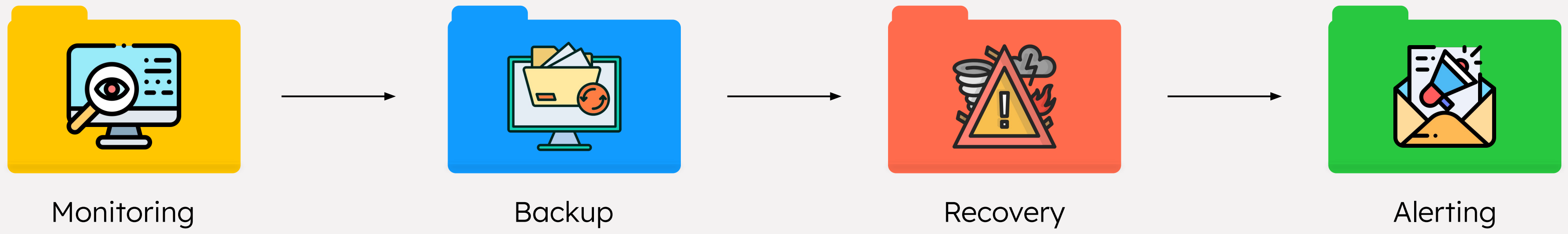

Monitoring
Backup
Recovery
Alerting

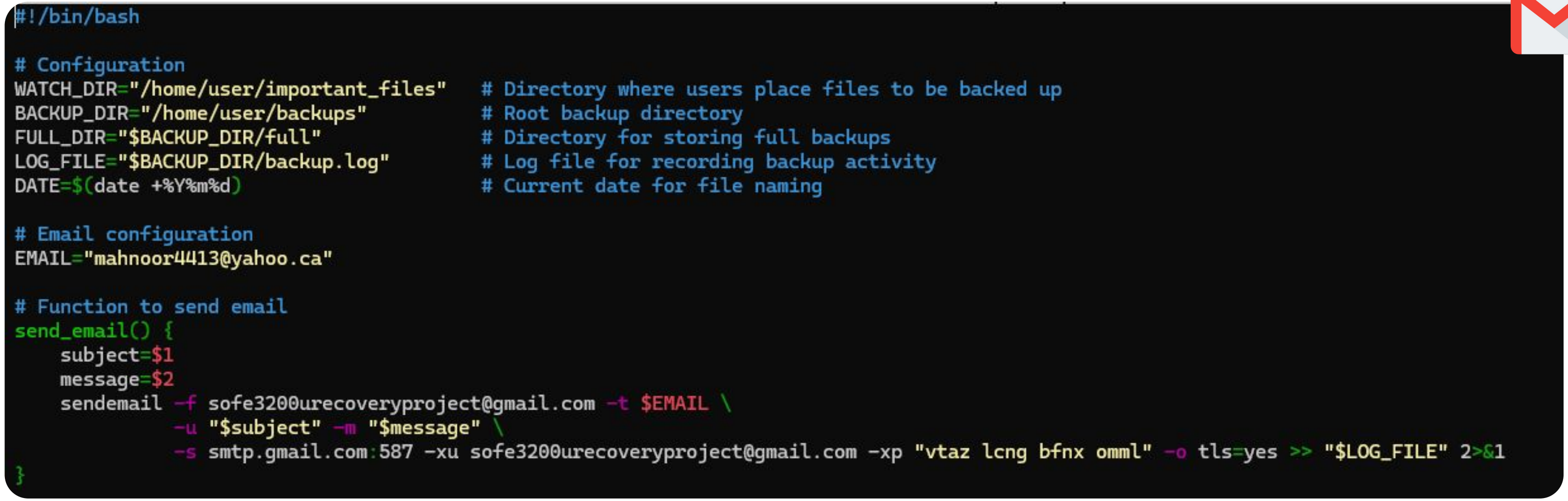
Overview

Back Up & Recovering with Alert and Monitoring System

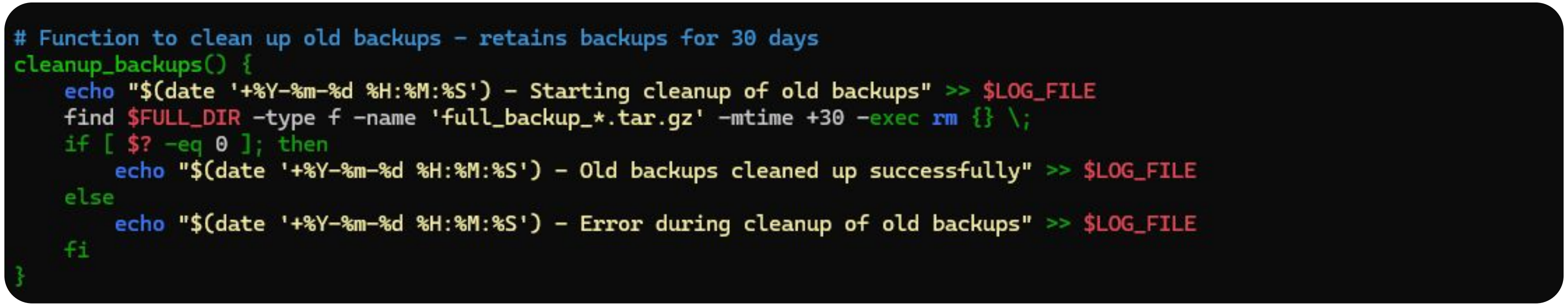
- Backup
  - Manages backups of specific files within the designated directory
- Monitoring
  - Tracks any new files added, changes made to existing files, or files that have been deleted
- Email
  - Notifies the user via email after each backup is completed
  - Works alongside the monitoring system to alert the user of any changes
- Recovery
  - Ensures every backup is securely stored to facilitate file recovery
- Automation
  - Automates backups to run at specified times each day

Script Structure

- In this section, it sets up the email notifications of any type of modifications and alerts
- It is split up in three subsections: Configuration, Email Configuration, and Send Function
- Configuration makes and sets up the directories.
- Email Configuration sets up what email it will be sent to and send function will send the email

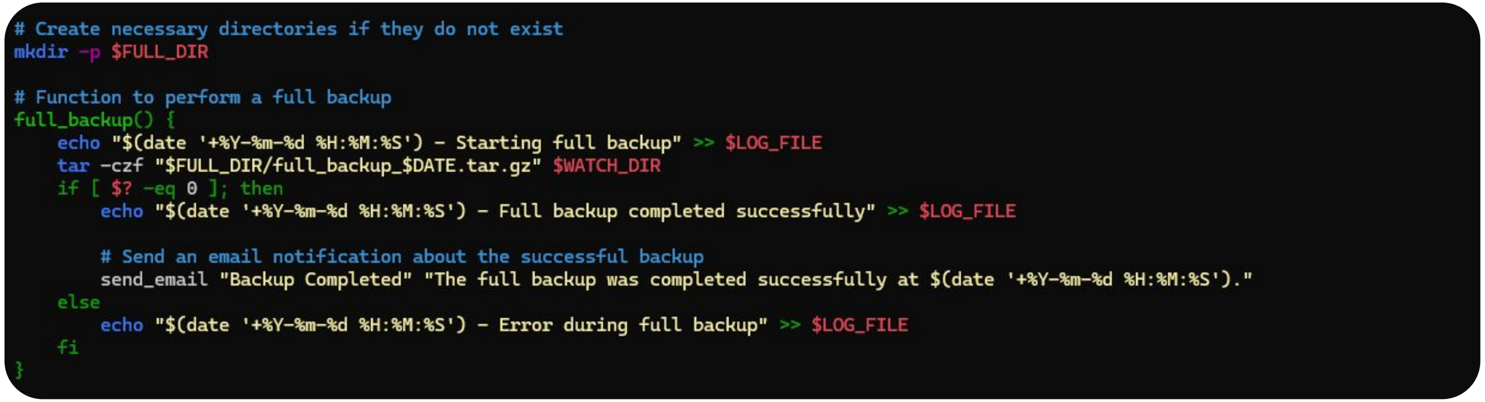


```
#!/bin/bash

# Configuration
WATCH_DIR="/home/user/important_files"    # Directory where users place files to be backed up
BACKUP_DIR="/home/user/backups"           # Root backup directory
FULL_DIR="$BACKUP_DIR/full"               # Directory for storing full backups
LOG_FILE="$BACKUP_DIR/backup.log"         # Log file for recording backup activity
DATE=$(date +%Y%m%d)                      # Current date for file naming

# Email configuration
EMAIL="mahnoor4413@yahoo.ca"

# Function to send email
send_email() {
    subject=$1
    message=$2
    sendemail -f sofe3200urecoveryproject@gmail.com -t $EMAIL \
              -u "$subject" -m "$message" \
              -s smtp.gmail.com:587 -xu sofe3200urecoveryproject@gmail.com -xp "vtaz lcng bfnx omml" -o tls=yes >> "$LOG_FILE" 2>&1
}
```



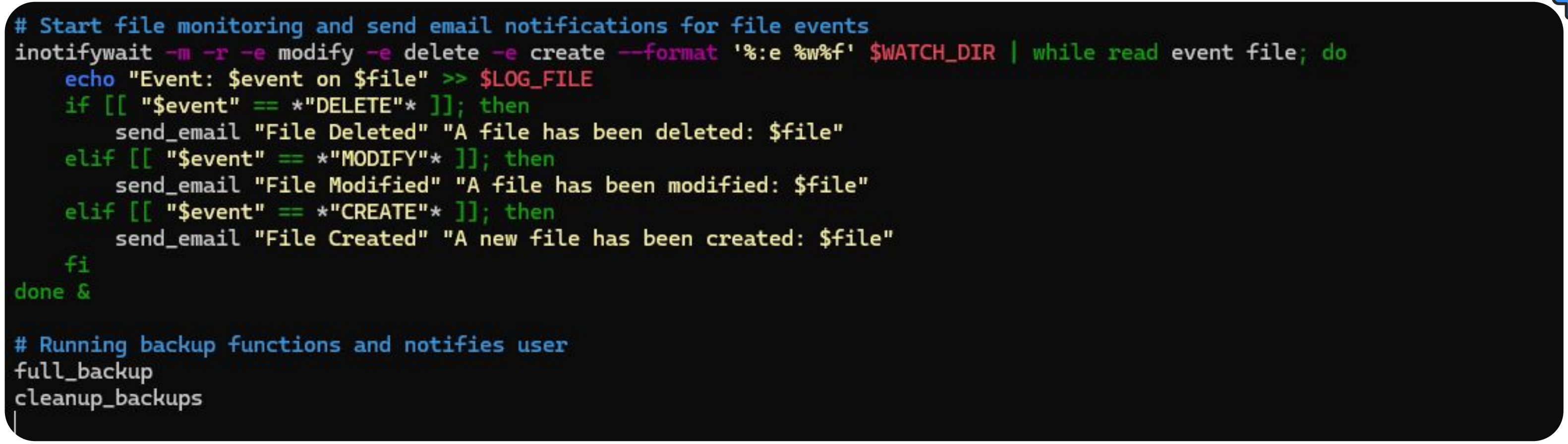
```
# Function to clean up old backups - retains backups for 30 days
cleanup_backups() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - Starting cleanup of old backups" >> $LOG_FILE
    find $FULL_DIR -type f -name 'full_backup_*.tar.gz' -mtime +30 -exec rm {} \;
    if [ $? -eq 0 ]; then
        echo "$(date '+%Y-%m-%d %H:%M:%S') - Old backups cleaned up successfully" >> $LOG_FILE
    else
        echo "$(date '+%Y-%m-%d %H:%M:%S') - Error during cleanup of old backups" >> $LOG_FILE
    fi
}
```

- In this section, the primary focus is on setting up a function that will handle cleaning up any old back ups.
- Each backup is set up to be saved for 30 days before the function cleans it up and makes space for incoming backups.
- <u>Error Handling:</u> If there happens to be an error. The system will recognize it and send alert stating “Error during cleanup of old backup” and it will be sent to the log files. I



```
# Create necessary directories if they do not exist
mkdir -p $FULL_DIR

# Function to perform a full backup
full_backup() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - Starting full backup" >> $LOG_FILE
    tar -czf "$FULL_DIR/full_backup_$DATE.tar.gz" $WATCH_DIR
    if [ $? -eq 0 ]; then
        echo "$(date '+%Y-%m-%d %H:%M:%S') - Full backup completed successfully" >> $LOG_FILE

        # Send an email notification about the successful backup
        send_email "Backup Completed" "The full backup was completed successfully at $(date '+%Y-%m-%d %H:%M:%S')."
    else
        echo "$(date '+%Y-%m-%d %H:%M:%S') - Error during full backup" >> $LOG_FILE
    fi
}
```

- The primary function of this section is perform the full backup and send the results to the log file, where it will be stored for 30 days.
- It first focuses on making sure that there is all the necessary directories if they do not exist.
- Then the main function will focus on performing the full backup successfully, once it does the system will send an email notification regarding the successful backup
- Error Handling: If there is error during the full backup, it will state “date- Error during full backup” and send it to the log file.

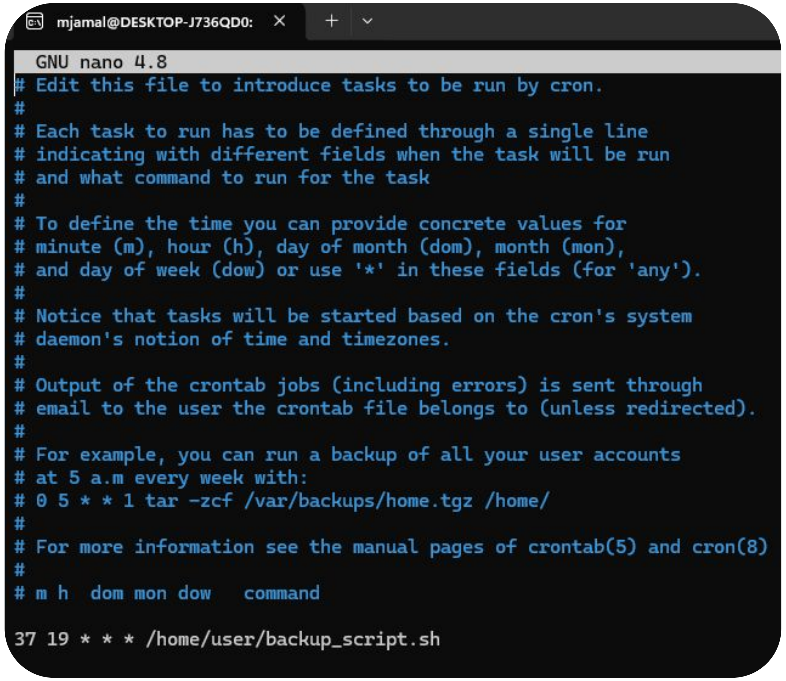


```
# Start file monitoring and send email notifications for file events
inotifywait -m -r -e modify -e delete -e create --format '%:e %w%f' $WATCH_DIR | while read event file; do
    echo "Event: $event on $file" >> $LOG_FILE
    if [[ "$event" == *"DELETE"* ]]; then
        send_email "File Deleted" "A file has been deleted: $file"
    elif [[ "$event" == *"MODIFY"* ]]; then
        send_email "File Modified" "A file has been modified: $file"
    elif [[ "$event" == *"CREATE"* ]]; then
        send_email "File Created" "A new file has been created: $file"
    fi
done &

# Running backup functions and notifies user
full_backup
cleanup_backups
```

- In this section, the main focus is to start the file monitoring process and send separate email notifications for file events. The file events are:
    - An event for monitoring when a files is being deleted. An email will be then sent to notify the user.
    - User will be notified for when there is event where a file is being modified.
    - If a file is being created, this will be considered as an event and send an email alert.
- Once the system has been monitored for each file event, the last section will focus on running the backup functions and then will finally notify the user.

Cron

Automation



```
mjamal@DESKTOP-J736QD0:
 GNU nano 4.8
# Edit this file to introduce tasks to be run by cron.
#
# 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).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

37 19 * * * /home/user/backup_script.sh
```

- File that contains the schedule of cron entries to be run and at specified times
- Crontab runs specified script at set time
- This line will ensure that the script is ran at a set time automatically and then notifies the user via email

Prerequisites

- Tar
- Smtp
- Sendemail
- Crontab
- Bash v5.0.17
- Inotify-tools

Installation

- sudo apt install tar
- sudo apt install msmtp msmtp-mta
- sudo apt install sendemail
- sudo apt install cron
- sudo apt install inotify-tools