MGM’s

Jawaharlal Nehru Engineering College

# Laboratory Manual

## Open Source Software Technology

## For

Second Year Students CSE

Dept: Computer Science & Engineering (NBA Accredited)

**FOREWORD**

It is my great pleasure to present this laboratory manual for Second year engineering students for the subject of Open Source Software Technology.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that MGM has already been awarded with ISO 9000 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO 9000 Certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Dr. S.D.Deshmukh.

Principal

### LABORATORY MANUAL CONTENTS

This manual is intended for the Second year students of Computer Science and Engineering in the subject of Open Source Software Technology. This manual typically contains practical/Lab Sessions related Open Source Software Technology covering various aspects related the subject to enhanced understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Prof. V .B. Musande Mr. S.S.More

HOD, CSE Dept. Asst.Prof.,CSE Dept

**MGM’s**

****

**Jawaharlal Nehru Engineering College, Aurangabad**

**Department of Computer Science and Engineering**

**Vision of CSE Department**

To develop computer engineers with necessary analytical ability and human values who can creatively design, implement a wide spectrum of computer systems for welfare of the society.

**Mission of the CSE Department:**

* Preparing graduates to work on multidisciplinary platforms associated with their professional position both independently and in a team environment.
* Preparing graduates for higher education and research in computer science and engineering enabling them to develop systems for society development.

**Programme Educational Objectives**

**Graduates will be able to**

1. To analyze, design and provide optimal solution for Computer Science & Engineering and multidisciplinary problems.
2. To pursue higher studies and research by applying knowledge of mathematics and fundamentals of computer science.
3. To exhibit professionalism, communication skills and adapt to current trends by engaging in lifelong learning.

**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 anddesign 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 ofthe 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 independent and life-long learning in the broadest context of technological change.

|  |  |  |  |
| --- | --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad.** | | | |
| **Practical Experiment Instruction Sheet** | | | |
| **Department:** CSE | | **Subject:**  Open Source Lab | Date: |
|  | | | |
| **List of Experiments** | | | |
| **Exp No.** | **Title** | | |
| 1 | Installation of Linux | | |
| 2 | Use of various commands | | |
| 3 | Use of Text Processing Tools: grep, cut. | | |
| 4 | User and Group Creation | | |
| 5 | Back up using tar | | |
| 6 | Installation using RPM | | |
| 7 | C/C++ program using cc / gcc | | |
| 8 | Configuring Apache | | |
| 9 | PHP scripts for other tasks | | |
| 10 | MySQL Installation, Configuration and Testing | | |
| 11 | Design of admission form using PHP – MYSQL | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No:03** |
| **Experiment Title:** Use of Text Processing Tools: grep, cut. | | |
| **Aim:** To study the use of various commands used for Text Processing.  **Theory:**  1. Search for the given string in a single file  The basic usage of grep command is to search for a specific string in the specified file as shown below.  Syntax:  grep "literal\_string" filename  $ grep "this" demo\_file  this line is the 1st lower case line in this file.  Two lines above this line is empty.  And this is the last line.  2. Checking for the given string in multiple files.  Syntax:  grep "string" FILE\_PATTERN  This is also a basic usage of grep command. For this example, let us copy the demo\_file to demo\_file1. The grep output will also include the file name in front of the line that matched the specific pattern as shown below. When the Linux shell sees the meta character, it does the expansion and gives all the files as input to grep.  $ cp demo\_file demo\_file1  $ grep "this" demo\_\*  demo\_file:this line is the 1st lower case line in this file.  demo\_file:Two lines above this line is empty.  demo\_file:And this is the last line.  demo\_file1:this line is the 1st lower case line in this file.  demo\_file1:Two lines above this line is empty.  demo\_file1:And this is the last line  .  3. Case insensitive search using grep -i  Syntax:  grep -i "string" FILE  This is also a basic usage of the grep. This searches for the given string/pattern case insensitively. So it matches all the words such as “the”, “THE” and “The” case insensitively as shown below.  $ grep -i "the" demo\_file  THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.  this line is the 1st lower case line in this file.  This Line Has All Its First Character Of The Word With Upper Case.  And this is the last line.  4. Match regular expression in files  Syntax:  grep "REGEX" filename  This is a very powerful feature, if you can use use regular expression effectively. In the following example, it searches for all the pattern that starts with “lines” and ends with “empty” with anything in-between. i.e To search “lines[anything in-between]empty” in the demo\_file.  $ grep "lines.\*empty" demo\_file  Two lines above this line is empty.  From documentation of grep: A regular expression may be followed by one of several repetition operators:   * ? The preceding item is optional and matched at most once. * \* The preceding item will be matched zero or more times. * + The preceding item will be matched one or more times. * {n} The preceding item is matched exactly n times. * {n,} The preceding item is matched n or more times. * {,m} The preceding item is matched at most m times. * {n,m} The preceding item is matched at least n times, but not more than m times.   5. Checking for full words, not for sub-strings using grep -w  If you want to search for a word, and to avoid it to match the substrings use -w option. Just doing out a normal search will show out all the lines.  The following example is the regular grep where it is searching for “is”. When you search for “is”, without any option it will show out “is”, “his”, “this” and everything which has the substring “is”.  $ grep -i "is" demo\_file  THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.  this line is the 1st lower case line in this file.  This Line Has All Its First Character Of The Word With Upper Case.  Two lines above this line is empty.  And this is the last line.  The following example is the WORD grep where it is searching only for the word “is”. Please note that this output does not contain the line “This Line Has All Its First Character Of The Word With Upper Case”, even though “is” is there in the “This”, as the following is looking only for the word “is” and not for “this”.  $ grep -iw "is" demo\_file  THIS LINE IS THE 1ST UPPER CASE LINE IN THIS FILE.  this line is the 1st lower case line in this file.  Two lines above this line is empty.  And this is the last line.  6. Displaying lines before/after/around the match using grep -A, -B and -C  When doing a grep on a huge file, it may be useful to see some lines after the match. You might feel handy if grep can show you not only the matching lines but also the lines after/before/around the match.  Please create the following demo\_text file for this example.  $ cat demo\_text  4. Vim Word Navigation  You may want to do several navigation in relation to the words, such as:  \* e - go to the end of the current word.  \* E - go to the end of the current WORD.  \* b - go to the previous (before) word.  \* B - go to the previous (before) WORD.  \* w - go to the next word.  \* W - go to the next WORD.  WORD - WORD consists of a sequence of non-blank characters, separated with white space.  word - word consists of a sequence of letters, digits and underscores.  Example to show the difference between WORD and word  \* 192.168.1.1 - single WORD  \* 192.168.1.1 - seven words.  **6.1 Display N lines after match**  -A is the option which prints the specified N lines after the match as shown below.  Syntax:  grep -A <N> "string" FILENAME  The following example prints the matched line, along with the 3 lines after it.  $ grep -A 3 -i "example" demo\_text  Example to show the difference between WORD and word  \* 192.168.1.1 - single WORD  \* 192.168.1.1 - seven words.  **6.2 Display N lines before match**  -B is the option which prints the specified N lines before the match.  Syntax:  grep -B <N> "string" FILENAME  When you had option to show the N lines after match, you have the -B option for the opposite.  $ grep -B 2 "single WORD" demo\_text  Example to show the difference between WORD and word  \* 192.168.1.1 - single WORD  **6.3 Display N lines around match**  -C is the option which prints the specified N lines before the match. In some occasion you might want the match to be appeared with the lines from both the side. This options shows N lines in both the side(before & after) of match.  $ grep -C 2 "Example" demo\_text  word - word consists of a sequence of letters, digits and underscores.  Example to show the difference between WORD and word  \* 192.168.1.1 - single WORD  7. Highlighting the search using GREP\_OPTIONS  As grep prints out lines from the file by the pattern / string you had given, if you wanted it to highlight which part matches the line, then you needs to follow the following way.  When you do the following export you will get the highlighting of the matched searches. In the following example, it will highlight all the this when you set the GREP\_OPTIONS environment variable as shown below.  $ export GREP\_OPTIONS='--color=auto' GREP\_COLOR='100;8'  $ grep this demo\_file  **this** line is the 1st lower case line in this file.  Two lines above **this** line is empty.  And **this** is the last line.  8. Searching in all files recursively using grep -r  When you want to search in all the files under the current directory and its sub directory. -r option is the one which you need to use. The following example will look for the string “ramesh” in all the files in the current directory and all it’s subdirectory.  $ grep -r "ramesh" \*  9. Invert match using grep -v  You had different options to show the lines matched, to show the lines before match, and to show the lines after match, and to highlight match. So definitely You’d also want the option -v to do invert match.  When you want to display the lines which does not matches the given string/pattern, use the option -v as shown below. This example will display all the lines that did not match the word “go”.  $ grep -v "go" demo\_text  4. Vim Word Navigation  You may want to do several navigation in relation to the words, such as:  WORD - WORD consists of a sequence of non-blank characters, separated with white space.  word - word consists of a sequence of letters, digits and underscores.  Example to show the difference between WORD and word  \* 192.168.1.1 - single WORD  \* 192.168.1.1 - seven words.  10. display the lines which does not matches all the given pattern.  Syntax:  grep -v -e "pattern" -e "pattern"  $ cat test-file.txt  a  b  c  d  $ grep -v -e "a" -e "b" -e "c" test-file.txt  11. Counting the number of matches using grep -c  When you want to count that how many lines matches the given pattern/string, then use the option -c.  Syntax:  grep -c "pattern" filename  $ grep -c "go" demo\_text  6  When you want do find out how many lines matches the pattern  $ grep -c this demo\_file  3  When you want do find out how many lines that does not match the pattern  $ grep -v -c this demo\_file  4  12. Display only the file names which matches the given pattern using grep -l  If you want the grep to show out only the file names which matched the given pattern, use the -l (lower-case L) option.  When you give multiple files to the grep as input, it displays the names of file which contains the text that matches the pattern, will be very handy when you try to find some notes in your whole directory structure.  $ grep -l this demo\_\*  demo\_file  demo\_file1  13. Show only the matched string  By default grep will show the line which matches the given pattern/string, but if you want the grep to show out only the matched string of the pattern then use the -o option.  It might not be that much useful when you give the string straight forward. But it becomes very useful when you give a regex pattern and trying to see what it matches as  $ grep -o "is.\*line" demo\_file  is line is the 1st lower case line  is line  is is the last line  14. Show the position of match in the line  When you want grep to show the position where it matches the pattern in the file, use the following options as  Syntax:  grep -o -b "pattern" file  $ cat temp-file.txt  12345  12345  $ grep -o -b "3" temp-file.txt  2:3  8:3  **2] Cut Command:**  Linux command cut is used for text processing. You can use this command to extract portion of text from a file by selecting columns. 1. Select Column of Characters To extract only a desired column from a file use -c option. The following example displays 2nd character from each line of a file test.txt  $ cut -c2 test.txt  a  p  s  As seen above, the characters a, p, s are the second character from each line of the test.txt file. 2. Select Column of Characters using Range Range of characters can also be extracted from a file by specifying start and end position delimited with -. The following example extracts first 3 characters of each line from a file called test.txt  $ cut -c1-3 test.txt  cat  cp  ls 3. Select Column of Characters using either Start or End Position Either start position or end position can be passed to cut command with -c option.  The following specifies only the start position before the ‘-’. This example extracts from 3rd character to end of each line from test.txt file.  $ cut -c3- test.txt  t command for file oriented operations.  command for copy files or directories.  command to list out files and directories with its attributes.  The following specifies only the end position after the ‘-’. This example extracts 8 characters from the beginning of each line from test.txt file.  $ cut -c-8 test.txt  cat comm  cp comma  ls comma  The entire line would get printed when you don’t specify a number before or after the ‘-’ as shown below.  $ cut -c- test.txt  cat command for file oriented operations.  cp command for copy files or directories.  ls command to list out files and directories with its attributes. 4. Select a Specific Field from a File Instead of selecting x number of characters, if you like to extract a whole field, you can combine option -f and -d. The option -f specifies which field you want to extract, and the option -d specifies what is the field delimiter that is used in the input file.  The following example displays only first field of each lines from /etc/passwd file using the field delimiter: (colon). In this case, the 1st field is the username. The file  $ cut -d':' -f1 /etc/passwd  root  daemon  bin  sys  sync  games  bala 5. Select Multiple Fields from a File You can also extract more than one fields from a file or stdout. Below example displays username and home directory of users who has the login shell as “/bin/bash”.  $ grep "/bin/bash" /etc/passwd | cut -d':' -f1,6  root:/root  bala:/home/bala  To display the range of fields specify start field and end field as shown below. In this example, we are selecting field 1 through 4, 6 and 7  $ grep "/bin/bash" /etc/passwd | cut -d':' -f1-4,6,7  root:x:0:0:/root:/bin/bash  bala:x:1000:1000:/home/bala:/bin/bash 6. Select Fields Only When a Line Contains the Delimiter In our /etc/passwd example, if you pass a different delimiter other than : (colon), cut will just display the whole line.  In the following example, we’ve specified the delimiter as | (pipe), and cut command simply displays the whole line, even when it doesn’t find any line that has | (pipe) as delimiter.  $ grep "/bin/bash" /etc/passwd | cut -d'|' -f1  root:x:0:0:root:/root:/bin/bash  bala:x:1000:1000:bala,,,:/home/bala:/bin/bash  But, it is possible to filter and display only the lines that contains the specified delimiter using -s option.  The following example doesn’t display any output, as the cut command didn’t find any lines that has | (pipe) as delimiter in the /etc/passwd file.  $ grep "/bin/bash" /etc/passwd | cut -d'|' -s -f1 7. Select All Fields Except the Specified Fields In order to complement the selection field list use option –complement.  The following example displays all the fields from /etc/passwd file except field 7  $ grep "/bin/bash" /etc/passwd | cut -d':' --complement -s -f7  root:x:0:0:root:/root  bala:x:1000:1000:bala,,,:/home/bala 8. Change Output Delimiter for Display By default the output delimiter is same as input delimiter that we specify in the cut -d option.  To change the output delimiter use the option –output-delimiter as shown below. In this example, the input delimiter is : (colon), but the output delimiter is # (hash).  $ grep "/bin/bash" /etc/passwd | cut -d':' -s -f1,6,7 --output-delimiter='#'  root#/root#/bin/bash  bala#/home/bala#/bin/bash 9. Change Output Delimiter to Newline In this example, each and every field of the cut command output is displayed in a separate line. We still used –output-delimiter, but the value is $’\n’ which indicates that we should add a newline as the output delimiter.  $ grep bala /etc/passwd | cut -d':' -f1,6,7 --output-delimiter=$'\n'  bala  /home/bala  /bin/bash 10. Combine Cut with Other Unix Command Output The power of cut command can be realized when you combine it with the stdout of some other Unix command.  Once you master the basic usage of cut command that we’ve explained above, you can wisely use cut command to solve lot of your text manipulation requirements.  The following example indicates how you can extract only useful information from the [ps command](http://www.thegeekstuff.com/2011/04/ps-command-examples/) output. We also showed how we’ve filtered the output of ps command using grep and sed before the final output was given to cut command. Here, we’ve used cut option -d and -f which we’ve explained in the above examples.  $ ps axu | grep python | sed 's/\s\+/ /g' | cut -d' ' -f2,11-  2231 /usr/bin/python /usr/lib/unity-lens-video/unity-lens-video  2311 /usr/bin/python /usr/lib/unity-scope-video-remote/unity-scope-video-remote  2414 /usr/bin/python /usr/lib/ubuntuone-client/ubuntuone-syncdaemon  2463 /usr/bin/python /usr/lib/system-service/system-service-d  3274 grep --color=auto python  **Conclusion:**  By using grep and cut command we can process the text as mentioned above. | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No:04** |
| **Experiment Title:** User and Group Creation | | |
| **Aim:** To study the creation of users and groups on Linux O.S.  **Theory:**  You can use the useradd or usermod commands to add a user to a group. The useradd command creates a new user or update default new user information. The usermod command modifies a user account and it is useful to add user to existing groups. There are two types of groups under Linux operating systems:   1. Primary user group. 2. Secondary or supplementary user group.   All user account related information are stored in the following files:   1. /etc/passwd - Contains one line for each user account. 2. /etc/shadow - Contains the password information in encrypted format for the system's accounts and optional account aging information. 3. /etc/group - Defines the groups on the system. 4. /etc/default/useradd - This file contains a value for the default group, if none is specified by the useradd command. 5. /etc/login.defs - This file defines the site-specific configuration for the shadow password suite stored in /etc/shadow file.  1: Creating user with all the default options, and with his own group. Following example creates user ramesh with group ramesh. Use Linux passwd command to change the password for the user immediately after user creation.  # **useradd ramesh**  # passwd ramesh  Changing password for user ramesh.  New UNIX password:  Retype new UNIX password:  passwd: all authentication tokens updated successfully.  # grep ramesh /etc/passwd  ramesh:x:500:500::/home/ramesh:/bin/bash  # grep ramesh /etc/group  ramesh:x:500:  [Note: default useradd command created ramesh as username and group] Example 2: Creating an user with all the default options, and with the default group. # **useradd -n sathiya**  # grep sathiya /etc/passwd  sathiya:x:511:100::/home/sathiya:/bin/bash  # grep sathiya /etc/group  [Note: No rows returned, as group sathiya was not created]  # grep 100 /etc/group  users:x:100:  [Note: useradd -n command created user sathiya with default group id 100]  # passwd sathiya  Changing password for user sathiya.  New UNIX password:  Retype new UNIX password:  passwd: all authentication tokens updated successfully. Example 3: Editing the default options used by useradd. The following example shows how to change the default shell from /bin/bash to /bin/ksh during user creation.  Syntax: # useradd -D --shell=<SHELLNAME>  # useradd -D  GROUP=100  HOME=/home  INACTIVE=-1  EXPIRE=  SHELL=/bin/bash  SKEL=/etc/skel  [Note: The default shell is /bin/bash]  # **useradd -D -s /bin/ksh**  # useradd -D  GROUP=100  HOME=/home  INACTIVE=-1  EXPIRE=  SHELL=/bin/ksh  SKEL=/etc/skel  [Note: Now the default shell changed to /bin/ksh]  # adduser priya  # grep priya /etc/passwd  priya:x:512:512::/home/priya:/bin/ksh  [Note: New users are getting created with /bin/ksh]  # **useradd -D -s /bin/bash**  [Note: Set it back to /bin/bash, as the above is only for testing purpose] Method 2: Linux useradd Command — Create Users With Custom Configurations Instead of accepting the default values (for example, group, shell etc.) that is given by the useradd command as shown in the above method, you can specify custom values in the command line as parameters to the useradd command.  Syntax: # useradd -s <SHELL> -m -d <HomeDir> -g <Group> UserName     * **-s SHELL** : Login shell for the user. * **-m** : Create user’s home directory if it does not exist. * **-d HomeDir** : Home directory of the user. * **-g Group** : Group name or number of the user. * **UserName** : Login id of the user.  Example 4: Create Linux User with Custom Configurations Using useradd Command The following example creates an account (lebron) with home directory /home/king, default shell as /bin/csh and with comment “LeBron James”.  # useradd -s /bin/csh -m -d /home/king -c "LeBron James" -g root lebron  # grep lebron /etc/passwd  lebron:x:513:0:LeBron James:/home/king:/bin/csh  **Note:** You can give the password using -p option, which should be encrypted password. Or you can use the passwd command to change the password of the user. Method 3: Linux adduser Command – Create Users Interactively These are the friendlier tools to the low level useradd. By default it chooses the Debian policy format for UID and GID. A very simple way of creating user in the command line interactively is using adduser command.  Syntax: # adduser USERNAME Example 5: Creating an User Interactively With adduser Command # **adduser spidey**  Adding user `spidey' ...  Adding new group `spidey' (1007) ...  Adding new user `spidey' (1007) with group `spidey' ...  Creating home directory `/home/spidey' ...  Copying files from `/etc/skel' ...  Enter new UNIX password:  Retype new UNIX password:  passwd: password updated successfully  Changing the user information for spidey  Enter the new value, or press ENTER for the default  Full Name []: Peter Parker  Room Number []:  Work Phone []:  Home Phone []:  Other []:  Is the information correct? [y/N] y Method 4: Linux newusers Command — Creating bulk users Sometimes you may want to to create multiple users at the same time. Using any one of the above 3 methods for bulk user creation can be very tedious and time consuming. Fortunately, Linux offers a way to upload users using newusers command. This can also be executed in batch mode as it cannot ask any input.  # newusers FILENAME  This file format is same as the password file.  loginname:password:uid:gid:comment:home\_dir:shell Example 6: Creating Large Number of Users Using newusers Command If Simpson family decides to join your organization and need access to your Linux server, you can create account for all of them together using newusers command as shown below.  # cat homer-family.txt  homer:HcZ600a9:1008:1000:Homer Simpson:/home/homer:/bin/bash  marge:1enz733N:1009:1000:Marge Simpson:/home/marge:/bin/csh  bart:1y5eJr8K:1010:1000:Bart Simpson:/home/bart:/bin/ksh  lisa:VGz638i9:1011:1000:Lisa Simpson:/home/lisa:/bin/sh  maggie:5lj3YGQo:1012:1000:Maggie Simpson:/home/maggie:/bin/bash 2] Group Creation: groupadd - create a new group SYNOPSIS  |  |  | | --- | --- | | **Tag** | **Description** | | **groupadd** [-g *gid* [-o]] [-r] [-f] [-K *KEY*=*VALUE*] *group* | |  DESCRIPTION The **groupadd** command creates a new group account using the values specified on the command line and the default values from the system. The new group will be entered into the system files as needed. OPTIONS The options which apply to the **groupadd** command are:   |  |  | | --- | --- | | **Tag** | **Description** | | **-f** | This option causes to just exit with success status if the specified group already exists. With **-g**, if specified GID already exists, other (unique) GID is chosen (i.e. **-g** is turned off). | | **-r** | This flag instructs **groupadd** to add a system account. The first available *gid* lower than 499 will be automatically selected unless the **-g** option is also given on the command line. This is an option added by Red Hat. | | **-g** *GID* | | |  | The numerical value of the group’s ID. This value must be unique, unless the **-o** option is used. The value must be non-negative. The default is to use the smallest ID value greater than 500 and greater than every other group. Values between 0 and 499 are typically reserved for system accounts. | | **-h**, **--help** | | |  | Display help message and exit. | | **-K***KEY***=***VALUE* | | |  | Overrides */etc/login.defs* defaults (GID\_MIN, GID\_MAX and others). Multiple **-K** options can be specified.  Example: **-K***GID\_MIN*=*100* **-K***GID\_MAX*=*499*  Note: **-K***GID\_MIN*=*10*,*GID\_MAX*=*499* doesn’t work yet. | | **-o** | This option permits to add group with non-unique GID. |  FILES  |  |  | | --- | --- | | **Tag** | **Description** | | */etc/group* | | |  | Group account information. | | */etc/gshadow* | | |  | Secure group account information. | | */etc/login.defs* | | |  | Shadow password suite configuration. | | | |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No:05** |
| **Experiment Title:** Back up using tar | | |
| **Aim:** Back up using tar.  **Theory:**  **Using tar** Tar stands for tape archive. tar command is used to create archive and extract the archive files. You will first perform a full backup and then a differential backup of some specific files and folders on your system.  **SYNTAX:**   The Syntax is       tar [options] [archive-file] [File or directory to be archived]  **OPTIONS:**   |  |  | | --- | --- | | -c | Creates Archive | | -x | Extract the archive | | -f | creates archive with give filename | | -t | displays or lists files in archived file | | -u | archives and adds to an existing archive file | | -v | Displays Verbose Information | | -A | Concatenates the archive files |   To Use tar to perform a full backup  1. Log on to the computer as root  2. Change your directory to the folder1 directory: [root@localhost root](http://www.labmanual.org/root@localhost%20root)# cd folder1  3. Create another directory named sample in the present working directory. And cd to the sample directory.  [root@localhost folder1](http://www.labmanual.org/root@localhost%20folder1)# mkdir sample && cd sample  4. Create five files .  [root@localhost sample](http://www.labmanual.org/root@localhost%20sample)# touch file1 file2 file3 file4 file5  5. Change back to the folder1 directory  [root@localhost sample](http://www.labmanual.org/root@localhost%20sample)# cd ..  6. Perform a full backup of the sample directory using tar. [root@localhost folder1](http://www.labmanual.org/root@localhost%20folder1)# tar -cvf sample.tar sample  sample/ sample/file1 sample/file2 sample/file3 sample/file4 sample/file5  7. Delete the sample directory and then restore it from the tar archive.  [root@localhost folder1](http://www.labmanual.org/root@localhost%20folder1)# rm –rf sample  8. Now restore the sample directory from the tar archive.  [root@localhost folder1](http://www.labmanual.org/root@localhost%20folder1)# tar –xvf sample.tar && cd sample  9. List the contents of the sample directory | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No:06** |
| **Experiment Title:** Installation using RPM. | | |
| **Aim:** Installation Using RPM**.**  **Theory:**  RPM stands for Red Hat Package Manager. However, these days RPM isn't only Red Hat specific because many other Linux distros use RPM for managing their software. For example, both Mandriva and SuSE use RPM for software management. With RPM, you can install, upgrade and uninstall software on Linux, as well as keep track of already installed RPM packages on your system. This can be done because RPM keeps a database of all software that was installed with it.  RPM uses software packages that have (surprise) the .rpm extension. An RPM package contains the actual software that gets installed, maybe some additional files for the software, information on where the software and its files get installed, and a list of other files you need to have on your system in order to run this specific piece of software.  When you use RPM for installing the software package, RPM checks if your system is suitable for the software the RPM package contains, figures out where to install the files the package provides, installs them on your system, and adds that piece of software into its database of installed RPM packages.  Note that different Linux distros may keep their software and the files related to that software in different directories. That's why it's important to use the RPM package that was made for your distribution. For example, if you install a SuSE specific software package on a Red Hat system, RPM may put the files from that package into wrong directories. In the worst case the result is that the program doesn't find all the files it needs and doesn't work properly.  Note that you need to be root when installing software in Linux. When you've got the root privileges, you use the rpm command with appropriate options to manage your RPM software packages. < Installing and upgrading RPM packages > For installing a software package, you use the rpm command with -i option (which stands for "install"). For example, to install an RPM package called software-2.3.4.rpm: # **rpm -i software-2.3.4.rpm**  If you already have some version installed on your system and want to upgrade it to the new version, you use -U option instead (which stands for "upgrade"). For example, if you have software-2.3.3.rpm installed and want to upgrade it: # **rpm -U software-2.3.4.rpm**  If all goes well, the files in your package will get installed into your system and you can happily run your new program. But where is your new program? Note that rpm doesn't usually create a special directory for the software package's files. Instead, the different files from the package get placed into appropriate existing directories on your Linux system. Executable programs go usually into /bin, /usr/bin, /usr/X11/bin, or /usr/X11R6/bin after installing with rpm.  But how can you run your new program if you don't know where the executable is? Sometimes the program gets automatically added into your menu, but usually you can just run the program by typing its name at the command prompt. In most cases you don't have to know where the program was installed because you don't have to type the whole path when running the program, only the program's name is needed. < Error: failed dependencies > Issuing rpm -i or rpm -U installs the software and you can start using it. RPM is very easy when it works. However, RPM can be a devil when it doesn't work. There are many reasons why installing software goes wrong, but usually it's because of failed dependencies.  You see, many Linux programs need other files or programs in order to work properly. In other words, a certain piece of software depends on other software. When you try to install an RPM package, RPM automatically checks its database for other files that the software being installed needs. If RPM can't find those files in its database, it stops installing the software and complains about failed dependencies.  When you get a dependency error, RPM spits out a list of files the program needs. Take a look at the list. The files in the list are probably ones you don't have on your system, or files you have but are wrong versions. When you get the dreaded dependency error, you'll have to find the files RPM complains about, install or upgrade those files first, and then try to install the package you were installing in the first place.  However, sometimes RPM is just plain stupid. You see, only software that was installed with **RPM** gets added into the database of installed software. This means that if you've used some other method for installing a certain program, RPM doesn't know the program exists on your system. In this case RPM complains about failed dependencies even when the needed program does exist on your system and there are no failed dependencies!  If you know the needed files are there and RPM is just being stupid, you can ignore the dependencies. Use the --nodeps option if you want to tell RPM not to check any dependencies before installing the package:  # **rpm -i software-2.3.4.rpm --nodeps**  This forces RPM to ignore dependency errors and install software anyway, but note that if the needed files are missing anyway, the program won't work well or won't work at all. Use the --nodeps option only when you know what you're doing or when you're bone-headed enough ;-) < Removing software installed with RPM > To remove software that was installed with RPM, you use the -e option (which stands for "erase"):  # **rpm -e software-2.3.4**  Note that when installing software, you have to type the name of the RPM package. But when removing software, you don't have to type the whole name of the package that contained the software. You don't have to type the .rpm extension when removing software. Probably you don't have to type the version number, either, so this would do exactly the same as the above: # **rpm -e software**  This rpm -e command uses the RPM database to check where all the files related to this software were installed and then automatically removes all of those files. After removing the program files, it also removes the program from the database of installed software.  This is why it's so important you NEVER remove RPM software manually (for example, deleting single files with rm). If you just run around your system randomly deleting files that were installed with RPM, you'll get rid of the software but RPM doesn't know it and doesn't remove the software package from its database. The result is that RPM still thinks the program is installed on your system, and you may run into dependency problems later.  If you used RPM for installing a certain piece of software, use RPM for removing that piece of software, too! < Querying the RPM database > As you already know, the RPM database contains a list of all installed RPM packages on your system. You can query this database to get info of the packages on your Linux system. To query a single package, you use the -q option. For example, to query a package whose name is "software": # **rpm -q software**  After issuing this command, rpm either tells you the version of the package, or that the package isn't installed.  If you want a list of all packages installed on your system, you'll have to query all with -qa: # **rpm -qa**  Most likely this list will be very long, so you'll need a way to scroll it. The best way is to pipe the list to less: # **rpm -qa | less**  If you're looking for packages whose names contain a specific word, you can use grep for finding those packages. For example, to get a list of all installed RPM packages whose names contain the word "kde", you could do something like this: # **rpm -qa | grep kde**  The above command makes rpm list all packages in its database and pass the list to grep. Then grep checks every line for "kde" and finally shows you all the lines that contain the word "kde". | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No: 07** |
| **Experiment Title:** C/C++ program using cc / gcc | | |
| **Aim:** C/C++ program using cc / gcc.  **Theory:**  **Step 1:** Open the Terminal  **Step 2:** Write command#**gedit** file\_name**.c**  (editor will open the file in write mode if exists otherwise it will create new  file )  **Step 3:** Write C/C++ program in linux editor & save the file with extension c or cpp  respectively.  **Step 4:** save the program and close the window.  **Step 5**: Write command#**gcc** file\_name**.cpp or** #**cc** file\_name.**c**  (program will be compiled if code is correct otherwise it will show the  errors)  **Step 6**: For Output write command **./a.out**  (It will show output on the terminal screen) | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No: 08** |
| **Experiment Title:**  Configuring Apache | | |
| **Aim:** Configuring apache Server  **Theory:**  **Configuration**  Apache has three main configuration files: ***access.conf***, ***httpd.conf***, and ***srm.conf***. If you are running Red Hat 4.0, these files will already be set with the correct directory paths. If you centralized the locations of all these files, but made those symbolic links as I mentioned above, things will still be fine since the symbolic links preserves where Red Hat installed everything.If you are doing a "generic" installation or have some other setup, then you will need to do the following:  In ***access.conf***, change/update these directory entries:  <Directory /httpd/html>  <Directory /httpd/cgi-bin>  In ***httpd.conf***:  ServerRoot /httpd  In ***srm.conf***:  DocumentRoot /httpd/html  Alias /icons/ /httpd/icons/  ScriptAlias /cgi-bin/ /httpd/cgi-bin/  Essentially, these are the necessary directives in the config files that need to be updated with the new    "centralized" organization.   * **Configuring PHP with Apache**   To **configure PHP** copy php.ini-dist which is in the PHP src directory to/usr/local/lib/php.ini Edit this file setting the options you wish, generally nothing needs to be edited. However, you can set various options such as a default MySQL username and password.  To **configure Apache** edit /usr/local/apache/conf/httpd.conf and set your document directory and any other Apache settings you may want. To enable Apache and PHP to work together the following line needs to be added:  AddType application/x-httpd-php .php  Look for this line or something similar already in the httpd.conf file and replace it with the above. Make sure to remove the # comment mark.  After editing the config file you need to **restart Apache**. The command to restart Apache is:  /etc/rc.d/init.d/httpd restart  To test that Apache and PHP work together, create a PHP file. Name it **test.php**. The entire contents of the file are:  <? php echo("<h1>Yay! PHP and Apache Work! </h1>"); ?> | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No:09** |
| **Experiment Title:** Write aPHP script for other task. | | |
| **Aim:** Write aPHP script for sending mail from gmail.  **Theory:**  **Create library file for the SMTP Settings 'library.php':**  <?php  error\_reporting(0);  define("SMTP\_HOST", "SMTP\_HOST\_NAME"); //Hostname of the mail server  define("SMTP\_PORT", "SMTP\_PORT"); //Port of the SMTP like to be 25, 80, 465 or 587  define("SMTP\_UNAME", "VALID\_EMAIL\_ACCOUNT"); //Username for SMTP authentication any valid email created in your domain  define("SMTP\_PWORD", "VALID\_EMAIL\_ACCOUNTS\_PASSWORD"); //Password for SMTP authentication  ?>  **Create new page called 'sendmail.php' for send SMTP mails:**  <form action="" method="post">      <table>      <tr>          <td><input type="email" placeholder="Email" name="email" /></td>      </tr>      <tr>          <td><input type="submit" name="send" value="Send via SMTP" /></td>      </tr>      </table>  </form>  **Make the form post and do the below actions:**  <?php  include 'library.php';  include "classes/class.phpmailer.php"; // include the class file name  if(isset($\_POST["send"])){      $email = $\_POST["email"];      $mail   = new PHPMailer; // call the class      $mail->IsSMTP();      $mail->Host = SMTP\_HOST; //Hostname of the mail server      $mail->Port = SMTP\_PORT; //Port of the SMTP like to be 25, 80, 465 or 587      $mail->SMTPAuth = true; //Whether to use SMTP authentication      $mail->Username = SMTP\_UNAME; //Username for SMTP authentication any valid email created in your domain      $mail->Password = SMTP\_PWORD; //Password for SMTP authentication      $mail->AddReplyTo("reply@yourdomain.com", "Reply name"); //reply-to address      $mail->SetFrom("from@yourdomain.com", "Asif18 SMTP Mailer"); //From address of the mail      // put your while loop here like below,      $mail->Subject = "Your SMTP Mail"; //Subject od your mail      $mail->AddAddress($email, "Asif18"); //To address who will receive this email      $mail->MsgHTML("<b>Hi, your first SMTP mail has been received. Great Job!.. <br/><br/>by <a href='[http://asif18.com](http://asif18.com/)'>Asif18</a></b>"); //Put your body of the message you can place html code here      $mail->AddAttachment("images/asif18-logo.png"); //Attach a file here if any or comment this line,      $send = $mail->Send(); //Send the mails      if($send){          echo '<center><h3 style="color:#009933;">Mail sent successfully</h3></center>';      }      else{          echo '<center><h3 style="color:#FF3300;">Mail error: </h3></center>'.$mail->ErrorInfo;      }  }  ?> | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No:10** |
| **Experiment Title:** MySQL Installation, Configuration and Testing | | |
| **Aim:** MySQL Installation, Configuration and Testing  **Theory:**  **Installing MySQL** 1. Download the latest stable relase of MySQL Download mySQL from [mysql.com](http://dev.mysql.com/downloads/mysql/5.1.html) .  Please download the community edition of MySQL for your appropriate Linux platform. I downloaded the “Red Hat Enterprise Linux 5 RPM (x86)”. Make sure to download MySQL Server, Client and “Headers and libraries” from the download page.   * MySQL-client-community-5.1.25-0.rhel5.i386.rpm * MySQL-server-community-5.1.25-0.rhel5.i386.rpm * MySQL-devel-community-5.1.25-0.rhel5.i386.rpm  2. Remove the existing default MySQL that came with the Linux distro Do not perform this on an system where the MySQL database is getting used by some application.  [local-host]# **rpm -qa | grep -i mysql**  mysql-5.0.22-2.1.0.1  mysqlclient10-3.23.58-4.RHEL4.1  [local-host]# **rpm -e mysql --nodeps**  warning: /etc/my.cnf saved as /etc/my.cnf.rpmsave  [local-host]# **rpm -e mysqlclient10** 3. Install the downloaded MySQL package Install the MySQL Server and Client packages as shown below.  [local-host]# **rpm -ivh MySQL-server-community-5.1.25-0.rhel5.i386.rpm MySQL-client-community-5.1.25-0.rhel5.i386.rpm**  Preparing...                ########################################### [100%]  1:MySQL-client-community ########################################### [ 50%]  2:MySQL-server-community ########################################### [100%]  This will also display the following output and start the MySQL daemon automatically.  PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !  To do so, start the server, then issue the following commands:  /usr/bin/mysqladmin -u root password 'new-password'  /usr/bin/mysqladmin -u root -h medica2 password 'new-password'  Alternatively you can run:  /usr/bin/mysql\_secure\_installation  which will also give you the option of removing the test  databases and anonymous user created by default.  This is  strongly recommended for production servers.  See the manual for more instructions.  Please report any problems with the /usr/bin/mysqlbug script!  The latest information about MySQL is available at http://www.mysql.com/  Support MySQL by buying support/licenses from http://shop.mysql.com/  **Starting MySQL.[  OK  ]**  Giving mysqld 2 seconds to start  Install the “Header and Libraries” that are part of the MySQL-devel packages.  [local-host]# **rpm -ivh MySQL-devel-community-5.1.25-0.rhel5.i386.rpm**  Preparing...                ########################################### [100%]  1:MySQL-devel-community  ########################################### [100%] | | |

|  |  |  |
| --- | --- | --- |
| **Mahatma Gandhi Mission’s**  **Jawaharlal Nehru Engineering College, Aurangabad** | | |
| **Practical Experiment Instruction Sheet** | | |
| **Department:** CSE | **Subject:** Open Source Lab  **Class:** S.E. (CSE) | **Experiment No:11** |
| **Experiment Title:** Design of admission form using PHP – MYSQL | | |
| **Step1:** Create HTML form for Admission.  **Step2:** Create PHP script to accept the information from the given HTML page.  **Step3:** Create database myDB and table MyGuests.  **Step4:** Insert the data using following Script (as per the fields script varies). Example (MySQL Object-oriented) <?php $servername = "localhost"; $username = "username"; $password = "password"; $dbname = "myDB";  // Create connection $conn = new mysql($servername, $username, $password, $dbname); // Check connection if ($conn->connect\_error) {     die("Connection failed: " . $conn->connect\_error); }   $sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe', 'john@example.com')";  if ($conn->query($sql) === TRUE) {     echo "New record created successfully"; } else {     echo "Error: " . $sql . "<br>" . $conn->error; } $conn->close(); ?> | | |