Backup Automation Script Documentation

This documentation outlines the use of a shell script to automate file and directory backups, with options for compression, logging, and cleanup. The script supports manual execution and scheduling via cron jobs.

Features

Input Parameters

1. **Source Directory (`<backup_from>`)**:
    - Specifies the directory to back up.
    - Must exist, or the script logs an error and exits.
2. **Destination Directory (`<backup_to>`)**:
    - Specifies the location to save the backup.
    - Created automatically if it doesn't exist.
3. **Compression Flag (`--compress`)** *(optional)*:
    - If provided, creates a compressed `.tar.gz` archive.
    - If omitted, copies the files to the destination directory.

Error Handling

1. **Source Directory Check**:
    - Logs an error and exits if the directory is missing.
2. **Destination Directory Check**:
    - Automatically creates the directory if it doesn't exist.
    - Logs an error and exits if creation fails.

Backup Process

1. **Compression Backup**:
    - Creates a compressed `.tar.gz` archive in the destination directory.
2. **Non-Compressed Backup**:
    - Copies files from the source to the destination in a new subdirectory.

Logging

- Logs all operations, errors, and timestamps in `backup.log`.
- Example log messages include success, failure, and cleanup events.

Cleanup

- Automatically removes backups older than **7 days** to manage storage:
  - Deletes compressed (`*.tar.gz`) and uncompressed backup directories.

Usage

Command Syntax

```
./backup.sh <backup_from> <backup_to> [--compress]
```

- Replace `<backup_from>` with the source directory path.
- Replace `<backup_to>` with the destination directory path.
- Add `--compress` to create a compressed backup.

Example Commands

1. **Backup without compression**:

   ```
   ./backup.sh /home/user/documents /backups
   ```

2. **Backup with compression**:

   ```
   ./backup.sh /home/user/documents /backups --compress
   ```

Scheduling with Cron

1. Open the cron editor:

   ```
   crontab -e
   ```

2. Add the following line to schedule the script daily at 2:00 AM:

   ```
   0 2 * * * /path/to/backup.sh /home/user/documents /backups --compress
   ```

Log File

- Location: `backup.log` in the script's directory.
- Contains:
  - Success and error messages for each operation.
  - Timestamps for actions performed.
  - Cleanup details.

Script Details

Backup Filename Format

1. **Compressed Backup**:
   - `backup_<source_name>_<YYYYMMDD_HHMMSS>.tar.gz`
2. **Uncompressed Backup**:
   - `backup_<source_name>_<YYYYMMDD_HHMMSS>` (as a subdirectory in the destination).

Old Backup Cleanup

- Removes:
  - Compressed backups older than 7 days (`*.tar.gz`).
  - Subdirectories older than 7 days (`backup_*`).
- Uses:

```
find "$DEST_DIR" -type f -name "*.tar.gz" -mtime +7 -exec rm -f {} \;
find "$DEST_DIR" -type d -name "backup_*" -mtime +7 -exec rm -rf {} \;
```

System Requirements

1. Linux or Unix-based system.
2. Sufficient permissions to read/write to the source and destination directories.

Customization

1. **Log File Location**:
   - Modify `LOG_FILE="backup.log"` to a preferred path.
2. **Retention Period**:

- Update the `-mtime +7` value in the `find` commands to adjust the cleanup duration.

Error Handling

1. **Missing Source Directory**:
   - Logs an error and exits with a failure status.
2. **Destination Directory Creation Failure**:
   - Logs an error and exits if the directory cannot be created.
3. **Backup Operation Failure**:
   - Logs errors for failed copy or compression operations.

Conclusion

This shell script automates backup tasks efficiently, ensuring critical data is archived securely. It incorporates logging, compression, and automatic cleanup, making it suitable for regular use or as a scheduled cron job.

Why This Documentation Fits Your Script:

- **Aligned Features**: Every feature in the script is covered, including compression, logging, and cleanup.
- **Precise Commands**: The usage examples directly map to how the script should be executed.
- **Cron and Cleanup**: Instructions for scheduling and cleanup match your script logic.
- **User-Friendly**: Clearly explains errors, logs, and customizations for different use cases.