VIMAL JYOTHI
ENGINEERING COLLEGE
Affiliated to APJ Abdul Kalam Technological University &
Kannur University | Approved by AICTE
Under the Archdiocese of Thalassery

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

CS232 -Free and Open Source Software Lab

Prepared By.....

VIMAL JYOTHI
ENGINEERING COLLEGE
JYOTHI NAGAR, CHEMPERI – 670632, KANNUR D.T., KERALA
An ISO 9001:2008 Cetified Institution

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE DEPARTMENT

To contribute to the society through excellence in scientific and knowledge-based education utilizing the potential of computer science and engineering with a deep passion for wisdom, culture and values.

MISSION OF THE DEPARTMENT

- To promote all-round growth of an individual by creating futuristic environment that fosters critical thinking, dynamism and innovation to transform them into globally competitive professionals.
- To undertake collaborative projects which offer opportunities for long-term interaction with academia and industry.
- To develop human potential to its fullest extent so that intellectually capable and optimistic leaders can emerge in a range of professions.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

I. Graduates will achieve broad and in-depth knowledge of Computer Science and Engineering relating to industrial practices and research to analyze the practical problems and think creatively to generate innovative solutions using appropriate technologies.

II. Graduates will make valid judgment, synthesize information from a range of sources and communicate them in sound ways appropriate to their discipline.

III. Graduates will sustain intellectual curiosity and pursue life-long learning not only in areas that are relevant to Computer Science and Engineering, but also that are important to society.

IV. Graduates will adapt to different roles and demonstrate leaderships in global working environment by respecting diversity, professionalism and ethical practices.

PROGRAMME OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering Fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/ Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct Investigations of Complex Problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The Engineer and Society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project Management and Finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.-

PROGRAM SPECIFIC OUTCOMES (PSOs)

1. An ability to apply development principles to analyze and design complex software and systems containing hardware and software components of varying complexity.

2. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the trade-offs involved in design choices.

SYLLABUS

Course No.	Course Name	L-T-P – Credits	Year of Introduction
CS232	Free and Open Source Software Lab	0-0-3-1	2016

List of Exercises/Experiments : (Minimum 12 exercises/experiments are mandatory)

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

2. Linux commands for redirection, pipes, filters, job control, file ownership, file permissions, links and file system hierarchy.

3. Shell Programming : Write shell script to show various system configuration like
   - Currently logged user and his logname
   - Your current shell
   - Your home directory
   - Your operating system type
   - Your current path setting
   - Your current working directory
   - Show Currently logged number of users

4. Write shell script to show various system configuration like
   - About your OS and version, release number, kernel version
   - Show all available shells
   - Show mouse settings
   - Show computer CPU information like processor type, speed etc
   - Show memory information
   - Show hard disk information like size of hard-disk, cache memory, model etc
   - File system (Mounted)

5. Shell script program for scientific calculator.

6. Write a script called addnames that is to be called as follows, where classlist is the name of the classlist file, and username is a particular student's username.

./addnamesclasslistusername
The script should
   - check that the correct number of arguments was received and print an usage message if not.
   - check whether the classlist file exists and print an error message if not,
   - check whether the username is already in the file, and then either
   - print a message stating that the name already existed, or
   - add the name to the end of the list.

7. Version Control System setup and usage using GIT.

- Creating a repository
- Checking out a repository
- Adding content to the repository
- Committing the data to a repository
- Updating the local copy
- Comparing different revisions
- Revert
- Conflicts and Solving a conflict

8. Text processing and regular expression with Perl, Awk: simple programs, connecting with database e.g., MariaDB

9. Shell script to implement a script which kills every process which uses more than a specified value of memory or CPU and is run upon system start.

10. GUI programming : Create scientific calculator – using Gambas or try using GTK or QT

11. Running PHP : simple applications like login forms after setting up a LAMP stack

12. Advanced linux commands curl, wget, ftp, ssh and grep

13. Application deployment on a cloud-based LAMP stack/server with PHP eg: Openshift, Linode etc.

14. Kernel configuration, compilation and installation : Download / access the latest kernel source code from kernel.org, compile the kernel and install it in the local system. Try to view the source code of the kernel

15. Virtualisation environment (e.g., xen, kqemu, virtualbox or lguest) to test an applications, new kernels and isolate applications. It could also be used to expose students to other alternate OSs like *BSD

16. Compiling from source : learn about the various build systems used like the auto* family, cmake, ant etc. instead of just running the commands. This could involve the full process like fetching from a cvs and also include autoconf, automake etc.,

17. Introduction to packet management system : Given a set of RPM or DEB, how to build and maintain, serve packages over http or ftp. and also how do you configure client systems to access the package repository

.

18. Installing various software packages. Either the package is yet to be installed or an older version is existing. The student can practice installing the latest version. Of course, this might need Internet access.

- Install samba and share files to windows
- Install Common Unix Printing System(CUPS)

List of Experiments		
Sl No	**Experiment**	**Page**
1	Familiarization with Linux basic commands and directory structure, execute file, directory operations.	
2	Understanding Linux commands for redirection, pipes, filters, job control, file ownership, file permissions, links.	
3	Shell programming	
4	Write shell script to show various system configurations	
5	Shell script program for scientific calculator.	
6	Configure a web server and create applications like login forms using PHP	
7	Version Control System setup and usage using GIT	
8	Set up a local repository and install software from configured repository.	
9	Design a script which kills every process which uses more than a specified value of memory or CPU and is run upon system start.	
10	Installing various software packages .	
11	Understanding advanced linux commands curl, wget, ftp, ssh and grep	
12	Set up a virtual box and install windows within Ubuntu system as an application.	

Program #: 1

OBJECTIVE

To familiarize with Linux basic commands and directory structure, execute file, directory operations.

clear	-> clear the screen
ls	-> list content
ls -l	-> list content in long listing format
ls -al	-> list all subcontent in long listing format
ll	-> an alias for the above
ls -R	-> list content recursively
l.	-> list hidden files
ls -F	-> list content and classify them
alias	-> display all aliases for current user
alias <statement>	-> make alias eg alias c='clear'
unalias <alias>	-> remove alias eg unalias c
exit	-> log out from the system
logout	- diito -
^d	- ditto -
tree	-> list content in a tree (hierarchial) diagram
tree -d	-> list subdirectories only - no files
tree -p	-> list content with their permissions
cd <directory>	-> change directory to...
cd ..	-> change to parent directory
cd -	-> change to previous directory
cd	-> change to home directory
cd ~	-^
pwd	-> print work (current) directory
pwd -P	-> print parent working dir of this symlink dir
mkdir <directory>	-> make directory
mkdir -p <directory>	-> make parent directories also if it does not exist

touch -> make a 0 byte file if it does not exist
or
update date stamp of file if it exists

cp -> copy (for files)
cp -a -> copy (for directories)
cp -p -> copy and preserve date and time
mv -> move OR rename
rmdir -> remove empty directory
rm -> remove (for files)
rm -f -> remove forcefully (" ")
rm -r -> remove recursively (for directories)
rm -rf -> remove recursively and forcefully (" ")
cat -> display content of the file
cat -n -> display content of the file and number the lines
cal -> display calendar for current month
date -> display system date and time
date -s '<value>' -> change system date and time in mm/dd/yy
hwclock -> display the hardware clock
hwclock --hctosys -> set the system time from the hardware clock
ln -s -> make a soft/sym/symbolic link
ln -> make a hard link
history -> display the list of the last 1000 commands
! 100 -> Run command 100 in history
vi -> text editor
pico -^
mcedit -^
joe -^
aspell -c <filename> -> check the spelling in the file
lynx -> web browser
lynx -dump <url>
links -^
elinks -^

mtools
======
mdir
mcopy
mformat

file -> display the type of file
which -> display the path of the binary
hostname -> display system name with domain
id -> display id info of current user
id -u -> display user id of current user
id -un -> display username of current user
id -g -> display group id of current user
id -gn -> display groupname of current user
uptime -> display for how long the system has been running
tty -> display current terminal number
users -> display no. of users currently logged in
whoami -> display username of current user
who -> display users logged in the system with their respective terminals and time since logged in
who am i -> display current user, terminal and uptime
w -> display is details which files are open on which terminal
finger
finger <user>
pinky
pinky <user>
ps -> display process status of current terminal
ps -l -> display process status of current terminal in detail
ps -e -> display process status of all terminals
ps -el -> display process status of all terminals in detail
uname -s -> display kernel name
uname -r -> display release
uname -v -> display version
uname -p -> display processor type
uname -m -> display machine type
reset -> reset the current terminal
locate <file> -> searches /var/lib/slocate/slocatedb
Use updatedb* to rebuild the database
find <$file> -name <file> eg find / -name dad -print Find file "dad"

eg find / -name "dad*" -print Find all files starting with dad
init 6 -> reboot the system
reboot -^
shutdown -tx -r now -^ where x is in seconds
shutdown +x -^ where x is in minutes
init 0 -> shutdown system
halt -> halt the system after shutdown
poweroff -^

Program #: 2

OBJECTIVE

To understand Linux commands for redirection, pipes, filters, job control, file ownership, file permissions, links and file system hierarchy.

|	-> give output of one binary to another binary (pipe)
| more	-> give output to more
| less	-> give output to less
| grep	-> search from the output of previous binary and display (global regular expression print)
q	-> quit
>	-> overwrite
>>	-> append
-h	-> human readable
wc	-> word count
head	-> extract from the beginning
tail	-> extract from the ending
``	-> execute first
man	-> display detail manual
--help	-> display brief help
nl	-> number the lines

Program #: 3

OBJECTIVE

To understand the Shell Programming

AIM

Write shell script to show various system configurations like

- Currently logged user and his log name
- Your current shell
- Your home directory
- Your operating system type
- Your current path setting
- Your current working directory
- Show Currently logged number of users

```
nouser=`who | wc -l`
echo -e "User name: $USER (Login name: $LOGNAME)" >> /tmp/info.tmp.01.$$$
echo -e "Current Shell: $SHELL"  >> /tmp/info.tmp.01.$$$
echo -e "Home Directory: $HOME" >> /tmp/info.tmp.01.$$$
echo -e "Your O/s Type: $OSTYPE" >> /tmp/info.tmp.01.$$$
echo -e "PATH: $PATH" >> /tmp/info.tmp.01.$$$
echo -e "Current directory:  `pwd`" >> /tmp/info.tmp.01.$$$
echo -e "Currently Logged: $nouser user(s)" >> /tmp/info.tmp.01.$$$
```

Program #: 4

OBJECTIVE

To understand system configuration

AIM

Write shell script to show various system configurations like

- about your OS and version, release number, kernel version
- Show all available shells
- Show mouse settings
- Show computer CPU information like processor type, speed etc
- Show memory information
- Show hard disk information like size of hard-disk, cache memory, model etc
- File system (Mounted)

```
if [ -f /etc/redhat-release ]
then
   echo -e "OS: `cat /etc/redhat-release`" >> /tmp/info.tmp.01.$$$
fi

if [ -f /etc/shells ]
then
   echo -e "Available Shells: " >> /tmp/info.tmp.01.$$$
   echo -e "`cat /etc/shells`"  >> /tmp/info.tmp.01.$$$
fi

if [ -f /etc/sysconfig/mouse ]
then
   echo -e "---------------------------------------------------------------" >>
/tmp/info.tmp.01.$$$
   echo -e "Computer Mouse Information: " >> /tmp/info.tmp.01.$$$
   echo -e "---------------------------------------------------------------" >>
/tmp/info.tmp.01.$$$
   echo -e "`cat /etc/sysconfig/mouse`" >> /tmp/info.tmp.01.$$$
fi
echo -e "---------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer CPU Information:" >> /tmp/info.tmp.01.$$$
echo -e "---------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/cpuinfo >> /tmp/info.tmp.01.$$$
```

```
echo -e "------------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "Computer Memory Information:" >> /tmp/info.tmp.01.$$$
echo -e "------------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/meminfo >> /tmp/info.tmp.01.$$$

if [ -d /proc/ide/hda ]
then
   echo -e "------------------------------------------------------------------" >>
/tmp/info.tmp.01.$$$
   echo -e "Hard disk information:" >> /tmp/info.tmp.01.$$$
   echo -e "------------------------------------------------------------------" >>
/tmp/info.tmp.01.$$$
   echo -e "Model: `cat /proc/ide/hda/model` " >> /tmp/info.tmp.01.$$$
   echo -e "Driver: `cat /proc/ide/hda/driver` " >> /tmp/info.tmp.01.$$$
   echo -e "Cache size: `cat /proc/ide/hda/cache` " >> /tmp/info.tmp.01.$$$
fi
echo -e "------------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
echo -e "File System (Mount):" >> /tmp/info.tmp.01.$$$
echo -e "------------------------------------------------------------------" >> /tmp/info.tmp.01.$$$
cat /proc/mounts >> /tmp/info.tmp.01.$$$

if which dialog > /dev/null
then
   dialog  --backtitle "Linux Software Diagnostics (LSD) Shell Script Ver.1.0" --title "Press
Up/Down Keys to move" --textbox  /tmp/info.tmp.01.$$$ 21 70
else
   cat /tmp/info.tmp.01.$$$ |more
fi

rm -f /tmp/info.tmp.01.$$$
```

Program #: 5

OBJECTIVE

To familiarize concepts of shell scripting

AIM

Shell script program for scientific calculator.

```
while true; do
   read -p "what's the first number? " n1
   read -p "what's the second number? " n2
   PS3="what's the operation? "
   select ans in add subtract multiply divide; do
      case $ans in
         add) op='+' ; break ;;
         subtract) op='-' ; break ;;
         multiply) op='*' ; break ;;
         divide) op='/' ; break ;;
         *) echo "invalid response" ;;
      esac
   done
   ans=$(echo "$n1 $op $n2" | bc -l)
   printf "%s %s %s = %s\n\n" "$n1" "$op" "$n2" "$ans"
done
```

Program #: 6

OBJECTIVE

Install and set up a LAMP server and run a PHP program.

AIM

Configure a web server and create applications like login forms using PHP.

Install Apache

To start off we will install Apache.

Open up the Terminal (Applications > Accessories > Terminal). (Ctrl+T also works)
Copy/Paste the following line of code into Terminal and then press enter:

sudo apt-get install apache2

The Terminal will then ask you for your password type it and then press enter.
Testing Apache

To make sure everything installed correctly we will now test Apache to ensure it is working properly.

Open up any web browser and then enter the following into the web address:
http://localhost/

You should see a folder entitled apache2-default/. Open it and you will see a message saying "It works!" . congrats to you!

Install PHP

In this part we will install PHP 5.

Step 1. Again open up the Terminal (Applications > Accessories > Terminal).
Step 2. Copy/Paste the following line into Terminal and press enter:

sudo apt-get install php5 libapache2-mod-php5

Step 3. In order for PHP to work and be compatible with Apache we must restart it. Type the following code in Terminal to do this:

sudo /etc/init.d/apache2 restart

Test PHP -- To ensure there are no issues with PHP let's give it a quick test run.

Step 1. In the terminal copy/paste the following line: updated

sudo gedit /var/www/html/testphp.php

This will open up a file called phptest.php.

Step 2. Copy/Paste this line into the phptest file:

<?php phpinfo(); ?>

Step 3. Save and close the file.

Step 4. Now open your web browser and type the following into the web address:

http://localhost/testphp.php

Program #: 7

OBJECTIVE

Version Control System setup and usage using GIT.

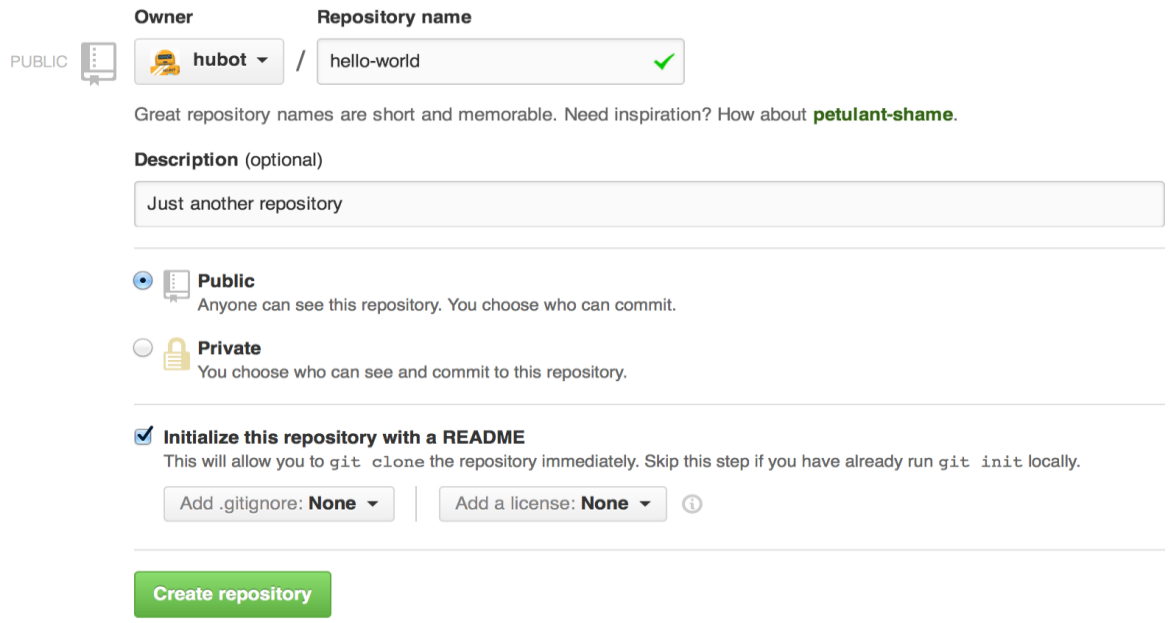
AIM

Create a repository to perform the following operations

- Checking out a repository
- Adding content to the repository
- Committing the data to a repository
- Updating the local copy
- Comparing different revisions
- Revert
- Conflicts and Solving a conflict

create a new repository

1. In the upper right corner, next to your avatar or identicon, click and then select **New repository**.
2. Name your repository hello-world.
3. Write a short description.
4. Select **initialize this repository with a README**



Step 2. Create a Branch

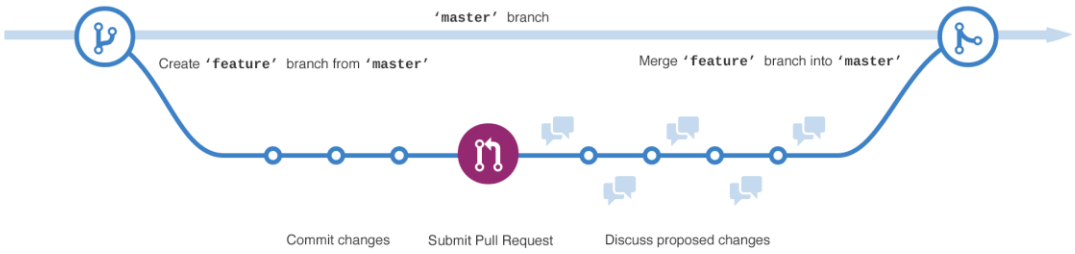
Branching is the way to work on different versions of a repository at one time.

By default your repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.

When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.

This diagram shows:

- The master branch
- A new branch called feature (because we're doing 'feature work' on this branch)
- The journey that feature takes before it's merged into master



To create a new branch

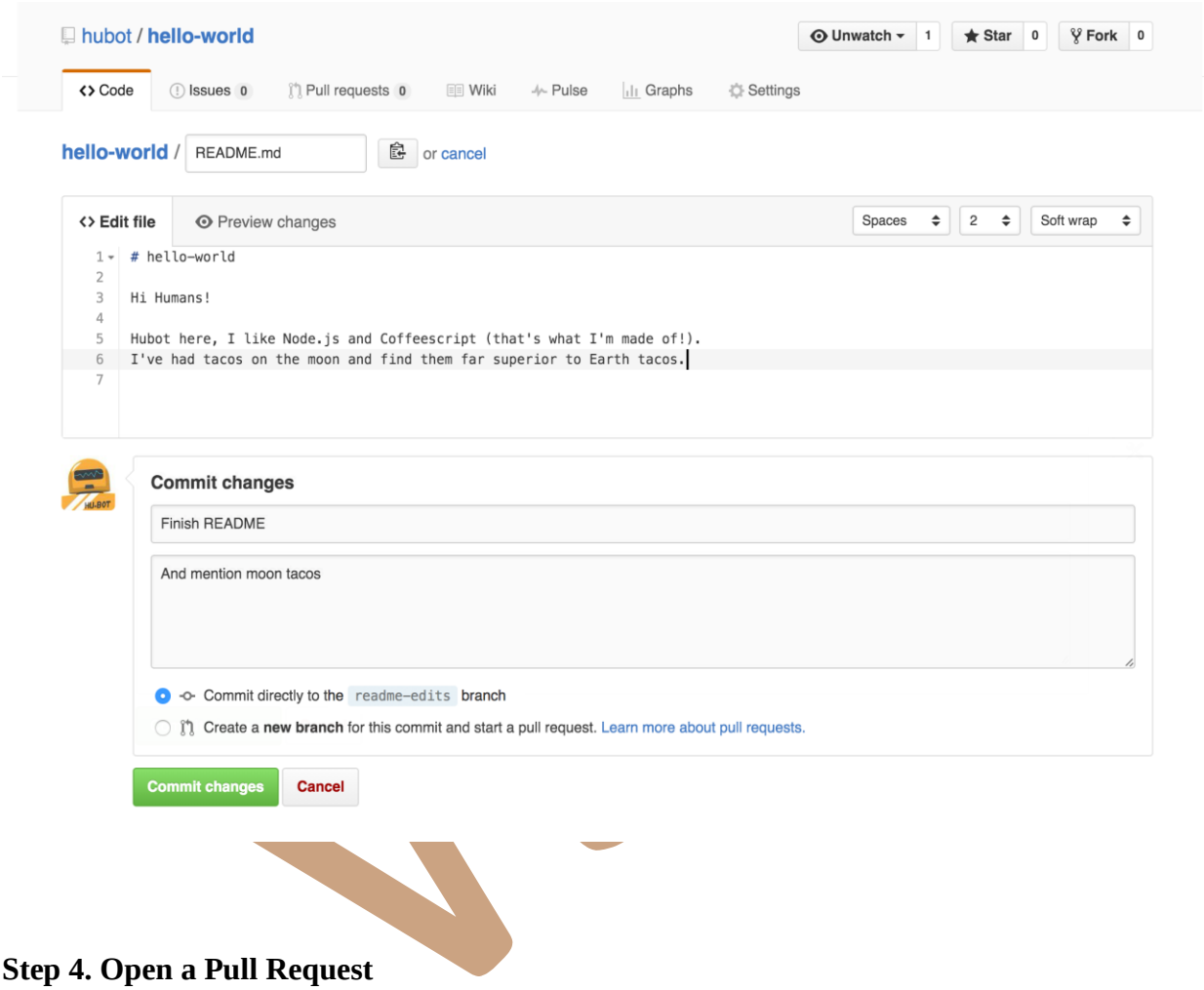
1. Go to your new repository hello-world.
2. Click the drop down at the top of the file list that says **branch: master**.
3. Type a branch name, readme-edits, into the new branch text box.
4. Select the blue **Create branch** box or hit "Enter" on your keyboard.

Step 3. Make and commit changes

On GitHub, saved changes are called *commits*. Each commit has an associated *commit message*, which is a description explaining why a particular change was made. Commit messages capture the history of your changes, so other contributors can understand what you've done and why.

Make and commit changes

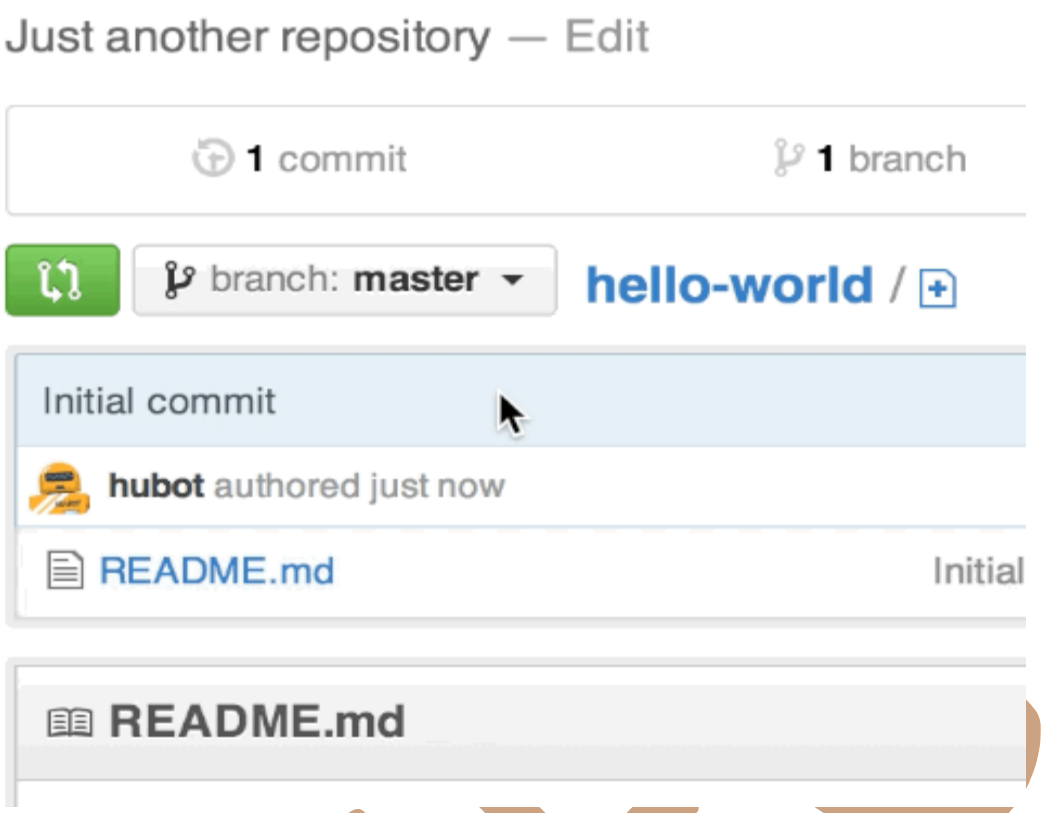
1. Click the README.md file.
2. Click the pencil icon in the upper right corner of the file view to edit.
3. In the editor, write a bit about yourself.
4. Write a commit message that describes your changes.
5. Click **Commit changes** button.



Step 4. Open a Pull Request

Pull Requests are the heart of collaboration on GitHub. When you open a *pull request*, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show *diffs*, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.



Step 5. Merge your Pull Request

In this final step, it's time to bring your changes together – merging your readme-edits branch into the master branch.

1. Click the green **Merge pull request** button to merge the changes into master.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.

Program #: 8

OBJECTIVE

Introduction to packet management system

AIM

Set up a local repository and install software from configured repository.

Creating local-repo

@server side

1) install dpkg-dev from syn-pack

2) Install apache2

3) install ur packages that u want to share

4) create a dir under /var/www/html -- here "my-repo/binary"

```
mkdir -p /var/www/html/my-repo/binary
```

5) copy all packages *.deb from /var/cache/apt/archives to /var/www/html/my-repo/binary

6) under "my-repo" enter the command dpkg-scanpackages binary /dev/null | gzip -9c > binary/Packages.gz

7) Under binary create a file called "Release"

```
Archive : stable
Component : base
Origin : Justin
Label : Justin Repo
Architecture : i386
```

Check List :

1) Open ur browser and type : http://127.0.0.1/my-repo/

Your Apache server works fine message will display

@Client Side

1) Backup ur sources.list in sources.list.orig

2) open new sources.list

vi /etc/apt/sources.list

3) Give an new entry to ur local repo

deb http://server-repo-ip/my-repo/ binary/

4) Save ur file

5) type a command
apt-get update

6) Install ur softwares from ur local repo

7) ENJOY

Program #:9

OBJECTIVE

Shell script to implement a script which kills every process which uses more than a specified value of memory or CPU and is run upon system start.

AIM

Design a script which kills every process which uses more than a specified value of memory or CPU and is run upon system start.

```
while [ 1 ];
do
echo
echo checking for run-away process ...

CPU_USAGE=$(uptime | cut -d"," -f4 | cut -d":" -f2 | cut -d" " -f2 | sed -e "s/\.//g")
CPU_USAGE_THRESHOLD=800
PROCESS=$(ps aux r)
TOPPROCESS=$(ps -eo pid -eo pcpu -eo command | sort -k 2 -r | grep -v PID | head -n 1)

if [ $CPU_USAGE -gt $CPU_USAGE_THRESHOLD] ; then
  kill -9 $(ps -eo pid | sort -k 1 -r | grep -v PID | head -n 1) #original
  kill -9 $(ps -eo pcpu | sort -k 1 -r | grep -v %CPU | head -n 1)
  kill -9 $TOPPROCESS
  echo system overloading!
  echo Top-most process killed $TOPPROCESS
     echo CPU USAGE is at $CPU_LOAD

else
    fi
    exit 0
    sleep 1;
    done
```

Program #: 10

OBJECTIVE

Installing various software packages.

AIM

- Install samba and share files to windows
- Install Common Unix Printing System (CUPS)

1. Install Samba

```
1. sudo apt-get update
2. sudo apt-get install samba
```

2. Set a password for your user in Samba

```
1. sudo smbpasswd -a <user_name>
```

1. Note: Samba uses a separate set of passwords than the standard Linux system accounts (stored in /etc/samba/smbpasswd), so you'll need to create a Samba password for yourself.

Tip1: Use the password for your own user to facilitate.
Tip2: Remember that your user must have permission to write and edit the folder you want to share.
Eg.:
sudo chown <user_name> /var/opt/blah/blahblah
sudo chown :<user_name> /var/opt/blah/blahblah
Tip3: If you're using another user than your own, it needs to exist in your system beforehand, you can create it without a shell access using the following command :
sudo useradd USERNAME --shell /bin/false

You can also hide the user on the login screen by adjusting lightdm's configuration, in /etc/lightdm/users.conf add the newly created user to the line :
hidden-users=

3. Create a directory to be shared

```
mkdir /home/<user_name>/<folder_name>
```

4. Make a safe backup copy of the original smb.conf file to your home folder, in case you make an error

```
sudo cp /etc/samba/smb.conf ~
```

5. Edit the file "/etc/samba/smb.conf"

```
sudo nano /etc/samba/smb.conf
```

1. Once "smb.conf" has loaded, add this to the very end of the file:
2.
3. [<folder_name>]
4. path = /home/<user_name>/<folder_name>
5. valid users = <user_name>
6. read only = no

6. Restart the samba:

```
sudo service smbd restart
```

7. Once Samba has restarted, use this command to check your smb.conf for any syntax errors

```
testparm
```

8. To access your network share

```
9.  sudo apt-get install smbclient
10. # List all shares:
11. smbclient -L //<HOST_IP_OR_NAME>/<folder_name> -U <user>
12. # connect:
    smbclient //<HOST_IP_OR_NAME>/<folder_name> -U <user>
```

II. **Procedure for CUPS Configuration.**

Step1: Open the terminal and log in.
Step2: Type the following command
$ rpm –qa | grep cups.
Step3: Type the command
$rpm –qi cups
Step4: Then using clear command clear the screen.
Step5: Login to super user .
$su
Step6: Check the status of the cups using the following command
#service cups status
Step7: Start the service of cups using the following command
#service cups status

Step8: Check the status of the cups again.
Step9: Click Mozilla browser and type
http://localhost:631/
Step10: Select the option Adding printer and classes.
Step11: Select the option “Add printer”
Step12: Give the root username and root password.
Step13: Select the printer type and select continue option.

Step14: Give the printer name, location and description and give continue option.
Step15: Select the maker of printer and select the continue option.
Step16: Select the model of printer and select the option ―Add printer‖
Step17: Select the paper size and set the default options.
Step18: Click on the printer name and see the jobs that are pending.

Program #: 11

OBJECTIVE

To understand Advanced linux commands curl, wget, ftp, ssh and grep

AIM

1. Implement a secure channel used to login to a remote machine and execute commands
2. Configure an ftp server to handle large number of connections effectively.

I. SSH (Secure Shell Protocol)

Typically used to log in to a remote machine and execute commands.

SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server.

Install SSH Server

#apt-get install openssh-server

#service ssh restart

SSH Client

#apt-get install openssh-client

Connect to the SSH server with a common user

#ssh user1@<ssh server ip>

eg: #ssh user1@192.168.1.105

Connect to the SSH server with root user

#ssh <ssh server ip>

#ssh 192.168.1.105

Secure Copy (scp)

To copy a file from your computer to another computer with ssh, go to a command-line and type:

scp <file> <username>@<IP address or hostname>:<Destination>

eg: scp file1 user2@172.16.20.231:.

scp file2 user2@172.16.20.231:Desktop

scp -r test1 user2@172.16.20.231:.

Copy from Remote machine to our machine

scp user2@172.16.20.231:file4 .

scp -r user2@172.16.20.231:test5

II. FTP Server – Vsftpd (Vsftpd - Very Secure FTP Daemon)

Fast, Stable & Secure, Handle large number of connections effectively.

Install Vsftpd

#apt-get install vsftpd

Edit vsftpd.conf file

#vim /etc/vsftpd.conf

anonymous_enable=YES

write_enable=YES

:wq

service vsftpd restart

Program #: 12

OBJECTIVE

Virtualization environment

AIM

Set up a virtual box and install windows within Ubuntu system as an application.

- Install Virtual Box.

1. Open Virtual Box and select New. A new window will come out.
2. Choose your guest OS and architecture (32 vs. 64 bit, eg select Ubuntu)
3. Set your Base Memory (RAM)
4. Click next until it show the vm storage size. Put how much space you need depending on your hardisk and finish the wizard by clicking the create button.
5. On Virtual Box main window, select START and pick your MEDIA SOURCE. In your case, select iso on your desktop.
6. Finish the installation as normal install.
7. Remove your installation iso from the virtual optical disk drive before restarting the VM.
8. Install Guest Additions.