

RASPBERRY PI EXERCISES PACKET

PREFACE

Scope of this exercise packet:

1. Perform basic system tasks to administer your PI yourself. In order to do that, you should know about basics in the Linux OS itself including
 - Setting your own user environment
 - File System
 - A bit about remote access
 - Navigating around Linux system
 - Administrate processes
 - Writing Scripts
2. Work with GPIO with sensors
3. Basics in Python

To be added in the future:

Explore IOT

Ref to : online PI Tutorial Packet at <http://learn.stormingrobots.com>.

CONTENTS

- SECTION 1 - GETTING STARTED AND MANAGE YOUR PI 3**
 - I - 1.A) Try out most commonly used commands4
 - I - 2) Learn Vim6
 - I - 3) Shell Exercises6
- SECTION II - GPIO EXERCISES..... 8**
 - II - 1) Write three scripts :8
 - II - 2) Write the same program in C program.....9
- SECTION III - KERNEL LEVEL PROGRAMMING FOR I2C INTERFACE..... 11**
 - III - 1) Practice Your base conversion..... 11
 - III - 2) Program to communicate with your I2C devices..... 12
 - III - 3) Program with a PING sensor 12
- SECTION IV – PYTHON..... 14**
 - IV - 1) Exercises -Basics : 14
 - IV - 2) Exercises -List : 17
 - IV - 3) Exercises – harder one for List : 19
 - IV - 4) Exercises – Tuple : 20
 - IV - 5) Exercises – Dictionary:..... 21
 - IV - 6) Work with ADC to communicate with analog input device 22

SECTION 1 - GETTING STARTED AND MANAGE YOUR PI

[Utilize Tutorial Section I to IV](#)

- Do's and Don'ts
- Installation
- Remote access your PI
- Know basics about the File System
- Setting your own user environment
- Navigating around the file system
- Do remote access with VNC, or ssh
- Install and deinstall software
- Know how to set up your profile / bashrc
- Know how to write basic bash script and run it
- Ping hostnames between your own computer and PI, set up hostnames, etc.
- Know basics about VIM editor and customizing it.
- Set up email sender service
- Do backup

I - 1.A) TRY OUT MOST COMMONLY USED COMMANDS

***** (Observe the behavior after each command. Be Inquisitive)**

```
1. man ls
2. cd /
3. pwd
4. cd ~/; pwd
5. cd .vnc; pwd
6. cd .. ;
7. ls -lS
8. ls -ltra
9. ls -ltra | more
10. ls -ltra > allFiles.txt
11. ls -l | wc -l >> allFiles.txt
12. cat "this is the end !" >> allFiles.txt
13. cat allFiles.txt
14. ls -lt all*
15. rm allFiles.txt
16. dir
17. dir -R
18. ls -lr
19. ls -lR
20. ls -lR | grep "^d"
21. alias ldir='ls -lR | grep "^d"'           #you just created a new command!
22. ldir
23. clear
24. date
25. date +%D
26. date +"%T"
27. time
28. man -? > help.txt
29. tail -f help.txt
30. ls /root/
31. sudo ls /root/
32. wget https://www.stormingrobots.com/prod/pdf/csSyllabus.pdf &           #& == runs in background
33. du
34. du -sh
35. du -h * -d 1
36. du -h * -d 3
37. du -h * | sort -n
38. du -h * | sort -nr
39. du -a . | sort -nr | head
40. du -a . | sort -nr | tail
41. df
42. df -h
43. df /home
```

```

44. echo "echo \"Hello World \" " > test.sh
45. cat test.sh
46. ls -l test.sh    # you see: -rw-r--r-- 1 ... test.sh
47. ./test.sh      # fail to run
48. chmod +x test.sh
49. ls -l test.sh    # you see: -rwxr-xr-x 1 ... test.sh
50. ./test.sh
51. ls -l | grep "^\\."    # list all hidden files and directories
52. ls -l | grep "^d"      # list directories
53. ls -l | wc -l        # count of nodes (files, directories, links, etc.)
54. ps -ef
55. watch -n 1 "ps -ef" &
56. ps                # find the PID (process ID) for this watch command
57. kill 3000         # replace 3000 with the PID of the "watch" process
58. ps                # now you see the process is gone
59. clear
60. history
61. top
62. Ctrl+Z
63. Jobs              # find the job # , e.g. [9]
64. fg 9              # type the # 9 from, the previous finding
65. bg 9              # put back to the background
    <now, you need to kill this process... >
66. find . -name lxde-pi-rc.xml
67. find . -print | grep -i lx
68. uptime

```

—Special note about redirection (>)

About standard file descriptors : 1, 2 == standard output and standard errors

- Syntax: command > file same as command 1> file
- echo "hello" > file == echo "hello" 1> file
- echo "hello" >&2 file == echo "hello" 1>&2 file
- application 1>/dev/null == suppress the standard output

I - 2) LEARN VIM

- 1) Go thru the "CRASH COURSE TO LEARN VIM" in the tutorial packet.
OR
- 2) Use : <http://www.openvim.com/tutorial.html> interactive tutorial

Go to <http://www.openvim.com/sandbox.html> to practice. When you are doing trying all the keystrokes in the crash course tutorial. Move onto the next exercise below.

I - 3) SHELL EXERCISES

1. Tracking down where disk space has gone to . find out who has eaten up the most disk space.
2. Try out all the tutorial packet:
 - a. "COMMON ERRORS IN PARSING"
 - b. "SCRIPTS SAMPLES"
3. Create a few of aliases and put them in ~/.bash_aliases .
4. Display "Good Morning", "Good Afternoon", or "Good Night" based on system time.
5. Read a file into bash array. Display them.
6. Create a script to produce value of factorial of "N" where N is from user input.
7. Redo (6), but make thata factorial codes as a function which takes in a single integer parameters.
8. Create a script to find all prime numbers using Sieve of Eratheosenes under N where N is from user input
9. Write a script to:
 - a. ping to find the round-trip delay to www.google.com
 - b. Use traceroute to see the network route taken to www.google.com
 - c. Run it infinitely with an interval of every 1 minute.
 - d. And start it to run in the background.
10. Kill that ping background process.
11. Schedule your PI to say hello in every 5 minutes.
12. Schedule your PI to remind you snack, and lunch time.
13. Schedule your PI to shutdown at 3:00pm, and reboot at 9:00am.
14. Write a script to monitor the growth of disk space, and schedule it to report whenever it increases by 100

Extras:

15. Customize your Prompt
16. Customize your vi environment and download the c.vim plug-in.
17. Write a script to display all the color from 16 to 255. E.g. printf "\e[48;5;24m Show This"

Raspberry Pi

016	017	018	019	020	021
022	023	024	025	026	027
028	029	030	031	032	033
034	035	036	037	038	039
040	041	042	043	044	045
046	047	048	049	050	051
052	053	054	055	056	057
058	059	060	061	062	063
064	065	066	067	068	069
070	071	072	073	074	075
076	077	078	079	080	081

18. Repeat Ex.9, but save the code to a file.

SECTION II - GPIO EXERCISES

[Utilize Tutorial Section V](#)

1. Writing scripts to manipulation basic digital devices via GPIO pins
2. Using WiringPI library to write the C programs to produce the same effect.

Special note for those who do not know C++ at all, do review the following link to at least get yourself familiar with how to read a class declaration and its methods:

Here is a simple Fruit/inventory purchase / sell class:

https://www.element14.com/community/community/code_exchange/blog/2013/03/06/c-tutorial--classes

In order to stay with the scope, this is the extent of OOPS (object-oriented programming system) you need to know.

II - 1) WRITE THREE SCRIPTS :

1. Turn on and off a LED with a push button with pull up . Display :
"Got Pushed... Light out" // if it is pushed
"Got Released... Light on" // if it is released
e.g. code segment

```
# gpio pin 27 to push button and a resistor to the GND  
gpio -g mode 27 up
```

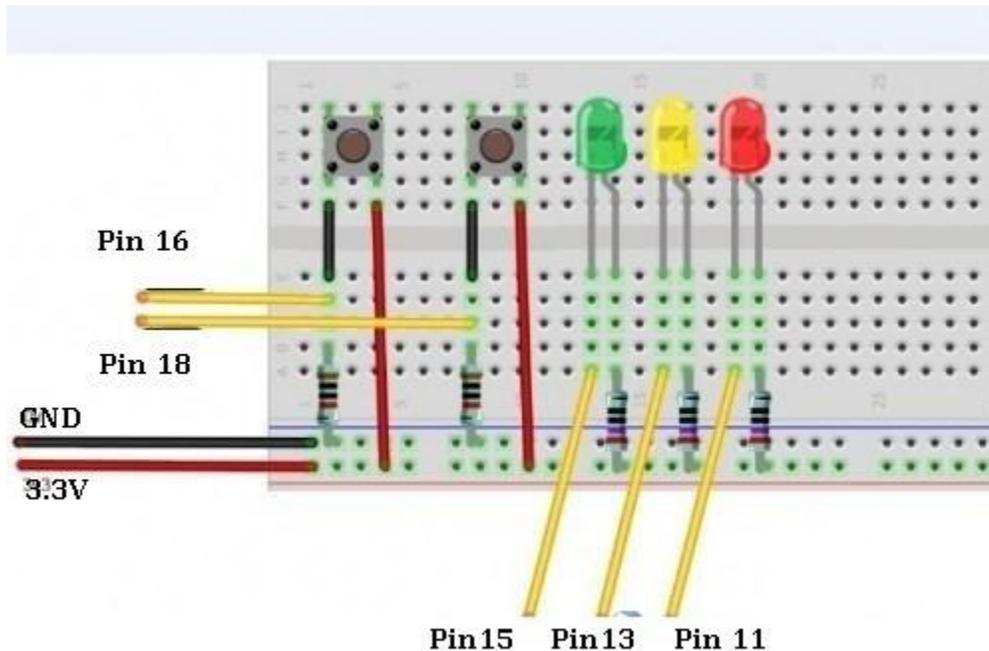
2. Turn on and off a LED with a push button with pull down. Display :
"Got Pushed... Light on" // if it is pushed
"Got Released... Light out" // if it is released

e.g. code segment

```
# gpio pin 27 to push button and a resistor to the 3V line  
gpio -g mode 27 down
```

3. Take user input to determine if you want to run pull down or pull down
e.g.

```
> runLed.sh up
> runLed.sh down
```



II - 2) WRITE THE SAME PROGRAM IN C PROGRAM

```
#include <stdio.h> // Used for printf() statements
#include <wiringPi.h> // Include WiringPi library!

// Pin number declarations. We're using the Broadcom chip pin numbers.
const int pwmPin = 18; // PWM LED - Broadcom pin 18, P1 pin 12
const int ledPin = 23; // Regular LED - Broadcom pin 23, P1 pin 16
const int butPin = 17; // Active-low button - Broadcom pin 17, P1 pin 11

const int pwmValue = 75; // Use this to set an LED brightness

int main(void)
{
    // Setup stuff:
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

    pinMode(pwmPin, PWM_OUTPUT); // Set PWM LED as PWM output
    pinMode(ledPin, OUTPUT); // Set regular LED as output
```

```
pinMode(butPin, INPUT); // Set button as INPUT
pullUpDnControl(butPin, PUD_UP); // Enable pull-up resistor on button

printf("Blinker is running! Press CTRL+C to quit.\n");

// Loop (while(1)):
while(1)
{
    if (digitalRead(butPin)) // Button is released if this returns 1
    {
        pwmWrite(pwmPin, pwmValue); // PWM LED at bright setting
        digitalWrite(ledPin, LOW); // Regular LED off
    }
    else // If digitalRead returns 0, button is pressed
    {
        pwmWrite(pwmPin, 1024 - pwmValue); // PWM LED at dim setting
        // Do some blinking on the ledPin:
        digitalWrite(ledPin, HIGH); // Turn LED ON
        delay(75); // Wait 75ms
        digitalWrite(ledPin, LOW); // Turn LED OFF
        delay(75); // Wait 75ms again
    }
}

return 0;
}

gcc -Wall blinker.c -l wiringPi blinker
```

```
sudo ./blinker
```

SECTION III - KERNEL LEVEL PROGRAMMING FOR I2C INTERFACE

III - 1) PRACTICE YOUR BASE CONVERSION

Decimal (base-10) to Hexadecimal (base-16) and Binary (base-2)

Decimal (base-10)	Hexadecimal (base-16)	Binary (base-2)
12		
14		
13		
15		
16		
127		
255		
256		
1023		
1024		
2047		
2048		
65535		
65536		

Binary (base-2) to Hexadecimal (base-16) and Decimal (base-10)

Binary (base-2)	Hexadecimal (base-16)	Decimal (base-10)
00001		
00011		
00111		
01001		
110011		
1001110		
00001001		
10001001		
01001001		
10101000		

III - 2) PROGRAM TO COMMUNICATE WITH YOUR I2C DEVICES

[Utilize Tutorial Section VI](#) - writing Device System Calls to access I2C devices

Hook up the following i2c sensors and display all the following:

What to test with	What you need to do :
HiTechnic IR Seeker	<ul style="list-style-type: none"> • Get version • # from all zones. Tested with an IR Ball
Mindsensors light array	<ul style="list-style-type: none"> • Get version • # from all 8 light sensors. Tested with a lines
Mindsensors I2C adapter connected to ev3 light	<ul style="list-style-type: none"> • Get the raw data
Mindsensors I2C adapter	<ul style="list-style-type: none"> • Get version • Modify the I2C address of the module

III - 3) PROGRAM WITH A PING SENSOR

There are four pins on the ultrasound module that are connected to the Raspberry:

- VCC to Pin 2 (VCC)
- GND to Pin 6 (GND)
- TRIG to Pin 12 (GPIO18)
- connect the 330Ω resistor to ECHO.
- On its end you connect it to Pin 18 (GPIO24) and through a 470Ω resistor (pull down) you connect it also to Pin6 (GND).

Pseudo-code

Set to BCM

set GPIO Pins

trigger pin = 18 as output

echo pin = 24 as input

set trigger pin high

wait after 0.01ms

set trigger pin to low

StartTime = time.time()

StopTime = time.time()

```
while echo pin is low (0)  
    save the start time
```

```
While echo pin == 1  
    Save the stop time
```

```
elapsed time = stop time – start time
```

```
# sonic speed = 34300 cm/s
```

```
distance = (elapsed time * sonic speed) / 2 because it is round trip
```

SECTION IV – PYTHON

[Utilize Tutorial Section VII](#)

Know the basic data types and structure about Python 3.X.

Many of the following exercises come from w3resource.com.

IV - 1) EXERCISES -BASICS :

1. Display the current date and time.

Output for the sample:

Current date and time :

2014-07-05 14:34:14

2. Write a Python program to count repeated characters in a string. (non-case sensitive)

Sample input: Booboo

Output: b: 2

o : 4

3. Write a function to produce gcd for N number of values.

Sample User Input:

Enter # of numbers : 12 64 100

Output:

Gcd (12,64,100) = 4

Hint: use Euclid Algorithm:

Gcd (80, 45) = 5

N	M	R = n%m
80	45	35 = 80 % 45
45	35	10 = 45 % 35
35	10	5 = 35 % 10
10	5	0 = 10 % 5

4. Calculate number of days between two dates.

Sample dates : (2014, 7, 2), (2014, 7, 11)

Sample Output : 9 days

5. Write a very short program to do the following :

10 bottles of drink on the wall, 10 bottles of drink!
 So take one down, pass it around, 9 more bottles of drink on the wall!
 9 bottles of drink on the wall, 9 bottles of drink!
 So take one down, pass it around, 8 more bottles of drink on the wall!
 8 bottles of drink on the wall, 8 bottles of drink!
 So take one down, pass it around,
 ...
 1 bottle of drink on the wall, 1 bottle of drink!
 So take it down, pass it around, no more bottles of drink on the wall!

6. Write a program to ask user to enter a word, then convert to pig latin.

Sample:

- (if it starts with a vowel – a,e,i,o,u and y) "awkward" == "awkwarday"
- (if it starts with a consonant) : "happy" == "appyhay" ..

Well supposedly, there is another variation with consonant cluster too, such as "sm", "fl", "st"... Up to you if you want to play with that too.

- (if it starts with a consonant cluster) : "stamp" == "ampstay" ; "trash" == "ashtray"

7. Let's try out the math floor, and ceiling and simple truncation. Write a program to display floor | ceiling | truncation of $X / 3$ where $X = 10$

8. Write a program to generate 1st 1000 prime numbers (prefer using Sieve of Eratosthenes.

9. Write a Python program that accepts the sum of an infinite series of e^1 :

$$\sum_{k=0}^n \frac{1}{k!}$$

i.e. sum e^1 up to N where N is from user input = $1 + \frac{1}{1!} + \frac{1}{2!} + \dots + \frac{1}{N!}$

10. Write a program to display the Bits of a number, display with 4bits as a chunk.

Sample n: 4602
 Expected Result: 0001 0001 1111 1010

11. Write a program to check whether a specified value is contained in a group of values.

Sample Test Data (you can hard code this in your program): [1, 5, 8, 3]
 User enter: 3
 Sample Output : 3 is in the list
 User enter: 9
 Sample Output : 9 is not in the list

12. Write a program to check whether a specified value is contained in a group of values.

Sample Test Data (you can hard code this in your program): [1, 5, 8, 3]

User enter: 3

13. Given strings a and b, return a single string after swap the 1st letter of the two words

e.g.

'mix', 'pot' becomes 'mot pix'
solder ducks becomes 'dolder sucks'

Assuming a and b are length 2 or more.

14. Write a program to display all the words and their matching anagram set number. For example: Your words list:

```
char words[ ][100] = { "aaabc", " cbaad", "aadbc", "cbaaa", "cdcdd", "dcccdd", "abcaa" };
```

Output should display the following:

```
aaabc      1
cbaad      2
aadbc      2
cbaaa      1
cdcdd      3
dcccdd     3
abcaa      1
```

15. Write a simple hash method to store the following names to an unique entry in an array.

Since there are 9 names, the default table size = 11, or 13, or 17, etc.

	Sum	1 st attempt	2 nd attempt	3 rd attempt	4 th attempt
	e.g. a-a=0, m-a=12, y-a=24	Sum % 11	(Sum(i) +3) %11	(Sum(i) +5) %11	(Sum(i) +7) %11
amy	0+12+24 =36	3			
james	9+0+12+4+18=43	10			
bob	=1+14+1=16	5			
Jake	9+0+10+4=23	1			
Jammy	9+0+12+12+24=57	2			
Patty	15+0+19+19+24=77	0			
Avery	0+21+4+17+24=66	0	3	5	7
Booboo	1+14+14+1+14+14 =58	3	6		

After that, write a function to find get the entry index. For example:

Find ("Avery") returns 7

Assumption: not case sensitive; so Avery == avery

IV - 2) EXERCISES -LIST :

For the following exercises, make them all functions.

1. Ask users to enter numbers into a List.
2. Perform `sum()`, `multiply()` with the items in a list entered by users. Your program should also ask user which operation they wish to run.
3. Return the largest and smallest number from a list.
4. Find the size of each object in the list, and the size of the whole list.
5. Return the count of words that are `>2` in len and have the same first and last char .

Sample input List : ['abc', 'xyz', 'aba', '1221']
 Result for this sample : 2

6. Create a list, sorted in increasing order by the last element in each tuple from a given list of non-empty tuples.

Sample input List : [(2, 5), (1, 2), (4, 4), (2, 3), (2, 1)]
 Result for this sample:: [(2, 1), (1, 2), (2, 3), (4, 4), (2, 5)]

7. Remove duplicates from the list in (6).
8. Clone or copy a list.
9. What happened to the list:

```
lst = ["HELLO", 5, [10], True]
print(lst)
lst[2].append(50)
print(lst)
```

10. Give an arbitrary list (you can create one in your program). Then, ask yourself to enter a letter or phrase. Your program should check if the user-input exist in a given list.
11. Find the list of words that are longer than n from a given list of words.
12. Function to read in two lists and returns True if they have at least one common member
13. Print a specified list after removing the zero-th, 4th and 5th elements.

Sample input List : ['Red', 'Green', 'White', 'Black', 'Pink', 'Yellow']
 Result for this sample: ['Green', 'White', 'Black']

14. Generate a $3 \times 4 \times 6$ 3D array with each element = '*'

15. Write a Program to print the numbers of a specified list after removing even numbers from it.

16. Combine two lists, but remove all duplicates.

17. Get the frequency of the elements in a list.

18. Split a list of a set of numbers into 3x4.

19. Create a list by concatenating a given list which range goes from 1 to n.

Sample input List : ['p', 'q']

n =5

Result for this sample: ['p1', 'q1', 'p2', 'q2', 'p3', 'q3', 'p4', 'q4', 'p5', 'q5']

20. Create a program to flatten a shallow list .

Sample input List : [1,2,3] [5,6,7] [10, 7, 6]

Result for this sample ; [1,2,3,5,6,7,10,7,6]

21. Modify the program (20) to also remove the duplicate.

IV - 3) EXERCISES – HARDER ONE FOR LIST :

1. Check whether a list contains a sublist.
2. Check whether two lists are a palindrome .
3. Split and combine as following:

Sample input List: ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n']

Nth: 3

Output for the sample: [['a', 'd', 'g', 'j', 'm'], ['b', 'e', 'h', 'k', 'n'], ['c', 'f', 'i', 'l']]

4. Compute the similarity between two lists.

Sample input list: ["red", "orange", "green", "blue", "white"], ["black", "yellow", "green", "blue"]

Output for the sample: Color1-Color2: ['white', 'orange', 'red']

Color2-Color1: ['black', 'yellow']

5. Replace the last element in a list with another list.

Sample data : [1, 3, 5, 7, 9, 10], [2, 4, 6, 8]

Output for the sample: [1, 3, 5, 7, 9, 2, 4, 6, 8]

6. Find the list in a list of lists whose sum of elements is the highest.

Sample input Lists: [1,2,3], [4,5,6], [10,11,12], [7,8,9]

Output for the sample: [10, 11, 12]

7. Creates a list of words, and determine the number of sets of anagrams.

IV - 4) EXERCISES – TUPLE :

1. Create a tuple `tup = ("HELLO", 5, [], True)`, unpack them back into different variables.
2. Expand the `[]` to 50 of them.
3. create the colon of a tuple.
4. Find the repeated items of a tuple.
5. Convert a list to a tuple.
6. Remove an item from a tuple.
7. Slice a tuple.
8. Find the length of a tuple. Then, return Nth item of a tuple. It should also check if the tuple is empty also.
9. unzip a list of tuples into individual lists.
10. reverse a tuple.
11. Print a tuple with string formatting.
12. Replace last value of tuples in a list. Do both of the following input sample.
Sample input List: `[(10, 20, 40), (40, 50, 60), (70, 80, 90)]`
Output for the sample: `[(10, 20, 100), (40, 50, 100), (70, 80, 100)]`

Sample input list: `[(0, 0), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]`
Output for the sample: `[('',), ('a', 'b'), ('a', 'b', 'c'), 'd']`
13. Sort a tuple by its float element.
Sample input list: `[('item1', '12.20'), ('item2', '15.10'), ('item3', '24.5')]`
Output for the sample: `[('item3', '24.5'), ('item2', '15.10'), ('item1', '12.20')]`
14. Count the elements in a list until an element is a tuple.
15. Reverse a tuple.

Raspberry PI

IV - 5) EXERCISES – DICTIONARY:

<more to come>

IV - 6) WORK WITH ADC TO COMMUNICATE WITH ANALOG INPUT DEVICE

Work with With adafruit ADC

- 1) Now you will write a python program to detect black & white using the ADC and photocell .
 - Write the photocell and create a voltage divider
 - Program to detect white or black by using hardcoded threshold
- 2) Do (1) , except using adjustable threshold by using the potentiometer.
 - Photocell
 - Potentiometer
 - Program the potentiometer to create adjustable threshold