IOT Lab Manual

Internet of things (Rajasthan Technical University)

1. Start Raspberry Pi and try various Linix commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, ping etc
To execute various Linux commands on your Raspberry Pi, you need to access the terminal.

Follow these steps:

1. **Power On Raspberry Pi:**

   - Connect your Raspberry Pi to a power source.

   - Ensure that the necessary peripherals (keyboard, mouse, display) are connected.

2. **Access the Terminal:**

   - Once your Raspberry Pi has booted up, you can access the terminal by clicking on the terminal icon or by using the keyboard shortcut `Ctrl + Alt + T`.

3. **Execute Linux Commands:**

   - **`ls` (List Files and Directories):**

```bash
ls
```

   - **`cd` (Change Directory):**

```bash
cd /path/to/directory
```

   - **`touch` (Create Empty File):**

```bash
touch filename.txt
```

   - **`mv` (Move/Rename File or Directory):**

```bash
```

```
mv sourcefile destination
```

- **`rm` (Remove/Delete File or Directory):**

  ```bash
  rm filename.txt
  ```

- **`man` (Manual):**

  ```bash
  man command
  ```

- **`mkdir` (Make Directory):**

  ```bash
  mkdir directoryname
  ```

- **`rmdir` (Remove Empty Directory):**

  ```bash
  rmdir directoryname
  ```

- **`tar` (Archive Tool):**

  ```bash
  tar -cvf archive.tar /path/to/directory
  ```

- **`gzip` (Compress a File):**

  ```bash
  ```

gzip filename.txt

- **`cat` (Concatenate and Display File Contents):**

```bash

cat filename.txt

```

- **`more` (Display File Content Page by Page):**

```bash

more filename.txt

```

- **`less` (Display File Content with Navigation):**

```bash

less filename.txt

```

- **`ps` (Display Process Status):**

```bash

ps aux

```

- **`sudo` (Execute a Command with Superuser Privileges):**

```bash

sudo command

```

- **`cron` (Schedule Tasks):**

```bash
```

crontab -e

- **`chown` (Change File Owner):**

```bash

chown newowner filename.txt

```

- **`chgrp` (Change File Group):**

```bash

chgrp newgroup filename.txt

```

- **`ping` (Check Network Connectivity):**

```bash

ping example.com

```

Feel free to experiment with these commands, but exercise caution, especially when using commands like `rm` (which deletes files) or commands with `sudo` (which have elevated privileges). Always double-check your commands to avoid unintentional data loss or system changes.

2. Run some python programs on Pi like:

a) Read your name and print Hello message with name

b) Read two numbers and print their sum, difference, product and division.

c) Word and character count of a given string.

d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.

Certainly! To run Python programs on your Raspberry Pi, you can use a text editor like Nano or any other code editor installed on your system. Here are Python programs for the specified tasks:

### a) Read your name and print a Hello message with the name:

```python
# Program to read a name and print a Hello message

# Read name from the user
name = input("Enter your name: ")
# Print a Hello message with the name
print("Hello, " + name + "!")
```

### b) Read two numbers and print their sum, difference, product, and division:

```python
# Program to read two numbers and perform basic arithmetic operations
# Read two numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
# Perform arithmetic operations
sum_result = num1 + num2
difference_result = num1 - num2
product_result = num1 * num2
# Avoid division by zero
if num2 != 0:
   division_result = num1 / num2
else:
   division_result = "Cannot divide by zero."
# Print the results
print("Sum:", sum_result)
print("Difference:", difference_result)

print("Product:", product_result)

print("Division:", division_result)
```
### c) Word and character count of a given string:

```python
# Program to count words and characters in a given string

# Read a string from the user

input_string = input("Enter a string: ")

# Count words and characters

word_count = len(input_string.split())

char_count = len(input_string)

# Print the results

print("Word Count:", word_count)

print("Character Count:", char_count)
```

### d) Area of a given shape (rectangle, triangle, and circle) reading shape and appropriate values from standard input:

```python
# Program to calculate the area of a shape based on user input

import math

# Read the shape from the user

shape = input("Enter the shape (rectangle, triangle, or circle): ")

# Calculate the area based on the shape

if shape == "rectangle":

    length = float(input("Enter the length of the rectangle: "))
    width = float(input("Enter the width of the rectangle: "))
    area = length * width
elif shape == "triangle":
    base = float(input("Enter the base of the triangle: "))
    height = float(input("Enter the height of the triangle: "))
    area = 0.5 * base * height
elif shape == "circle":
    radius = float(input("Enter the radius of the circle: "))
    area = math.pi * radius ** 2
else:
    print("Invalid shape entered. Please enter rectangle, triangle, or circle.")
    area = None
# Print the result
if area is not None:
    print("Area of the", shape.capitalize() + ":", area)
```

Copy and paste each program into a separate Python file (e.g., `program_a.py`, `program_b.py`, `program_c.py`, `program_d.py`) and run them using the command:

```bash
python program_a.py
``Replace `program_a.py` with the name of the specific file you want to run.
```

3. Run some python programs on Pi like:

a) Print a name 'n' times, where name and n are read from standard input, using for and while loops.

b) Handle Divided by Zero Exception.

c) Print current time for 10 times with an interval of 10 seconds.

d) Read a file line by line and print the word count of each line.

Certainly! Below are Python programs for the specified tasks:

### a) Print a name 'n' times, where name and n are read from standard input, using for and while loops:

```python

# Program to print a name 'n' times using for and while loops

# Read name and n from the user

name = input("Enter a name: ")

n = int(input("Enter the number of times to repeat: "))

# Using for loop

print("Using for loop:")

for _ in range(n):

    print(name)

# Using while loop

print("\nUsing while loop:")

count = 0
```

```
while count < n:
    print(name)
    count += 1
```

### b) Handle Divided by Zero Exception:

```python
# Program to handle Divided by Zero Exception
# Read two numbers from the user
num1 = float(input("Enter the numerator: "))
num2 = float(input("Enter the denominator: "))
# Handle Divided by Zero Exception
try:
    result = num1 / num2
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Cannot divide by zero.")
```

### c) Print current time for 10 times with an interval of 10 seconds:

```python
# Program to print the current time for 10 times with a 10-second interval
import time
# Set the number of iterations
iterations = 10
# Print the current time with a 10-second interval
```

```python
for _ in range(iterations):
    current_time = time.strftime("%H:%M:%S", time.localtime())
    print("Current Time:", current_time)
    time.sleep(10)
```

### d) Read a file line by line and print the word count of each line:

```python
# Program to read a file line by line and print the word count of each line
# Read the file name from the user
file_name = input("Enter the file name: ")
# Read the file line by line and print word count for each line
try:
    with open(file_name, 'r') as file:
        for line_number, line in enumerate(file, start=1):
            word_count = len(line.split())
            print(f"Line {line_number}: Word Count - {word_count}")
except FileNotFoundError:
    print(f"Error: File '{file_name}' not found.")
except Exception as e:
    print(f"Error: {e}")
```

For each program, copy and paste it into a separate Python file (e.g., `program_a.py`, `program_b.py`, `program_c.py`, `program_d.py`) and run them using the command:

```bash
python program_a.py
```

Replace `program_a.py` with the name of the specific file you want to run.

4. a) Light an LED through Python program b) Get input from two switches and switch on corresponding LEDs c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file.

To interact with hardware components like LEDs and switches on a Raspberry Pi using Python, you'll likely use a library like RPi.GPIO. Make sure to install it first if you haven't already:

```bash
pip install RPi.GPIO
```

Here are Python programs for the specified tasks:

### a) Light an LED through a Python program:

```python
import RPi.GPIO as GPIO

import time

# Set up GPIO mode and pin

led_pin = 17  # Change this to the actual GPIO pin number you are using

GPIO.setmode(GPIO.BCM)

GPIO.setup(led_pin, GPIO.OUT)

# Turn on the LED for 5 seconds

GPIO.output(led_pin, GPIO.HIGH)

time.sleep(5)

# Turn off the LED and cleanup GPIO

GPIO.output(led_pin, GPIO.LOW)

GPIO.cleanup()
```

### b) Get input from two switches and switch on corresponding LEDs:

```python
```

```
import RPi.GPIO as GPIO
import time
# Set up GPIO mode and pins for switches and LEDs
switch1_pin = 17  # Change this to the actual GPIO pin number for switch 1
switch2_pin = 18  # Change this to the actual GPIO pin number for switch 2
led1_pin = 23     # Change this to the actual GPIO pin number for LED 1
led2_pin = 24     # Change this to the actual GPIO pin number for LED 2

GPIO.setmode(GPIO.BCM)
GPIO.setup(switch1_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(switch2_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led1_pin, GPIO.OUT)
GPIO.setup(led2_pin, GPIO.OUT)
try:
    while True:
        # Check the state of switch 1
        if GPIO.input(switch1_pin) == GPIO.LOW:
            GPIO.output(led1_pin, GPIO.HIGH)
        else:
            GPIO.output(led1_pin, GPIO.LOW)
        # Check the state of switch 2
        if GPIO.input(switch2_pin) == GPIO.LOW:
            GPIO.output(led2_pin, GPIO.HIGH)
        else:
            GPIO.output(led2_pin, GPIO.LOW)
```

```
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

### c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file:

```python
import RPi.GPIO as GPIO
import time
# Set up GPIO mode and pin
led_pin = 17  # Change this to the actual GPIO pin number you are using
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
# Read on time and off time from a file (on_time, off_time in seconds)
try:
    with open("flash_times.txt", "r") as file:
        on_time, off_time = map(float, file.readline().split())
except FileNotFoundError:
    print("Error: File 'flash_times.txt' not found.")
    on_time, off_time = 1, 1  # Default values
try:
    while True:
        # Turn on the LED for on_time seconds
        GPIO.output(led_pin, GPIO.HIGH)
        time.sleep(on_time)
        # Turn off the LED for off_time seconds

        GPIO.output(led_pin, GPIO.LOW)

        time.sleep(off_time)

except KeyboardInterrupt:

    pass

finally:

    GPIO.cleanup()
```

Make sure to connect the LEDs and switches to the appropriate GPIO pins on your Raspberry Pi and adjust the pin numbers accordingly in the programs.

5. a) Flash an LED based on cron output (acts as an alarm) b) Switch on a relay at a given time using cron, where the relay's contact terminals are connected to a load. c) Get the status of a bulb at a remote place (on the LAN) through web.

To achieve these tasks using Python on a Raspberry Pi, you can use the `RPi.GPIO` library for controlling GPIO pins and the `cron` scheduler for scheduling tasks. Additionally, for remote control and status monitoring (Task c), you can create a simple web interface using Flask.

### a) Flash an LED based on cron output (acts as an alarm):

Python script (`alarm_flash.py`):

```python

import RPi.GPIO as GPIO

import time

# Set up GPIO mode and pin

led_pin = 17  # Change this to the actual GPIO pin number you are using

GPIO.setmode(GPIO.BCM)

GPIO.setup(led_pin, GPIO.OUT)

# Flash the LED for 10 seconds

GPIO.output(led_pin, GPIO.HIGH)
```

```
time.sleep(10)
GPIO.output(led_pin, GPIO.LOW)
# Clean up GPIO
GPIO.cleanup()
```

Cron Job:

```
# Edit the crontab file
crontab -e
# Add the following line to run the script every day at 7:00 AM
0 7 * * * python3 /path/to/alarm_flash.py
```

### b) Switch on a relay at a given time using cron, where the relay's contact terminals are connected to a load:

Python script (`switch_relay.py`):

```python
import RPi.GPIO as GPIO
import time
# Set up GPIO mode and pin
relay_pin = 17  # Change this to the actual GPIO pin number you are using
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay_pin, GPIO.OUT)
# Switch on the relay for 5 seconds
GPIO.output(relay_pin, GPIO.HIGH)
time.sleep(5)
```

```
GPIO.output(relay_pin, GPIO.LOW)
# Clean up GPIO
GPIO.cleanup()
```

Cron Job:

```
# Edit the crontab file
crontab -e
# Add the following line to run the script every day at 8:00 AM
0 8 * * * python3 /path/to/switch_relay.py
```

### c) Get the status of a bulb at a remote place (on the LAN) through the web:

Python script (`bulb_status.py`):

```python
from flask import Flask, render_template
import RPi.GPIO as GPIO
app = Flask(__name__)
# Set up GPIO mode and pin
bulb_pin = 17  # Change this to the actual GPIO pin number you are using
GPIO.setmode(GPIO.BCM)
GPIO.setup(bulb_pin, GPIO.IN)
@app.route('/')
def index():
    bulb_status = GPIO.input(bulb_pin)
    return render_template('index.html', bulb_status=bulb_status)
```

```
if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5000)
```

HTML template (`templates/index.html`):

```html
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Bulb Status</title>

</head>

<body>

    <h1>Bulb Status</h1>

    <p>The bulb is {% if bulb_status == 1 %}ON{% else %}OFF{% endif %}</p>

</body>

</html>
```

Run the Flask web application using the following command:

```bash
python3 bulb_status.py
```

Access the status page through a web browser at `http://raspberry_pi_ip:5000/`. Replace `raspberry_pi_ip` with the actual IP address of your Raspberry Pi on the LAN.