FOSS Lab Manual - Part 1

1.0, Jun 2011

Table of Contents

Chapter 1. Getting Started

1. Getting Started Lab I

1.1. Objective

Getting started with Linux, learn basic commands and directory structure, execute file and directory operations

1.2. Starting Terminal

Select Application # Accessories # Terminal

1.3. Download and Extract Resource Files

The `linux-resources` package contains files that are required while trying out examples and exercises in this chapter. Download the `linux-resources` package and extract it on to your home folder. The commands to achieve this are given below.

```
$ cd
$ wget http://foss-lab-manual.googlecode.com/files/linux-resources.tar.gz
$ tar --gunzip -x -f linux-resources.tar.gz
```

1.4. File Commands

Perform the following file related operations at the shell prompt.

1. Create file named `linux_distros.txt` Add the names of the linux distributions you know.
2. Display the content of the file `linux_distros.txt`.
3. Copy the file `linux_distros.txt` to a new file `list_of_linux_distributions.txt`.
4. Rename the file `list_of_linux_distributions.txt` to `linux.txt`.
5. Delete the file `linux.txt`.

```
$ cat > linux_distros.txt
debian
fedora
ubuntu
slackware
centos
OpenSuSE
^D

$ cat linux_distros.txt
debian
fedora
ubuntu
slackware
centos
OpenSuSE

$ cp linux_distros.txt list_of_linux_distributions.txt

$ mv list_of_linux_distributions.txt linux.txt

$ rm linux.txt
```

1.5. Directory Commands

Perform the following directory related operations at the shell prompt.

1. Get a list of files in the current directory.
2. Create a directory called `foss`.
3. Delete the directory `foss`
4. Create the following directory structure with a single command: `one/two/three`.
5. Delete the directory `one` with all its sub directories.
6. Print the path of current working directory.

1.6. Hidden files

Perform the following hidden files related operations at the shell prompt.

1. Create a hidden file called `.a`.
2. Create a hidden directory called `.config`.
3. List all the files in the current directory including hidden files.

```
$ touch .a

$ mkdir .config

$ ls -a
.       ..      .a      .config
```

1.7. Multiple Files

The directory `pattern` has files with names containing specific patterns. Perform the following operations on the files.

1. Goto the directory `pattern`.
2. Remove all files starting with `a`.
3. Remove files ending with `1.txt`.
4. Remove all files start with `doc` and ending with `.txt`.
5. Remove all files.

```
$ cd pattern

$ rm a*

$ rm *1.txt

$ rm doc*.txt

$ rm *
```

1.8. Shell Variables

Perform the following shell variable related operations at the shell prompt.

- Store the string `~/mybin` in a variable `myvar`
- List the contents of the directory using the variable.
- Copy the contents of the directory to `/tmp` using the variable.

```
$ myvar=~/mybin
```

```
$ ls $myvar
hello   ls

$ cp $myvar/* /tmp
```

1.9. PATH Variable

There are two executables `ls` and `hello` in `~/mybin`. Perform the following operations related to the PATH environment variable.

1. Take a backup of the PATH variable.
2. Invoke `hello`, the command will fail.
3. Add the directory `mybin` to the end of PATH, invoke `ls` and `hello`.
4. Add the directory `mybin` to the beginning of PATH, invoke `ls` and `hello`.
5. Restore your original PATH.

```
$ BPATH=$PATH

$ hello
bash: hello: command not found

$ PATH=$PATH:~/mybin
$ hello
Hello user!
$ ls
...

$ PATH=~/mybin:$PATH
$ hello
Hello user!
$ ls
custom ls from ~/mybin!

$ PATH=$BPATH
```

1.10. Finding Files

1. Using `find` determine the locations of all HTML files under `/usr`.
2. Using `find` determine the location of all directories called `bin` under `/usr`.
3. Using `locate` find out the location of all PNG files in the system.

```
$ find /usr -name "*.html"
...

$ find /usr -name bin -type d
/usr/bin
...

$ locate *.jpg
...
```

1.11. File Type

Find the file types of the following files:

1. `/bin/bash`

2. `/etc/passwd`

```
$ file /bin/bash
/bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.18, stripped

$ file /etc/passwd
/etc/passwd: ASCII text
```

1.12. Getting Help

1. Print the command usage for `mkdir`.
2. Read the manual page for `mkdir`. Press `q` to quit.

```
$ mkdir --help
...

$ man mkdir
```

2. Getting Started Lab II

2.1. Objective

Review redirection, pipes, filters and job control.

2.2. Redirection

Perform the following redirection related operations.

1. Perform a file copy using the `cat` command and redirection operators.
2. Join the 3 files `elements1.txt`, `elements2,txt`, `elements3.txt` into a single file using a single `cat` command and the redirection operator.
3. Join the 3 files using multiple `cat` commands and the redirection with append operator.

```
$ cat < cities.txt > copy.txt

$ cat elements1.txt elements2.txt elements3.txt > concat1.txt

$ cat elements1.txt > concat2.txt
$ cat elements2.txt >> concat2.txt
$ cat elements3.txt >> concat2.txt
```

2.3. Pagers

- Display the file `elements.txt` using the command `more`.
- Display the file `elements.txt` using the command `less`.

2.4. head and `tail`

The file `cities.txt` contains the 10 largest cities in the world (by population). Perform the following operations using `head` and `tail`.

1. Display the largest city in the world.
2. Display the 10th largest city in the world.
3. Display the 3rd largest city in the world.

```
$ cat cities.txt | head -n 1
```

```
$ cat cities.txt | tail -n 1

$ cat cities.txt | head -n 3 | tail -n 1
```

2.5. cut and sort

The file `elements.txt` contains the list of elements and their abbreviation. Perform the following operations on `elements.txt`, using `cut` and `sort`.

1. Sort all elements in `elements.txt` on the basis of abbreviation.
2. Display only the element names in `elements.txt`.

2.6. Job Control

1. Run `sleep 20`, suspend and resume it in the foreground.
2. Run `sleep 20`, suspend and resume it in the background.
3. Start `sleep 20` in the background.
4. Start multiple `sleep 20` in the background, kill a `sleep` process with the kill command, verify using `jobs`.

```
$ sleep 20
^Z
[1]+  Stopped                 sleep 20
$ fg
sleep 20
$
```

```
$ sleep 20
^Z
[1]+  Stopped                 sleep 20
$ bg
[1]+ sleep 20 &
$
[1]+  Done                    sleep 20
```

```
$ sleep 20 &
[1] 32172
$
[1]+  Done                    sleep 20
```

```
$ sleep 20 &
[1] 32178
$ sleep 20 &
[2] 32179
$ kill %1
[1]-  Terminated              sleep 20
$ jobs
[2]+  Running                 sleep 20 &
```

2.7. Processes

Perform the following process related operations at the shell prompt.

1. List all the processes in the system.
2. List all processes belonging to `root`.
3. List all processes corresponding to the program `getty`.

4. List the process tree of the system.

3. Getting Started Lab III

3.1. Objective

Review file ownership, file permissions, links and file system hierarchy.

3.2. Links

Perform the following links related operations, at the shell prompt.

1. Create a file, and then create a soft link to the file.
2. Check what happens when the original file is deleted.
3. Create a file, and then create a hard link to the file.
4. Check what happens when the original file is deleted.
5. Try creating a soft link and a hard link to a directory.

3.3. Permissions and Ownership

Perform the following permissions and ownership related operations, at the shell prompt.

1. Find the user-owner and group-owner of the file `/etc/passwd`.
2. Find the user-owner and group-owner of the file `linux-distros.txt`.
3. Change the permissions of the file `linux_distros.txt`, such that the file is writable by any user.
4. Change the permissions of the file `linux_distros.txt`, such that nobody can read the contents of the file.

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1553 Apr 19 17:38 /etc/passwd

$ ls -l linux_distros.txt
-rw-r--r-- 1 user user 0 Jun 15 06:29 linux_distros.txt

$ chmod a+w linux_distros.txt

$ chmod a-r linux_distros.txt
$ cat linux_distros.txt
cat: linux_distros.txt: Permission denied
```

3.4. Disk space

1. Find the size of the current folder using `du`.
2. Find how much of space is used and how much of space is available free in the system.

```
$ du -s -h
...

$ df -h
...
```

3.5. Filesystem Hierarchy Standard

Determine the location of the following file based on the Filesystem Hierarchy Standard.

1. `cat` and `ls` executable.

2. `firefox` executable.
3. C library file `libc.so.6`.
4. Icons used by the program `gimp`.
5. Kernel log file `kern.log`.

Chapter 2. Python

1. Python Lab I

1.1. Objective

Get started with Python and learn the basic types and control flow statements.

1.2. Required Reading

- Section *An Informal Introduction to Python* of the Python Tutorial [http://docs.python.org/tutorial/].
- Section *More Control Flow Tools* of the Python Tutorial [http://docs.python.org/tutorial/].

1.3. Starting Python

Start Python in interactive mode by invoking `python` in the command line.

```
$ python
Python 2.6.6 (r266:84292, Dec 27 2010, 00:02:40)
[GCC 4.4.5] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

1.4. Arithmetic expressions

Convert temperature specified in celsius to farenheit, using Python's interactive prompt.

```
>>> celsius = 100
>>> farenheit = (9 * celsius / 5) + 32
>>> print farenheit
212
>>>
```

1.5. Lists and slicing

The wind speed data (in km/hr) from a weather station, for a single day, obtained every 2 hours, starting from 12:00am: 3, 5, 3, 2, 0, 0, 5, 5, 11, 5, 10, 2.

1. Represent the data using a list.
2. Extract the wind speed at noon.
3. Extract all the wind speeds in the afternoon.
4. Extract the last measurement of the day.

```
>>> wind_speed = [3, 5, 3, 2, 0, 0, 5, 5, 11, 5, 10, 2]
>>> noon = wind_speed[6]
>>> print noon
5
>>> afternoon = wind_speed[7:12]
>>> print afternoon
[5, 11, 5, 10, 2]
>>> last = wind_speed[-1]
>>> print last
2
>>>
```

1.6. List iteration

Calculate the average wind speed of the day using the data specified in Task 3.

```
>>> wind_speed = [3, 5, 3, 2, 0, 0, 5, 5, 11, 5, 10, 0]
>>> sum = 0
>>> for item in wind_speed:
...     sum = sum + item
...
>>> avg = sum / len(wind_speed)
>>> print avg
4
>>>
```

The line should be preceded by a single TAB. The loop body is terminated by a blank line.

1.7. Emulate C's `for` statement

Print the first 10 numbers in the Fibonacci series. Use the `range()` function to emulate C's for statement: `for (i = 0; i < 10; i++)`.

```
>>> range(4)
[0, 1, 2, 3]
>>> a, b = 0, 1
>>> for i in range(10):
...     print b,
...     a, b = b, a+b
...
1 1 2 3 5 8 13 21 34 55
>>>
```

1.8. Strings and slicing

Perform the following operations, on the word "newsworthy":

1. Extract the second letter.
2. Extract the first four letters.
3. Extract the last six letters.

```
>>> word = "newsworthy"
>>> print len(word)
10
>>> print word[1]
e
>>> print word[0:4]
news
>>> print word[4:10]
worthy
>>>
```

1.9. String iteration and concatenation

Reverse the word "linux" using string iteration and concatenation.

```
>>> word = "linux"
>>> reversed = ""
>>> for ch in word:
...     reversed = ch + reversed
```

```
...
>>> print reversed
xunil
>>>
```

1.10. String multiplication

Print a 5 step right-angled triangle using string multiplication.

```
>>> max = 5
>>> for i in range(0, max):
...    print "*" * (i+1)
...
*
**
***
****
*****
>>>
```

1.11. Writing Python scripts

Write a python script to print "Hello World".

1. Create a file with the following statement.

```
print "Hello World"
```

2. Save the script as `hello.py`. Invoke the python interpreter with `hello.py` as argument.

```
$ python hello.py
Hello World
```

1.12. Using the hashbang sequence

Write a Python script that prints "Hello World", and uses the Hashbang sequence, to specify the interpreter.

1. Create a file with the following contents

```
#!/usr/bin/python

print "Hello World"
```

2. Save the script as `hello2.py`, set the execute permission using `chmod` and execute the script.

```
$ chmod +x hello2.py
$ ./hello2.py
Hello World
```

1.13. `if` statement

Write a Python script to find the largest of three numbers.

```
a = 10
b = 12
c = 15
if a > b:
    if a > c:
```

```
        print a
    else:
        print c
else:
    if b > c:
        print b
    else:
        print c
```

1.14. `break` statement and `else` clause

Write a Python script that prints prime numbers less than 20.

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print n, 'equals', x, '*', n/x
            break
    else:
        # loop fell through without finding a factor
        print n, 'is a prime number'
```

1.15. `while` loop

Write a Python script to find the first factorial that has more than 100 digits.

```
n = 1
fact = 1
while fact < (10 ** 100):
    n = n + 1
    fact = fact * n
```

2. Python Lab II

2.1. Objective

1. Learn about functions - definition, default arguments, multiple return values, variable arguments.
2. Learn Python's data structures - lists, dictionaries, and tuples, in detail.

2.2. Required Reading

- Section *More Control Flow Tools* of the Python Tutorial [http://docs.python.org/tutorial/].
- Section *Data Structures* of the Python Tutorial [http://docs.python.org/tutorial/].

2.3. Defining functions

Write a function that accepts a text and a character as argument and returns the no. of occurrences of the character in the text.

```
def count(text, ch):
    n = 0
    for t in text:
        if ch == t:
           n += 1
    return n

# Test
print count("coordination", "o")
```

2.4. Default argument values

Modify the function in the previous task, such that the caller can specify whether case should be ignored. The default is to ignore case.

```
def count(text, ch, ignore_case=True):
    if ignore_case:
        text = text.lower()
        ch = ch.lower()

    n = 0
    for t in text:
        if ch == t:
           n += 1
    return n

# Test
print count("coordinatiOn", "O", False)
print count("coordination", "O", True)
print count("coordination", "O")
```

2.5. Multiple return values

Write a function that returns the smallest and largest element in a list.

```
def min_max(numbers):
    smallest = largest = numbers[0]
    for item in numbers:
        if item > largest:
            largest = item
        elif item < smallest:
            smallest = item
    return smallest, largest

# Test
smallest, largest = min_max([1, 2, 7, 6, 3, 1, 2, 8, 4])
```

2.6. Tuples

Color is represented using three bytes, one each for red, green and blue. For the color with components red = 240, green = 50 and blue = 150.

1. Represent the color as a tuple.
2. Show that tuples are immutable.
3. Represent the color as a tuples using the alternate syntax (without parenthesis).
4. Extract the components into separate variable for red, green, and blue.

```
>>> rgb = (240, 50, 150)
>>>
>>> rgb[0] = 150
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
>>> rgb = 240, 50, 150
>>> print rgb
(240, 50, 150)
```

```
>>>
>>> r, g, b = rgb
>>> print r
240
>>> print g
50
>>> print b
150
>>>
```

2.7. Dictionaries

A table of names and telephone no. is given below.

Name	Telephone
jack	4098
sape	4139

1. Represent the table as dictionary mapping names to telephone numbers.
2. Add a new entry that maps guido to 4127.
3. Get the telephone no. of jack, from the dictionary.
4. Remove the entry for sape from the dictionary.
5. Get the names from the dictionary.
6. Get the telephone nos. from the dictionary.
7. Check if guido is in the dictionary.

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>>
>>> tel['guido'] = 4127
>>> tel == {'sape': 4139, 'guido': 4127, 'jack': 4098}
True
>>>
>>> print tel['jack']
4098
>>>
>>> tel.pop('sape')
4139
>>> tel == {'guido': 4127, 'jack': 4098}
True
>>>
>>> print sorted(tel.keys())
['guido', 'jack']
>>>
>>> print sorted(tel.values())
[4098, 4127]
>>>
>>> print 'guido' in tel
True
>>>
```

2.8. Dictionaries, looping and defaults

Write a python script to determine the occurrence of each character in a given string.

```
def count(text):
    text = text.lower()
```

```
    stat = {}
    for ch in text:
        n = stat.get(ch, 0)
        stat[ch] = n + 1
    return stat

def display(stat):
    for key in stat:
        print key, stat[key]

stat = count("Hello World")
display(stat)
```

2.9. Lock step iteration

Define a function that accepts the student names as a list in the first argument, and the student's mark as a list in the second argument and prints the name and the mark.

```
# Non-pythonic
def print_marks1(names, marks):
    for i in range(len(names)):
        print names[i], marks[i]

# Pythonic
def print_marks2(names, marks):
    for n, m in zip(names, marks):
        print n, m

names = ["alice", "bob", "charlie", "dave"]
marks = [85, 90, 95, 80]
print_marks2(names, marks)
```

2.10. Enumerated Iteration

Write a function that print a list of menu items to choose from.

```
# Non-Pythonic
def print_menu1(menu_items):
    nitems = len(menu_items):
    for i in range(nitems):
        print i+1, item[i]
    print "Select an item 1 -", nitems

# Pythonic
def print_menu2(menu_items):
    nitems = len(menu_items)
    for i, item in enumerate(menu_items):
        print i+1, item
    print "Select an item 1 -", nitems

# Test
print_menu2(["Naan", "Chappathi", "Puri", "Parotha"])
```

2.11. List Comprehension

Use list comprehension to perform the following operations on the list of words - abc, algol, c, c++, haskel, java, lisp, modula-3.

1. Create a list containing the words in title case (First letter in upper case.)
2. Create a list of words whose length is greater than 3.

```
>>> words = ["abc", "algol", "c", "c++", "haskel", "java", "lisp", "modula-3"]
>>>
>>> titles = [w.title() for w in words]
>>> print titles
['Abc', 'Algol', 'C', 'C++', 'Haskel', 'Java', 'Lisp', 'Modula-3']
>>>
>>> filtered = [w for w in words if len(w) > 3]
>>> print filtered
['algol', 'haskel', 'java', 'lisp', 'modula-3']
>>>
```

2.12. Variable Arguments

Write a function to generate an SQL insert statement, given the table name and field values. Use variable arguments to get the field values.

```
def quote(x):
    return "'" + str(x) + "'"

def sql_insert1(table, *fields):
    query = []
    query += "INSERT INTO " + table + " "
    query += "VALUES ("
    query += ", ".join([quote(field) for field in fields])
    query += ")"

    query = "".join(query)
    return query

def sql_insert2(table, **fields):
    query = []
    query += "INSERT INTO " + table + " ("
    query += ", ".join(fields.keys()),
    query += ") VALUES (",
    query += ", ".join([quote(field) for field in fields.values()])
    query += ")"

    query = "".join(query)
    return query

# Test
print sql_insert1("order", "PO001", "22/7/2011", 2000.00)
print sql_insert2("order", code="PO001", date="22/7/2011", total=2000.00)
```

3. Python Lab III

3.1. Objective

1. Learn about modules, imports, listing module contents, standard modules.
2. Learn to build simple Python apps.

3.2. Required Reading

- Section *Modules* of the Python Tutorial [http://docs.python.org/tutorial/].

3.3. Modules

1. Write a module with the following contents, and name it `fibo.py`.

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b

def fib2(n): # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

2. Import the module.
3. Access the function `fib` defined within the module.
4. Access and print the module's name.
5. Assign a local name for the function `fibo.fib`, and invoke the function using local name.
6. Import the function `fibo.fib` directly into the local name `fib`.

```
>>> import fibo

>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

>>> print fibo.__name__
fibo

>>> myfib = fibo.fib
>>> myfib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377

>>> from fibo import fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
>>>
```

3.4. Builtins and Standard Modules

Perform the following operations at the Python interpreter prompt.

1. Import the `random` module.
2. List the contents of the `random` module.
3. Print the help information for `random.randint`.
4. Invoke the `random.randint` function.
5. List the builtin functions.
6. Create a variable and a function and list the contents of the local name space.

```
>>> import random
```

```
>>> dir(random)
[...'randint', 'random', 'randrange'...]

>>> help(random.randint)
Help on method randint in module random:
<BLANKLINE>
randint(self, a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
<BLANKLINE>

>>> random.randint(1, 99)
57

>>> dir(__builtins__)
['ArithmeticError'...'zip']

>>> def testfunc():
...     pass
...
>>> testvar = 10
>>> dir()
['__builtins__'...'testfunc', 'testvar']
>>>
```

3.5. Arithmetic Quiz

In the Arithmetic Quiz program, the computer prints an arithmetic expression using the two 2 digit numbers (selected in random) and the operator + or - (selected in random). The user has to find out the value of the expression. Based on the user's input the computer updates the user's score. After 20 repetitions the computer prints the user's score. A sample run is shown below.

```
10 + 15 ? 25
Correct!
21 + 33 ? 55
Wrong!
31 - 10 ? 21
Correct!
...
...
Your score is 18/20.
```

Write a Python script that implements the Arithmetic Quiz.

4. Python Lab IV

4.1. Objective

1. Learn about string formatting and file I/O.
2. Learn about errors, exceptions and exception handling.

4.2. Required Reading

- Section *Input and Output* of the Python Tutorial [http://docs.python.org/tutorial/].
- Section *Errors and Exceptions* of the Python Tutorial [http://docs.python.org/tutorial/].

4.3. String Formatting

Write a python script to print the current date in the following format.

```
Wed May 25 02:26:23 IST 2011
```

The function `time.localtime` returns the time as object, with the following members.

`tm_year`	Year
`tm_mon`	Month, range [1,12], January is 1
`tm_mday`	Day of the month
`tm_hour`	Hours in 24hr format
`tm_min`	Minutes
`tm_sec`	Seconds
`tm_wday`	Day of the week, range [0,6], Monday is 0
`tm_yday`	Day of the year

```
from time import localtime

week = [ "Mon", "Tue", "Wed", "Thu", "Fri", "Sun" ]
year = [ "Jan", "Feb", "Mar", "Apr", "May", "Jun",
         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" ]

def print_date():
    t = localtime()
    fmt = "%s %s %d %02d:%02d:%02d IST %d"
    day = week[t.tm_wday]
    mon = year[t.tm_mon - 1]
    print  fmt % (day, mon, t.tm_mday, t.tm_hour, t.tm_min,
                  t.tm_sec, t.tm_year)

print_date()
```

4.4. Reading Text Files

Write a function that prints the contents of a file in uppercase. The function should accept the filename as argument.

```
def upper(filename):
    infile = open(filename)
    for line in infile:
        print line.upper(),
    infile.close()

upper("test.txt")
```

4.5. Writing to Text Files

Write a function that sorts that contents of the file. The function should accept the filename as argument.

```
def sort(filename):
    infile = open(filename)
    lines = infile.readlines()
    infile.close()

    lines.sort()

    outfile = open(filename, "w")
```

```
    outfile.writelines(lines)
    outfile.close()

sort("test.txt")
```

4.6. Errors and Exceptions

Generate the following errors/exceptions, at the Python prompt.

1. Generate a syntax error.
2. Generate a ZeroDivisionError.
3. Generate a NameError.
4. Generate a TypeError.

```
>>> l = [1, 2, 3]]
  File "<stdin>", line 1
    l = [1, 2, 3]]
                 ^
SyntaxError: invalid syntax

>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ZeroDivisionError: integer division or modulo by zero

>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'spam' is not defined

>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: cannot concatenate 'str' and 'int' objects
```

4.7. Handling Exceptions

Write a function that inputs an integer from the user. If the user input is not a valid no. the function should ask the user to re-enter the no.

```
def input_num(msg):
    while True:
        try:
            num = raw_input(msg)
            num = int(num)
            return num
        except ValueError:
            print "Invalid integer", num

print input_num("Enter a number: ")
```

4.8. Capital Quiz

Write a python script that quizzes the user about countries and their capitals. The program maintans a list of countries and their capitals in a file called `capitals.txt`. The program prints a country name in random and the user is supposed to enter the country's capital. This goes on till all the countries in the list are exhausted.

Each line in `capitals.txt`, is in the format

```
country:capital
```

An example file is shown below.

```
Australia:Canberra
Bangladesh:Dhaka
Bhutan:Thimphu
Egypt:Cairo
India:New Delhi
Indonesia:Jakarta
Japan:Tokyo
Nepal:Kathmandu
Pakistan:Islamabad
Sri Lanka:Kotte
Thailand:Bangkok
China:Beijing
```

The program should be print an appropriate error message, if `capitals.txt` is malformed, or error occurs while opening the file.

```
import sys
import random

def open_capitals():
    try:
        # Throws IOError, if open fails.
        return open("capitals.txt")
    except IOError as e:
        print "error opening capitals.txt: %s" % e.strerror
        sys.exit(1)

def parse_capitals(cfile):
    capitals = {}
    for i, line in enumerate(cfile):
        try:
            # Throws ValueError, if no. of elements in LHS and RHS
            # does not match.
            country, capital = line.split(":")
            country = country.strip()
            capital = capital.strip()
            capitals[country] = capital
        except ValueError:
            print "error parsing line %d" % (i+1)
            sys.exit(1)

    return capitals

def quiz():
    cfile = open_capitals()
    capitals = parse_capitals(cfile)
    country_list = capitals.keys()

    while len(country_list) > 0:
        country = random.choice(country_list)
        capital = capitals[country]
        country_list.remove(country)
```

```
        user_input = raw_input("%s ? " % country)
        if user_input.upper() != capital.upper():
            print "Wrong, the answer is '%s'." % capital

if __name__ == "__main__":
    quiz()
```

5. Python Lab V

5.1. Objective

1. Learn to access MySQL databases from Python.
2. Learn the Decimal datatype.

5.2. Required Reading

- Section *Decimal Floating Point Arithmetic* of the Python Tutorial [http://docs.python.org/tutorial/].
- Writing MySQL Scripts with Python DB-API [http://www.kitebird.com/articles/pydbapi.html].

5.3. Accounts Database

Wrie a Python script to create a populate a customer account balance database. The database consists of records of 3 fields:

1. Account ID
2. Account Name
3. Account Balance

Solution. Make sure you know the username, password and database on your MySQL server. Create the file `conf.py` with the following contents. Replace your server hostname, username, password and database name.

Listing 2.1. `conf.py`

```
server = "your-server-name-here"
username = "your-username-here"
password = "your-password-here"
database = "your-databaes-here"
```

```
import MySQLdb as db
import sys
import conf

conn = db.connect("shark", conf.username, conf.password, conf.database)
cursor = conn.cursor()
cursor.execute("DROP TABLE IF EXISTS accounts")

# Create Tables
sql = """CREATE TABLE accounts (id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
                                name VARCHAR(50),
                                balance DECIMAL(10, 2))"""
cursor.execute(sql)

# Insert into Tables
sql = """INSERT INTO accounts (name, balance) VALUES ('Alice', 500.00),
                                                     ('Bob', 50000.00),
                                                     ('Charlie', 100.00)"""
```

```
cursor.execute(sql)
```

5.4. Displaying Account Balance

Write Python script to display the current balance in the accounts database.

5.5. Decimal Datatype

Demonstrate the following properties of the Decimal datatype.

1. Exact representation.
2. Ability to test for equality
3. User alterable precision.

```
>>> from decimal import *
>>> 1 + 0.1
1.1000000000000001
>>> 1 + Decimal(0.1)
Decimal(1.1)

>>> val = 0
>>> for i in range(10):
...     val += 1.1
>>> print val == 11
False
>>> val = 0
>>> for i in range(10):
...     val += Decimal(1.1)
>>> print val == 11
True

>>> getcontext().prec = 4
>>> Decimal("1") / Decimal("3")
Decimal('0.3333')
>>> getcontext().prec = 28
>>> Decimal("1") / Decimal("3")
Decimal('0.3333333333333333333333333333')
```

5.6. Updating Account Balance

Accounting transactions are used to update the account balance. Each transaction consists of the account no. and the amount. The amount can be positive or negative depending upon whether the transaction is credit or debit.

Write a Python program that takes in transactions as input, updates the database.

```
import MySQLdb as db
import conf
from decimal import Decimal

def input_num(msg, numtype):
    while True:
        try:
            num = raw_input(msg)
            num = numtype(num)
            return num
        except ValueError, e:
            print "Invalid number."
```

```
def get_balance(accno):
    cursor.execute("SELECT balance FROM accounts WHERE id = %d" % accno)
    row = cursor.fetchone()
    if row == None:
        return 0
    else:
        return row[0]

def set_balance(accno, balance):
    cursor.execute("UPDATE accounts SET balance = %s WHERE id = %d" %
                   (balance, accno))

def transact():
    while True:
        accno = input_num("Enter Account No.: ", int)
        if accno == -1: break
        balance = get_balance(accno)
        print "The current balance is", balance
        amount = input_num("Enter amount to credit/debit: ", Decimal)
        set_balance(accno, balance + amount)
        print "The new balance is", get_balance(accno)

def main():
    global cursor
    conn = db.connect(conf.server, conf.username,
                      conf.password, conf.database)
    cursor = conn.cursor()
    transact()

if __name__ == "__main__":
    main()
```

6. Python Lab VI

6.1. Objective

1. Learn Object Oriented Programming in Python.
2. Learn operator overloading and inheritance.

6.2. Required Reading

- Chapter 13, 14, 15, 16 and 17 of the book How to Think Like a Computer Scientist [http://openbookproject.net/thinkcs/python/english2e/].
- Section *HTMLParser - Simple HTML and XHTML parser* of The Python Standard Library [http://docs.python.org/library/].

6.3. Classes and Objects

Write a `Time` class that has three fields: hours, minutes and seconds. The class should have the following methods.

1. Initialization method.
2. `print_time` - that prints the time.
3. `increment` - that increments the time.

```
class Time:
    def __init__(self, hours=0, minutes=0, seconds=0):
```

```
        self.hours = hours
        self.minutes = minutes
        self.seconds = seconds

    def print_time(self):
        print (str(self.hours) + ":" +
               str(self.minutes) + ":" +
               str(self.seconds))

    def increment(self, seconds):
        self.seconds = seconds + self.seconds

        while self.seconds >= 60:
            self.seconds = self.seconds - 60
            self.minutes = self.minutes + 1

        while self.minutes >= 60:
            self.minutes = self.minutes - 60
            self.hours = self.hours + 1

# Test
time = Time(10, 0, 0)
time.print_time()
time.increment(60)
time.print_time()
```

6.4. Operator Overloading

Write a `Point` class that has two fields: `x` and `y`. The class should have the following methods. The `Point` objects should overload + to support addition and should overload * to support dot product.

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

    def __mul__(self, other):
        return self.x * other.x + self.y * other.y

    def __str__(self):
        return "(%d, %d)" % (self.x, self.y)

# Test
p1 = Point(1, 2)
p2 = Point(3, 4)
print p1, p2
print p1 + p2
print p1 * p2
```

6.5. Inheritance

Write a HTML parser that counts the occurrence of each tag in a HTML page, and has a `stats` method to print the statistics.

```
from HTMLParser import HTMLParser
```

```
class MyParser(HTMLParser):
    def __init__(self):
        HTMLParser.__init__(self);
        self.counter = {}

    def handle_starttag(self, tag, attrs):
        val = self.counter.get(tag, 0)
        self.counter[tag] = val + 1

    def stats(self):
        for key in self.counter:
            print key, self.counter[key]

parser = MyParser()

parser.feed("""<html>
<body>
<ol>
<li>Hello</li>
<li>World</li>
<li>Linux</li>
</body>
</html>""")

parser.stats()
```

Chapter 3. GUI Programming

1. GUI Programming Lab I

1.1. Objective

1. Get started with GTK+.
2. Learn to create widgets and associate callbacks.

1.2. Required Reading

- *Getting Started* and *Movin On* sections of the PyGTK 2.0 Tutorial [http://www.pygtk.org/pygtk2tutorial/index.html]
- *ComboBox Widget* section of the PyGTK 2.0 Tutorial [http://www.pygtk.org/pygtk2tutorial/index.html]

1.3. Hello World

Write a Python script that creates a button with the text "Hello World".

```
import pygtk
pygtk.require('2.0')
import gtk

class HelloWorld:
        def __init__(self):
                self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
                self.window.set_border_width(5)
                self.button = gtk.Button("Hello World!")
                self.window.add(self.button)
                self.window.show_all()

        def main(self):
                gtk.main()

if __name__ == "__main__":
        hello = HelloWorld()
        hello.main()
```

1.4. Callbacks

Extend the previous script, such that "Hello World" is printed when the user click on the button.

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

import pygtk
pygtk.require('2.0')
import gtk

class HelloWorld:
        def on_button_clicked(self, widget, data=None):
                print "Hello World"

        def on_window_destroy(self, widget, data=None):
                gtk.main_quit()
```

```
        def __init__(self):
                self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
                self.window.connect("destroy", self.on_window_destroy)
                self.window.set_border_width(5)

                self.button = gtk.Button("Hello World")
                self.button.connect("clicked", self.on_button_clicked)
                self.window.add(self.button)

                self.window.show_all()

        def main(self):
                gtk.main()

if __name__ == "__main__":
        hello = HelloWorld()
        hello.main()
```

1.5. More Signals

Write a Python script that creates a combo-box with three elements. When the selection is changed the selected item is to be printed.

Tip: The `gtk.combo_box_new_text()` helper function creates a combo-box. The `append_text()` method can be used to add entries to the combo-box. The combo-box emits the `changed` signal when the combo-box selection changes.

2. GUI Programming Lab II

2.1. Objective

1. Learn about packing using boxes.
2. Learn about packing using tables.

2.2. Required Reading

- *Packing Widgets* section in the PyGTK 2.0 Tutorial [http://www.pygtk.org/pygtk2tutorial/index.html]
- *Text Entries* section in the PyGTK 2.0 Tutorial [http://www.pygtk.org/pygtk2tutorial/index.html]
- *TextView Overview* and *TextViews* section in the PyGTK 2.0 Tutorial [http://www.pygtk.org/pygtk2tutorial/index.html]
- *pango.FontDescription* section in the PyGTK 2.0 Reference Manual [http://developer.gnome.org/pygtk/stable/]

2.3. Packing Using Boxes

Write a program that displays the following system information from `/proc` files.

1. CPU Information `/proc/cpuinfo`
2. Memory Usage Information `/proc/meminfo`
3. Interrupt Information `/proc/interrupts`

The program window should have two panels. The panel on the left should have buttons to select one of the above specified information. The panel on the right should have `TextView` that displays the selected information. When window is scaled only the `TextView` should expand and fill to occupy the available space.

```
import pygtk
```

```
pygtk.require('2.0')
import gtk
import pango

class SysInfo:
    def on_button_clicked(self, button, filename):
        text = open("/proc/" + filename).read()
        self.info_view.get_buffer().set_text(text)

    def on_window_destroy(self, window):
        gtk.main_quit();

    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.connect("destroy", self.on_window_destroy)
        self.window.set_border_width(5)

        self.main_hbox = gtk.HBox()
        self.window.add(self.main_hbox)

        self.button_vbox = gtk.VBox()
        self.main_hbox.pack_start(self.button_vbox, expand=False)

        filename_list = ["cpuinfo", "meminfo", "interrupts"]
        for filename in filename_list:
            button = gtk.Button(filename)
            self.button_vbox.pack_start(button, expand=False)
            button.connect("clicked", self.on_button_clicked, filename)

        self.scroll_win = gtk.ScrolledWindow()
        self.main_hbox.pack_start(self.scroll_win, expand=True, fill=True)
        self.info_view = gtk.TextView()
        self.scroll_win.add(self.info_view)

        self.info_view.set_editable(False)
        self.info_view.modify_font(pango.FontDescription("courier"))

        self.window.show_all()

    def main(self):
        gtk.main()

if __name__ == "__main__":
    sysinfo = SysInfo()
    sysinfo.main()
```

2.4. Packing Using Tables

Write a program that converts from currency from USD to INR. Use tables to pack the label and entry widgets.

```
import pygtk
pygtk.require('2.0')
import gtk

class Conversion:
    def on_convert_clicked(self,widget):
        usd = self.usd_entry.get_text()
```

```
        usd = int(usd)
        inr = usd * 44.75
        self.inr_entry.set_text(str(inr))

    def on_window_destroy(self, widget, data=None):
        gtk.main_quit()

    def __init__(self):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.connect("destroy", self.on_window_destroy)
        self.window.set_border_width(5)

        table = gtk.Table(4, 2, homogeneous=False)
        self.window.add(table)

        ef = gtk.EXPAND | gtk.FILL

        self.usd_label = gtk.Label("Enter USD Amount: ")
        table.attach(self.usd_label, 0, 1, 0, 1, 0, 0, 5, 5)
        self.usd_entry = gtk.Entry(0)
        table.attach(self.usd_entry, 1, 2, 0, 1, ef, 0, 5, 5)

        self.inr_label = gtk.Label("Indian Value Is (INR): ")
        table.attach(self.inr_label, 0, 1, 1, 2, 0, 0, 5, 5)
        self.inr_entry = gtk.Entry(0)
        table.attach(self.inr_entry, 1, 2, 1, 2, ef, 0, 5, 5)

        self.convert_button = gtk.Button("Convert")
        self.convert_button.connect("clicked", self.on_convert_clicked)
        table.attach(self.convert_button, 0, 2, 3, 4, ef, 0, 5, 5)

        self.window.show_all()

    def main(self):
        gtk.main()

if __name__ == "__main__":
    conversion = Conversion()
    conversion.main()
```

3. GUI Programming Lab III & IV

3.1. Objective

Learn to write simple applicatons using the GTK+ toolkit.

3.2. Required Reading

Images section in the PyGTK 2.0 Tutorial [http://www.pygtk.org/pygtk2tutorial/index.html]

3.3. Image Viewer

Write a simple image viewer application. The application should have three widgets:

1. An `Image` widget to display the image.
2. An `Entry` widget to enter the filename.
3. A `Button` widget when clicked will set the image file on the `Image` widget.

Chapter 4. Virtualization

1. Virtualization Lab I

1.1. Objective

Learn how to install and run a guest OS within Qemu.

1.2. Required Reading

- QEMU article on Wikipedia [http://en.wikipedia.org/wiki/QEMU].
- Section *Creating and Manipulating Images* of the QEMU Wikibook [http://en.wikibooks.org/wiki/QEMU].

1.3. Booting FreeDOS

Step 1: Getting the disk image. The contents of the emulated hard disk is stored in a file. When the guest system accesses the hard disk, the data is stored and retrieved from the file. Download the disk image containing a FreeDOS installation from http://foss-lab-manual.googlecode.com/files/freedos.qcow2

```
[host]$ wget http://foss-lab-manual.googlecode.com/files/freedos.qcow2
```

Step 2: Booting the disk image. Start Qemu by specifying the FreeDOS disk image using the `-hda` option.

```
[host]$ qemu -hda freedos.qcow2
```

Step 3: Switching between host and guest. When the guest window is clicked, Qemu enters the "Grab mode". In Grab mode, all keystrokes are sent to the guest. To exit Grab mode, and give back control to the host press and release the `Ctrl-Alt` key combination.

Step 4: Shutting down the guest. After using the guest, the guest can shutdown like a normal system, by issuing the OS' own shutdown command. In the case of FreeDOS issue the `halt` command.

```
C:\> halt
```

1.4. Installing FreeDOS

Step 1: Creating the disk image. The disk image into which FreeDOS will be installed is created first. A 100MB disk image should be sufficient for the FreeDOS installation. The `qcow2` format will be used, because of its size advantages.

Create the disk image using the `qemu-img` command. The `create` sub-command is to create hard disk images. The format is specified using the `-f qcow2` option. The hard disk image file to be created is specified as the first argument. The hard disk image size is specified as the second argument.

```
[host]$ qemu-img create -f qcow2 hd.qcow2 100M
```

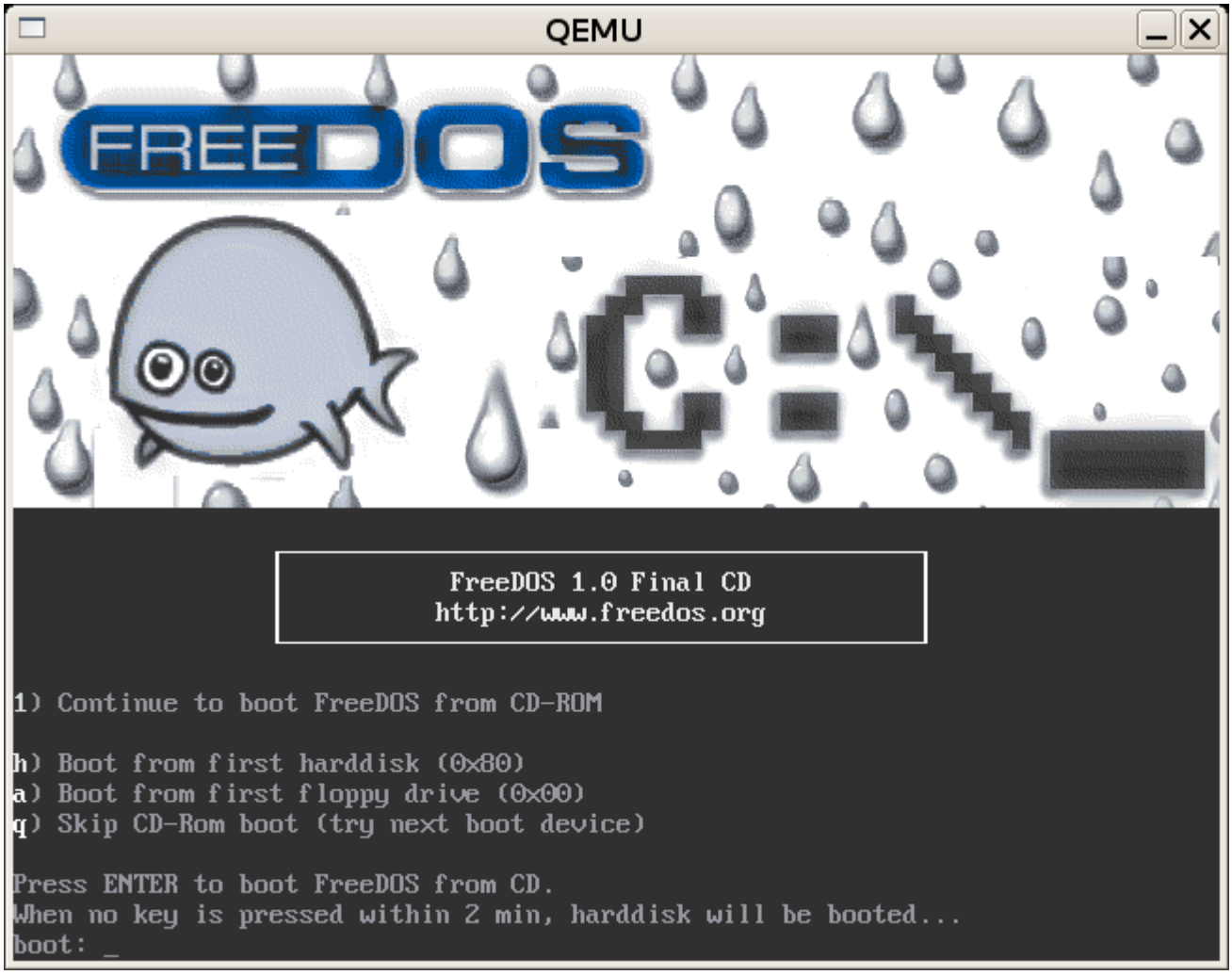
Step 2: Get the FreeDOS installation CD. Download the FreeDOS installation CD available from http://foss-lab-manual.googlecode.com/files/fdbasecd.iso

```
[host]$ wget http://foss-lab-manual.googlecode.com/files/fdbasecd.iso
```

Step 3: Boot the FreeDOS installation CD. The guest is booted from the FreeDOS installation CD and the installer installs FreeDOS on to the hard disk. Start Qemu by specifying the hard disk image

using `-hda` option, the CD-ROM image using `-cdrom` option. Now, since there are two bootable media - hard disk and CD-ROM, the CD-ROM drive is specified as the boot media using `-boot d` option. In the boot option, the hard disk is represented using the drive letter `c` and the CD-ROM is represented using the drive letter `d`.

```
[host]$ qemu -hda hd.qcow2 -cdrom fdbasecd.iso -boot d
```



1. In the installation CD boot prompt, press `Enter` to continue booting the installer.
2. In the installer menu select `Install to harddisk using FreeDOS SETUP`, and press `Enter`.
3. Select the language and keyboard layout as `English (US)`.

Step 4: Partition the hard disk. The created hard disk image is like a blank hard disk, and does not contain partitions. Atleast one partition has to be created to install the OS.

1. Select `Prepare the harddisk for FreeDOS by running XFdisk`, to start the partitioner.
2. Select the un-partitioned free space, and press `Enter`, to open the `Options` menu.
3. From the `Options` menu, select `New Partition` and press `Enter`.
4. From the `New Partition` sub-menu, select `Primary Partition` and press `Enter`.
5. Specify the partition size as 100 and press `Enter`, to create a partition of size 100MB.
6. Select YES in the `Initialise Partition Area` dialog box, to format the created partition.
7. Select YES in the `Initialise Partition Area` dialog box, that appears again. For some reason unknown to the author, this is asked twice.

8. Press F3 to quit the partitioner.
9. Select YES in the `Write Partition Table` dialog box, to write the partition table to disk.
10. Select YES in the `Restart Computer` dialog box. If after selecting the option the guest does not restart. Close Qemu, and restart Qemu with same command as before.
11. When the system boots back, follow the 3 instructions in step 3, to reach the installer menu.
12. Select `Continue with FreeDOS installation`, to resume the installer.

Step 5: Install FreeDOS

1. Select `Start installation of FreeDOS`, to start the installation.
2. Make sure you read the copyright notice and press any key.
3. The installer prompts for the OS install path with `C:\FDOS` as default. Press Enter to accept the default.
4. The installer prompts for the programs to be installed. Check/uncheck programs as required using the `Space` key. Press `Enter` when done.
5. The installer installs the OS and the selected programs.
6. Enter Y when asked if the system can be rebooted. If the guest does not restart, close the Qemu window, and invoke Qemu by specifying the hard disk image alone.

```
[host]$ qemu -hda hd.qcow2
```

1.5. Saving and Restoring Guest State

Step 1: Boot FreeDOS. First boot into the guest system, using the steps described in section Section 1.3, “Booting FreeDOS”.

Step 2: Save the guest state. The current state of the guest system can be saved using the `savevm` command in the monitor interface.

1. Switch to the monitor interface by pressing `Ctrl-Alt-2`.
2. Save the guest state using the `savevm` command, with the tag `booted`.

```
(qemu) savevm booted
```

Step 3: Restore the guest state. The guest state can be restored using the `loadvm` monitor command.

1. Switch back to the console using `Ctrl-Alt-1`.
2. Make some changes to the filesystem by creating files.

```
C:\> echo hello > file.txt
```

3. Go back to the monitor interface and restore the guest state using the `loadvm` command.

```
(qemu) loadvm booted
```

4. Go back to the console and check for the file `file.txt`.

Step 4: Start from saved guest state.

1. Shutdown the guest system using the `halt` command.
2. Start the guest system using the saved guest state, by specifying the `-loadvm` option to `qemu`. The option accepts the tag as argument.

```
[host]$ qemu -hda hd.qcow2 -loadvm booted
```

2. Virtualization Lab II

2.1. Objective

Learn how to share and transfer files between the host and the guest.

2.2. Required Reading

- Section *Creating and Manipulating Images* of the QEMU Wikibook [http://en.wikibooks.org/wiki/QEMU].

2.3. Transferring Files

Step 1: Boot the Linux guest system. Boot the Linux guest with disk image from http://foss-lab-manual.googlecode.com/files/linux.qcow2

Step 2: Transfer files using scp. Qemu simulates a network between the host and the guest. The host has a network address of `10.0.2.2`. Files can be transferred from the host to the guest and vice-versa using the `scp` command. The `/etc/passwd` file on the host can be transferred to the guest using the following command in the guest.

```
[guest]$ scp user@10.0.2.2:/etc/passwd .
```

Replace `user` with your user-name on the host system.

Step 3: Mount a remote folder using SSH Filesystem. Though files can be transferred using `scp`, it would be more convenient if a host folder can be shared between the host and the guest. The FUSE based SSH Filesystem, can be used to mount a remote folder locally. The home folder in the host can be mounted in the guest using the following command sequence.

```
[guest]$ mkdir host-home
[guest]$ sshfs user@10.0.2.2: host-home
```

The host files are visible in the guest and all changes made in the guest are immediately reflected in the host.

2.4. Modifying Raw Images

Step 1: Ensure guest is not booted. Get the disk image from http://foss-lab-manual.googlecode.com/files/freedos.img. Disk images should not be modified when they are in use by a guest system. Make sure there are no guest systems using the disk image.

Step 2: Determine the partition offset. The disk image usually has partitions within it. Determine the partition offsets using the `fdisk` command. Specify the disk image as argument. Specify the `-l` option to list the partitions. Specify the `-u` option to specify the partition offsets in sectors.

```
[host]$ /sbin/fdisk -l -u freedos.img
You must set cylinders.
You can do this from the extra functions menu.

Disk hd.img: 0 MB, 0 bytes
16 heads, 63 sectors/track, 0 cylinders, total 0 sectors
Units = sectors of 1 * 512 = 512 bytes

 Device Boot      Start         End      Blocks   Id  System
hd.img1   *          63      204623      102280+   e  W95 FAT16 (LBA)
```

The offset of the partition in bytes is specified in sectors. Calculate the byte offset by multiplying the offset by the size of a sector (512 bytes). `63 * 512 = 32256`.

Step 3: Associate the partition to a loopback device. Associate a partition in the disk image to a loopback device using the `losetup` command. Specify the loopback device file and the raw disk image file as arguments. Specify the partition offset using `--offset` option. The option accepts the offset in bytes.

```
[host]$ losetup --offset=32256 /dev/loop0 freedos.img
```

Step 4: Mount the loopback device. Create a mount point and mount the loopback device using the mount command.

```
[host]$ mkdir raw-contents
[host]$ mount /dev/loop0 raw-contents
```

2.5. Modifying Qcow2 Images

Step 1: Ensure guest is not booted. Get the disk image from http://foss-lab-manual.googlecode.com/files/freedos.qcow2. Disk images should not be modified when they are in use by a guest system. Make sure there are no guest systems using the disk image.

Step 2: Serve the image as a block device. Expose a partition in the `qcow2` image as a block device using the `qemu-nbd` command. Specify the disk image as argument to `qemu-nbd`. Specify the partition to be exposed using the `--partition` option. The option accepts the partition no. as argument. Specify the port to listen on using the `--port` option. The option accepts the port no. as argument. The default port is 1024.

```
[host]$ qemu-nbd --persistent --partition=1 linux.qcow2 &
```

Step 3: Use the exposed block device. Associate the exposed block device with a Network Block Device using the `nbd-client`. Pass the server host, server port and an nbd device file as argument to `nbd-client`.

```
[host]$ nbd-client localhost 1024 /dev/nbd0
```

Step 4: Mount the block device. Create a mount point and mount the network block device using the `mount` command.

```
[host]$ mkdir qcow2-contents
[host]$ mount /dev/nbd0 qcow2-contents
```

Chapter 5. Credits

1. Background

The FOSS Lab was introduced as part of the Anna University syllabus, in 2010. Though it was a great attempt at introducing Free and Open Source software to engineering students, it had its own shortcomings. One of the problems was the lack of a lab manual, for the syllabus. The Chennai Linux User Group (ILUGC), a not-for-profit organization, decided to fill in the gap. A team of volunteers from ILUGC stepped up to create the lab manual, and the FOSS Lab Manual project was born.

2. People

The following is a not so complete list of people who have contributed to the FOSS Lab Manual project.

- Allen B. Downey - Python OOP examples were adapted from his book "Think Python: How to Think Like a Computer Scientist"
- Arun S.A.G., `sagarun@gmail.com` - wrote the Packages section.
- Balakrishnan, `balakpm101@gmail.com` - contributed PyGTK+ and PyQt examples.
- Dhilip, `dhilip.jec@gmail.com` - wrote the initial Packages and Perl lab sections.
- Mohan, `mohan43u@gmail.com` - wrote the initial Compiling From Source section.
- Satyaakam Goswami, `satyaakam@gmail.com` - wrote the initial Packages section.
- Shrinivasan T., `tshrinivasan@gmail.com` - wrote the initial Getting Started section.
- Vignesh Kumar B., `vignesh@bravetux.com` - wrote the initial Kernel session.
- Vignesh Nanda Kumar, `viky.nandha@gmail.com` - wrote the initial PHP section.
- Vijay Kumar B., `vijaykumar@bravegnu.org` - wrote the Python, Virtualization, and GUI Programming sections. Wrote the guidelines for the overall layout and structure of the manual.
- Zilogic Systems, `code@zilogic.com` - contributed the examples from the Using GNU/Linux training programme. Contributed the stylesheets for generating the PDF.

3. Contributing

The FOSS Lab Manual is a work in progress. Contributions in any form - reviews, feedbacks and bug fixes are welcome. The project is hosted at http://code.google.com/p/foss-lab-manual The project mailing list is at http://groups.google.com/group/foss-lab-manual-dev