script exits with an error.

3. Rotation Mechanism

```
mapfile -t BACKUPS < <(find "$TARGET_DIR" -maxdepth 1 -type d -name "backup_*" | sort
-r)

if [[ ${#BACKUPS[@]} -gt 3 ]]; then
    echo "Rotating backups..."
    for ((i=3; i<${#BACKUPS[@]}; i++)); do
        echo "Removing old backup: ${BACKUPS[$i]}"
        rm -rf "${BACKUPS[$i]}"
    done
fi
```

- The `find` command identifies all existing backup folders (`backup_*`) in the target directory and sorts them in descending order (newest first).
- If there are more than 3 backups, older backups are identified using `tail` and deleted using `rm -rf`.

4. Output Messages

```
echo "Backup created: $BACKUP_DIR"
echo "Rotating backups..."
echo "Removing old backup: ${BACKUPS[$i]}"
```

- The script provides clear feedback when:
  - A new backup is created.
  - Old backups are being removed during rotation.

Notes:

1. **Backup Exclusion**: Existing backup folders (`backup_*`) are excluded from being copied into new backups using `--exclude='backup_*'`.
2. **Error Handling**: The script ensures incomplete backups are cleaned up if any step fails.
3. **Retention Policy**: The rotation mechanism ensures that only the last 3 backups are retained, saving disk space.

This script is ideal for managing regular backups while keeping storage usage under control!