Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Summer, Year: 2022), B.Sc. in CSE (Day)

Course Title: Operating System Lab

Course Code: CSE 310 Section: DI

- **Lab Project Name: Utility Software Application**

Student Details

Name		ID
1.	Sahin Alam	201002015

Submission Date : 09-05-2022

Course Teacher's Name : Mr. Mozdaher Abdul Quader

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …… **Signature:……**
Comments:…… **Date:……**

Table of Contents

Chapter 1
Introduction

Introduction:

We develop a utility software program. It is a graphical user interface (GUI) project. For creating interfaces, the gnome Library is used.

Utility software is designed to assist in the analysis, configuration, optimization, or maintenance of a computer. In contrast to application software, which is aimed at directly performing tasks that benefit ordinary users, it is used to support the computer infrastructure. Utilities, on the other hand, are frequently included in application systems. A batch job could, for example, run user-written code to update a database and then include a step that runs a utility to back up the database, or a job could compress a disk before copying files.

Although an operating system (OS) generally includes a basic set of utility programs, and this first-party utility software is frequently considered part of the operating system, users frequently install replacements or additional utilities. Those utilities may provide additional capabilities for tasks that are beyond the operating system's capabilities.

Many utilities that have the potential to affect the entire computer system require elevated privileges, whereas others that only affect the user's data do not.

Design Goal/Objective

OBJECTIVES/AIMS: The main goal of this project is to create a Shell application with a graphical user interface and some useful features. Use Shell script, For real-world applications. In order to improve our understanding of shell scripts.

Graphical shells are easy to use and impose a low load on new computer users. Most GUI-enabled operating systems also have CLI shells, which have their own set of drawbacks.

Chapter 2

Implementation of the project

What is Shell?

A shell is a program that exposes the services of an operating system to a human user or other applications. Depending on the purpose and operation of a computer, operating system shells employ either a command-line interface (CLI) or a graphical user interface (GUI). It is named a shell because it is the outermost layer around the operating system.

2.1 Features

There are three features in our shell application. Which's are

- Alarm
- Weather
- News

2.2 TOOLS & TECHNOLOGIES

- Linux Operating System.
- Bash Script (Bourne Again Shell).
- Zenity package (Gnome library package for building dialog box).
- Sox (Command line media player).
- Python.
- Rest API (OpenWeatherAPI & NewsAPI).
- VsCode(For text editor) and its terminal.

2.3 Environments

First, we upgrade our Linux operating system using the "apt get update" command. Then, one by one, we install Python, Zenity, and SOX. We go on to implementing our project after setting up all of the prerequisites.

2.4 Source Code

This is the main Window of the utility tool On running this shell script the user is prompted with a radio button Zenity User Interface offering three choices namely:

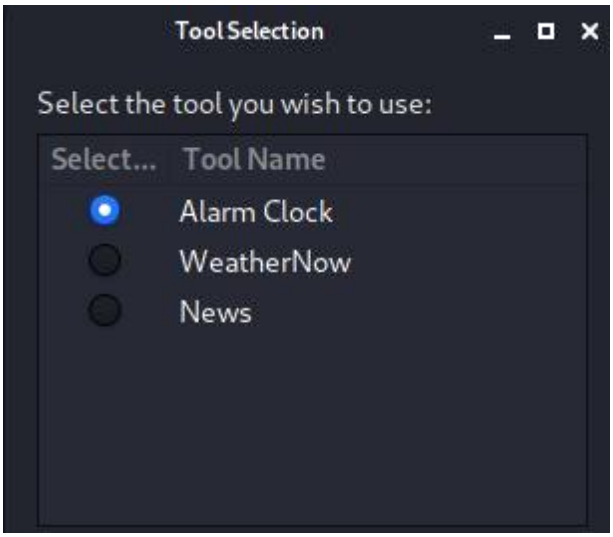
1.Alarm Clock

2.WeatherNow

3.News

On selecting the choice, the appropriate shell script runs

```
#!/bin/bash
#!/usr/bin/env python

opt1="Alarm Clock"
opt2="WeatherNow"
opt3="News"

#'int' stores the input request on clicking the appropriate button int=`zenity --height=275 --width=300 --list --radiolist --title 'Tool Selection' --text 'Select the tool you wish to use:' --column 'Select...' --column 'Tool Name' TRUE "$opt1" FALSE "$opt2" FALSE "$opt3" `
echo "Chosen option: "$int

#selecting the required shell script to run if [ "$int" == "$opt1" ] then

        #run the Alarm Clock script
        echo "`/bin/bash ./alarm3.sh`"
elif [ "$int" == "$opt2" ]
then
        #take in a place name as argument for the WeatherNow script
        place=$(zenity --entry --title "WeatherNow" --text "Enter place name") echo "`/bin/bash ./weather_script.sh $place`"

elif [ "$int" == "$opt3" ]
then
        #run the news script
        echo "`/bin/bash ./news_script.sh`"
else
        echo "No option selected"
fi
```


Tool Selection
Select the tool you wish to use:
Select...
Tool Name
Alarm Clock
WeatherNow
News

Chapter 3

Performance Evaluation

Application Performance

When choosing option 1. Alarms

This is the script that runs the alarm clock application when selected from the main menu of the zenity UI It takes in the hour, minute and am/pm and ringtone options through a zenity UI prompt and calculates the time remaining in seconds and puts the system to sleep for that amount of time After the time is elapsed we use a command called 'play' downloaded from the 'sox' package to play a file stored in the '/home/aman/os-lab/os-project/' directory until interrupted by ctrl-Z.

```
#!/bin/bash


#Storing hours of the day for the dropdown menu array=( 0 1 2
3 4 5 6 7 8 9 10 11 12 )

#Storing ringtone options for the dropdown menu
# ring=(0 CallingSanta Iphone XiaomiMi2 wakeupAlarmTone )

ring=( 0 CallingSanta Iphone XiaomiMi2 wakeupAlarmTone )

#am/pm option
ap=( 0 am pm )


#input hours, store in 'H'
H=$(zenity --entry --title "Alarm clock" --text "${array[@]}" --text "Hour") echo $H

#input minutes, store in 'M'
M=$(zenity --entry --title "Alarm clock"                    --text "Minute")
echo $M

#input am/pm, store in 'T'
T=$(zenity --entry --title "Alarm clock" --text "${ap[@]}" --text "AM/PM") echo $T
```

```
#select ringtone, store in 'A'
A=$(zenity --entry --title "Alarm clock" --text "${ring[@]}" --text "Select Ringtone")
echo $A
```

```
#if option selected is pm, a 12 hour offset is added(24-hr format) and stores it in variable 'newHour'

p="pm"

if [ "$T" == "$p" ]
then

        newH=$(($H+12))

        if [[ $H == 12          ]]
        then
                newH=$H
        fi
else
        newH=$H

fi

#take system Hour and Minute
curH=`date +"%H"`
curM=`date +"%M"`



#  echo "System hour: $curH; System minute: $curM" y=24
x=59

#failsafe incase 'newHour' exceeds 24(due to invalid input)
if [ "$newH" -gt "$y" ]
then
        while [ "$newH" -gt "$y" ]
        do
                read -p "Error!!! please enter appropriate value for hours: " H
                if [ "$T" == "$p" ]
                then
                        newH=$((H+12))
                else
                        newH=$H
                fi
        done
```

```
        #failsafe incase 'M'(stores user input minute) exceeds 59
        if [ "$M" -gt "$x" ]
        then
                while [ "$M" -gt "$x" ]
                do
                        read -p "Error!!! please enter appropriate value for minute: " M done

        fi
fi

#calculating the difference in current time and alarm time #finds the hour
difference
if((curH>newH))
then
        diffH=$((24-curH+newH))
else
        diffH=$((newH-curH))
fi

#finds the minute difference
if((curM>M))
then
        diffM=$((60-curM+M))
else
        diffM=$((M-curM))
fi




#calculates difference in time in seconds and store in 'total'
secondH=$((diffH*3600))
secondM=$((diffM*60))
total=$((secondH+secondM))

echo -e "Time left for alarm to ring is : $diffH : $diffM (HH:MM)\n"

#do nothing for 'total' number of seconds

zenity --info --title="Time Left" --text="Time left for alarm to ring is :
$diffH : $diffM (HH:MM)" --width=600 --height=400
sleep $total
```

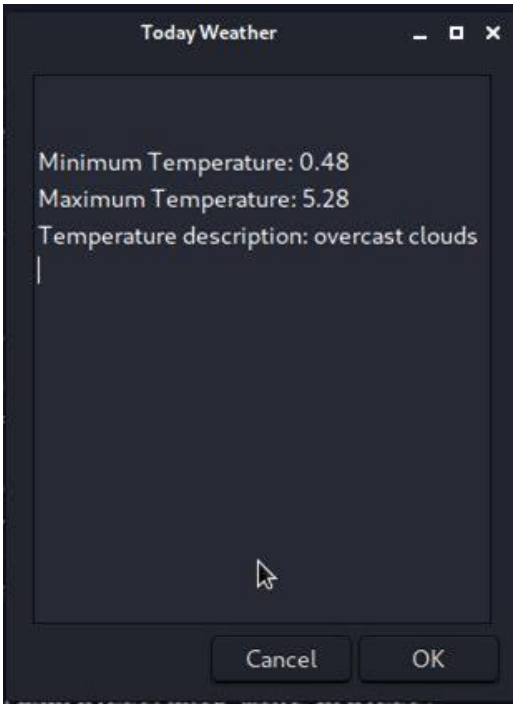
```
#run an infinite loop until ctrl-Z is pressed x=1
while [ $x -gt 0 ]
do
	echo "Press Control Z to switch off alarm"
	play "/home/aman/oslab/OSproject/os_project/$A.ogg"

done
```

When choosing option 2. Weather

After selecting option 2, the weather script file is executed, which in turn runs the weather tool.py python file. When the python code is run, it calls an open API (application programming interface) called OpenWeatherAPI, which fetches the current date's highest, lowest, and description of temperatures and saves them to the info.txt file. Then, using the zenity info dialoge, display the text from the info.txt file.

```
#!/bin/bash
python weather_tool.py                                          #Runs news_tool.py

FILE=`dirname $0`/info.txt

zenity --text-info \
	--title="Today Weather" \
	--filename=$FILE
\


#Weather_tools.py file

import requests                 #HTTP Library For Python
import sys                      #Library that enables command line input
CITY_NAME=sys.argv[1]           #Variable 'CITY_NAME' for storing the command line
input

#The JSON format is accessed using an openweather URL in which the city name and the app id will
have to be specified.
URL="http://api.openweathermap.org/data/2.5/weather?q="+CITY_NAME+"&mode=json&
units=metric&cnt=7&appid=85d97b3a5c555f0d8f3a9b3a21f36059"
```

```
#In the above URL,'CITY_NAME' is the name of the city and 'appid' is the registered key which is
available when you have a OpenWeatherMap registered account.
r = requests.get(URL) #GET request using python 'requests' library # print("Print details:",r.json())


temp_min = r.json()['main']['temp_min']
fo=open("info.txt","w")                                                    #Writi
ng down all the values in a file 'info.txt' which were stored in the
respective lists
fo.write("%s today temperature" % CITY_NAME )
fo.close()                              #Closing the file
fo=open("info.txt","a")                                                    #Writi
ng down all the values in a file 'info.txt' which were stored in the respective lists
fo.write("\n\nMinimum Temperature: %s\n" % temp_min)
fo.close()                              #Closing the file
temp_max = r.json()['main']['temp_max']
fo=open("info.txt","a")                                                    #Writi
ng down all the values in a file 'info.txt' which were stored in the
respective lists
fo.write("\nMaximum Temperature: %s\n" % temp_max)
fo.close()                                                                 #Closi
ng the file

description = r.json()['weather'][0]['description']
fo=open("info.txt","a")                                                    #Writi
ng down all the values in a file 'info.txt' which were stored in the
respective lists
fo.write("\nTemperature description: %s\n" % description)
fo.close()                                                                 #Closi
ng the file
```

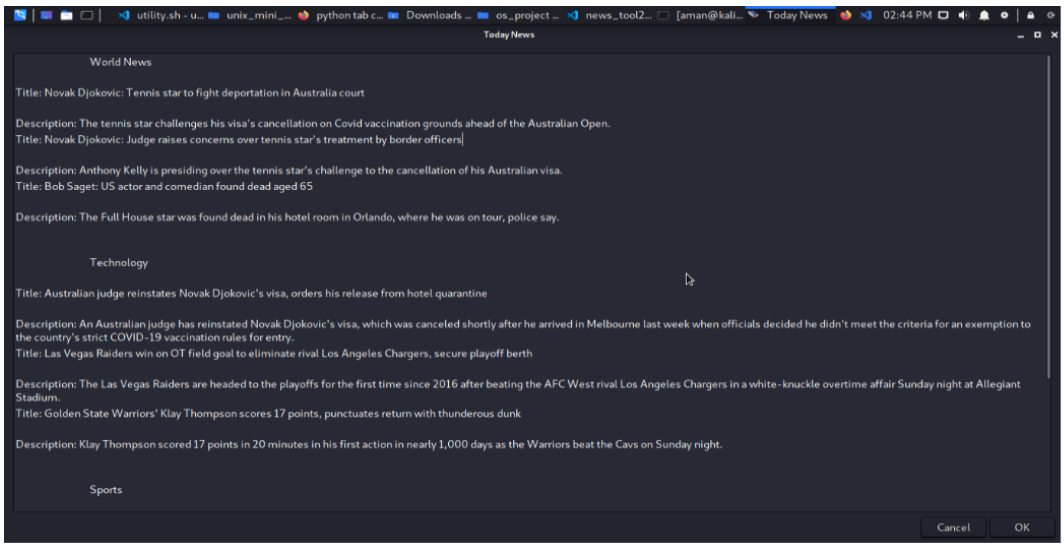


When choosing option 3. News

After selecting option 3, the news script file is executed, which in turn runs the news python files. When the python code is run, it calls an open API called NewAPI, which fetches the current date's news then save the news into the info1.txt file. For world news run news_tool.py , for technology news run news_tool1.py and and for sports news it's run news_tool2.py. Then, using the Zenity info dialog, display the text from the info1.txt file.

```
#!/bin/bash

python news_tool.py                    #Runs news_tool.py
python news_tool1.py                   #Runs news_tool1.py
python news_tool2.py                   #Runs news_tool2.py

FILE=`dirname $0`/info1.txt

zenity --text-info \
        --title="Today News" \
        --filename=$FILE \
        --width=600 --height=400
```

```
echo "World News"
for (( i=0; i<3; i++ ))                    #Running the for loop to extract the
data of World new for array 'arr'
do
        echo "${arr[i]}"
        echo "${arr[i+3]}"
        echo
done
echo
echo

echo "Technology"                          #Running the for loop to extract the
data of Technology news for array 'arr'
for (( i=6; i<9; i++ ))
do
        echo "${arr[i]}"
        echo "${arr[i+3]}"
        echo
done
echo
echo

echo "Sports"                              #Running the for loop to extract the
data of Sports news for array 'arr'
for (( i=12; i<15; i++ ))
do
        echo "${arr[i]}"
        echo "${arr[i+3]}"
        echo
done
```



Chapter 3
Conclusion

4.1 Conclusion

For each software product, there is always room for improvement. It doesn't matter how good or efficient it is. The most significant factor, though, should be the ability to absorb further changes. We're just dealing with these three features for now. This program may be expanded in the future to incorporate functions such as a converter and a music player. We can also change the app's UI to make it more appealing and responsive. We may arrange the specifics of each feature in a more organized manner.

4.2 Practical Implications

The practical implication of an event is the conclusion or end result that occurs when specified occurrences occur. In this instance, practical refers to the actual results of an event, whereas implication refers to the logical relationship between the occurrence and the conclusion.

4.3 Scope of Future work

We will improve the application in the future by adding more features such as

- CGPA Calculator
- Unit conversion
- Money Conversion
- Modify UI
- Music Players
- Improve current features

Chapter 3

References

1. DevHint (Click Here)
2. Shell Cheat Sheet (Click Here)
3. Zenity (Click Here , link2)
4. SOX (Click Here)
5. Python (Click Here)
6. Rest API (Click Here)
7. Lab Mannuals (Click Here)

----THE END----