1. Connecting to the Unix Server

## 1.1: Connecting to the Unix Server

**Step 1:** Enter your login name and password to login to the UNIX system.

**Login- BI\_ATP19**

**Password-> BI\_ATP19**

## 1.2: Logging out of the system

**Step 1**: Type the exit command at $ prompt or else, press ctrl and d together to log out.

**$ exit**

**Or ‘ctrl + D’**

1. Unix Basic Command

## 2:1 Executing basic commands:

1. To display the current working directory, the command is:

**$ pwd**

The output is as follows.

**/home/BI\_ATP19**

1. Display the path to and name of your HOME directory.

**$ cd ..**

**[BI\_ATP19@NDAUNIX home]$**

1. Display the login name using which you have logged into the system

**$ logname**

1. Display the hidden files of your current directory.

**$ ls –a**

1. List the names of all the files in your home directory.

**$ ls**

1. Using the long listing format to display the files in your directory.

**$ ls –l**

1. List the files beginning with chap followed by any number or any lower case alphabet. (Example, it should display all files whose names are like chap1, chap2, chap3 ……., chapa,ahapb,chapc,……..)

**$ ls chap\***

1. Give appropriate command to create a directory called C\_prog under your home directory. (Note: Check the directory using ls)

**$ mkdir C\_prog**

1. Create the following directories under your home directory. (Note: Check using ls)

newdir

newdirectory

**$ mkdir newdir**

**$ mkdir newdirectory**

1. List the names of all the files, including the contents of the sub directories under your home directory.

**$ ls –R**

1. Remove the directory called newdirectory from your working directory.

.

**$ rmdir newdirectory**

1. Create a directory called temp under your home directory.

.

**$ mkdir temp**

1. Remove the directory called newdir under your home directory and verify the above with the help of the directory listing command.

.

**$ rmdir newdir**

**$ ls –d \*/**

1. Create another directory directorynew under the temp directory.

.

**$ cd temp**

**$ mkdir directorynew**

1. Change the directory to your home directory.

.

**$ cd ..**

1. From your home directory, change the directory to directorynew using relative and absolute path.

**$ cd temp/directorynew** 🡪 absolute path

**$ cd temp**

**$ cd directorynew**

1. Remove the directory called c\_prog, which is in your home directory.

.

**$ rmdir C\_prog**

1. Change to the directory /etc and display the files present in it.

.

**$ cd /etc**

**$ ls**

1. List the names of all the files that begin with a dot in the /usr/bin directory.

.

**$ cd /usr/bin**

**$ ls .\***

1. Create a file first.unix with the following contents.

Hi! Good Morning everybody.

Welcome to the First exercise on UNIX.

Hope you enjoy doing the assignments.

$ cd

**$ cat > first.unix**

Hi! Good Morning everybody.

Welcome to the First exercise on UNIX.

Hope you enjoy doing the assignments.

1. Copy the file first.unix in your home directory to first.unics.

(Note: checked using ls, first.unix file also should exist along with first.unics)

**$ cp first.unix first.unics**

1. List the contents of first.unix and first.unics with a single command.

.

**$ cat** **first.unix first.unics**

1. Create a new directory under the temp directory.

.

**$cd temp**

**$ mkdir newdirectoty**

1. From your home directory, copy all the files to the directory created under the temp sub directory.

.

$ cp \* /home/BI\_13JUN\_B1\_BLR9/temp/newdirectory/

1. Move the file first.unix to the directory temp as second.unix

.

**$ mv first.unix ~/temp/second.unix**

1. Remove the file called first.unics from the home directory.

.

**$ cd**

**$ rm first.unics**

1. Change your directory to temp and issue the command rm \*. What do you observe?

.

**$ cd temp**

**$ rm \***

1. Move all files whose names end with a, c and o to the HOME directory.

.

**$ cp `ls | grep -i '[\*a|\*c|\*o]$’’`~**

1. Copy all files that end with a ‘UNIX’ to the temp directory.

.

**$ cp \*unix ~/temp**

1. Issuing a single command, remove all the files from the directory temp and the directory itself.

.

**$ rm –R ~/temp**

1. Try commands cp and mv with invalid number of arguments and note the results.

.

**cp achap chap2 newfile**

**cp: target `newfile' is not a directory**

**mv achap chap2 newfile**

**mv: target `newfile' is not a directory**

1. Use the cat command to create a file friends, with the following data:

Madhu 6966456 09/07/68

Jamil 2345215 08/09/67

Ajay 5546785 01/04/66

Mano 7820022 09/07/68

David 8281292 09/09/60

Simmi 7864563 12/12/70

Navin 2224311 30/05/68

The fields should be separated by a tab.

.

**$ cat > friends**

Madhu 6966456 09/07/68

Jamil 2345215 08/09/67

Ajay 5546785 01/04/66

Mano 7820022 09/07/68

David 8281292 09/09/60

Simmi 7864563 12/12/70

Navin 2224311 30/05/68

1. Display contents of the file friends.

.

**$ cat friends**

1. Copy contents of friends to newfriend without using the cp command.

.

**$ cat friends > newfriends**

1. Display contents of the file friends and newfriends in a single command.

.

**$ cat friends newfriends**

1. Find all users currently working on the system and store the output in a file named as users.

.

**$ who > users**

**$ cat users**

1. Append contents of friends file to the file, users.

.

**$ cat friends >> users**

**$ cat users**

1. Display current system date and time and record your observations. How is the time displayed?

.

**$ date**

1. Display calendar for the month and year of your birth.

.

**$ cal 3 95**

1. Try following commands and record your observations.

date “+ %”

date “+%m”

date “+%D”

date “+%/%Training Activity”

date “+%Training Activity”

date “+%r”

1. UNIX File System & Permissions

## 3.1: Viewing the File System and Granting/Removing Permissions

## (Note: Create required files if doesn’t exists.)

1. Give the execute permission for the user for a file chap1

.

**$ chmod u+x chap1**

1. Give the execute permission for user, group and others for a file add.c

.

**$ chmod u,g,o+x add.c**

**Or**

**$ chmod a+x add.c**

1. Remove the execute permission from user, give read permission to group and others for a file aa.c

.

**$ chmod u-x,g+r,o+r aa.c**

1. Give execute permission for users for a.c, kk.c, nato and myfile using single command

.

**$ touch a.c kk.c nato myfile**

**$ chmod u+x a.c kk.c nato myfile**

**$ ls –l a.c kk.c nato myfile**

1. Change the directory to root directory. Check the system directories, like bin, etc, usr etc

.

**$ cd /**

**$ ls –l bin etc usr**

1. Simple and Advance Filetrs

|  |  |
| --- | --- |
| **Goals** | * Learn to use Pipes & Filters in UNIX |
| **Time**  **Lab Setup** | 100 min  Telnet with Unix Server |

## 4.1: Using Pipes and Filters:

1: Redirect the content of the help document ls, into a file called as lsdoc.

**$ ls –help > lsdoc**

2: Display the content of the lsdoc page wise.

**$ cat lsdoc | more**

3: Display only the first 4 lines of the lsdoc file.

**$ head –n 5 lsdoc**

4: Display only the last 7 lines of the file lsdoc.

**$ tail –n 7 lsdoc**

5: Remove the file lsdoc.

**$ rm lsdoc**

6: There will be B’day celebration from the friends file, find how many B’day parties will be held. If two of the friends have the B’date on the same day, then we will be having one party on that day.

**$ cat friends | cut -f3 | sort | uniq | wc –l**

7: Display the lines starting with Ma, in the file friends.

**$ grep "^Ma" friends**

8: Display the lines starting with Ma, ending with i or ending with id, in the file friends.

**cat friends | grep -i '^Ma' | grep -i '[id|i]$'**

9: Print all the files and the directory files from the current directory across all the sub directories, along with its path

**$ ls –Rl $pwd/\***

10: Print only the Directory files.

**$ ls -l -d \*/**

11: Display the files starting with chap, along with its path.

**ls –l $PWD/chap\***

12: Sort the file friends in ascending order of names.

**$ sort -k1 friends**

13: Display the contents of the file friends in uppercase letters.

**sort -t" " -n -k123 friends**

14: Store the contents of your home directory in a file called dir.

**cd ~**

**mkdir dir**

**cp \* dir**

15: From the above file dir, display the file permissions and the name of the file only.

**ls –ld dir**

16: From the same dir file, store only the file names in a file called files.

**ls –p|grep –v “./” > files**

17: From the same dir file, store only the permissions of files in a file called perms.

**ls -l|cut -d" " -f1 > perms**

18: From the same dir file, store only the file sizes in a file called sizes.

**ls -l|cut -d" " -f5>sizes**

19: Display the file names, sizes and permissions from your directory in that order.

**paste files sizes perms**

20: Display the number of users working on the system.

**Who | ls –l**

21: Find out the smallest file in your directory.

**Who | ls –l**

22: Display the total number of lines present in the file friends.

**wc -l < friends.txt**

23:Create the following fixed record format files (with “|” delimiter between fields) with the structure given below, and populate them with relevant data use these files to solve following questions

emp.lst: Empid(4),Name(18),Designation(9),Dept(10),Date of Birth(8),Salary(5)

dept.lst : Dept.Code(2),Name(10),Head of Dept’s id(4)

desig.lst: Designation Abbr.(2), Name (9)

1. Find the record lengths of each file.
2. Display only the date of birth and salary of the last employee record.
3. Extract only employee names and designations. (Use column specifications). Save output as cfile1.
4. Extract Emp.id, dept, dob and salary. (Use field specifications). Save output as cfile2.
5. Fix the files cfile1 and cfile2 laterally, along with the delimiter.
6. Sort the emp.lst file in reverse order of Emp. Names.
7. Sort the emp.lst file on the salary field, and store the result in file srtf.
8. Sort the emp.lst file on designation followed by name.
9. Sort the emp.lst file on the year of birth.
10. Find out the various designations in the employee file. Eliminate duplicate listing of designations.
11. Find the non-repeated designation in the employee file.
12. Find the number of employees with various designations in the employee file.

**23.1. cat emp | wc –l**

**23.2. cut –d”|” –f5,6 emp|tail –n 1**

**23.3. cut –d”|” –f2,3 > cfile1**

**23.4. cut –d”|” –f1,4,5,6 > cfile2**

**23.5. paste cfile1 cfile2**

**23.6. sort –t”|” –n –r –k2 emp**

**23.7. sort –t”|” –n –k6 > srtf**

**23.8. sort –t”|” –n –k3,2**

**23.9. sort –t”|” –n –k5**

**23.10. cut –d”|” –f3 | uniq**

**23.11. cut –d”|” –f4 | uniq -u**

**23.12. cut –d”|” –f3 | uniq | wc -l**

1. Vi Editor

## 5.1: Working wth Vi Editor

1. Create a file using Vi. Enter the following text:

*A network is a group of computers that can communicate with each other, share resources, and access remote hosts or other networks. Netware is a computer network operating system designed to connect, manage, and maintain a network and its services. Some of the network services are Netware Directory Services (NDS), file system, printing and security.*

$ vi demo.sh

1. Change the word “Netware” in the second line to “Novell Netware”.
2. Insert the text “(such as hard disks and printers)” after “share resources” in the first line.
3. Append the following text to the file:

“Managing NDS is a fundamental administrator role because NDS provides a single point for accessing and managing most network resources.”

1. Shell Script

|  |  |
| --- | --- |
| **Goals** | Learn to write simple shell scripts |
| **Time**  **Lab Setup** | 150 min  Telnet with Unix Server |

## 6.1: Writing Shell-Scripts

1. Display the Primary and Secondary prompt. Change the primary prompt to your name: temporarily.

.

**Echo $PS1**

**Echo $PS2**

**PS1=”Ayush :”**

2: As soon as you login, the prompt should be changed to your name: also the name of the home directory should be automatically displayed.

Q

3: Check the content of the Environmental variable SHELL.

**echo $SHELL**

4: Try the below exercise and check the output.

**Note**: Type every line and press enter, do not type the entire code in a vi editor.

$continent=”Africa”

$echo “$continent”

------------🡪 Africa

$sh

$echo “$continent”

------------🡪 No Response

$continent=”Asia”

$echo “$continent”

------------🡪 Asia

$ctrl + d

$echo “$continent”

------------🡪 Africa

$sh

$echo “$continent”

------------🡪 No Response

$ctrl + d

5: Try the below exercise and check the output. (Export variables)

**Note**: Type every line and press enter, do not type the entire code in a vi editor.

$continent=”Africa”

export continent

$echo “$continent”

------------🡪 Africa

$sh

$echo “$continent”

------------🡪 Africa

$continent=”Asia”

$echo “$continent”

------------🡪 Asia

$ctrl + d

$echo “$continent”

------------🡪 Africa

6: Write a shell script that takes the user name as input and reports whether he / she has logged in or not.

$ vi lab6-6.sh

**echo Enter User Name**

**read un**

**who | cut -d " " -f 1 | grep -w "$un"**

**if [ $? -eq 0 ]**

**then**

**echo $un is Loged In.**

**else**

**echo Not Loged In.**

**fi**

7: Write a shell script to display the file name and its contents of all the files that is there in the current directory.

$ vi lab6-7.sh

**clear**

**for n**

**echo “All the files are:”**

**ls –FR**

8: Write a shell script, which will take a file name as argument and check whether the file exists and display its access permissions for user.

$ vi lab6-8.sh

**echo Filename - $1**

**if [ -s "$1" ]**

**then**

**echo " File exists. "**

**echo Permision of File $1 is**

**ls -l $1 | cut -d " " -f 1**

**else**

**echo $1 doesnot exists.**

**fi:**

9: Pass three numbers as command line arguments and display the largest number in the given three numbers.

$ vi lab6-9.sh

**echo Enter three numbers :- $1 $2 $3**

**if [ $1 -ge $2 -a $1 -ge $3 ]**

**then**

**echo $1 is the largest number.**

**elif [ $2 -ge $1 -a $2 -ge $3 ]**

**then**

**echo $2 is the largest number.**

**else**

**echo $3 is the largest number.**

**Fi**

10: Write a shell script which will accept a pattern and a file name. The pattern will be searched in the file provided. Display appropriate messages and perform necessary validations on file.

$ vi lab6-10.sh

**echo Pattern is $1**

**echo Filename is $2**

**if [ ! -f "$2" ]**

**then**

**echo File doesnot exists.**

**else**

**grep -n "$1" $2**

**fi**

11: To create a menu program for a) creating a file, b) Creating a directory, c) copying a file, d) moving a file. (use functions)

* 1. If the file exists already give the appropriate message
  2. If the dir exists already give the appropriate error message
  3. Source file should exist if not give a message, It should have read permission if not another message, Destination file either there or not, if not there then create it and copy it. If there, then ask whether to overwrite or not, if yes then overwrite it or else give a message file exists already and not overwritten.

$ vi lab6-11.sh

**echo All your options are :**

**echo "a. Creating a File"**

**echo "b. Creating a Directory"**

**echo "c. Copying a File"**

**echo "d. Moving a File"**

**echo " Enter Choise (a,b,c,d) : \c"**

**read choise**

**case $choise in**

**a) echo Enter file Name:**

**read fn**

**if [ -f "$fn" ]**

**then**

**echo Sorry ! A file is already exixts with $fn name.**

**else**

**cat $fn**

**echo File created.**

**fi ;;**

**b) echo Enter Directory Name:**

**read dn**

**if [ -d "$dn" ]**

**then**

**echo Sorry ! A Directory is already exixts with $dn name.**

**else**

**mkdir $dn**

**echo Directory Created.**

**fi;;**

**c) echo Enter Source File**

**read sf**

**echo Enter Destination File**

**read df**

**if [ -z "$sf" ]**

**then**

**echo Error ! You havent entered Source file.**

**elif [ -z "$df" ]**

**then**

**echo Error ! You havent entered Destination File.**

**elif [ ! -r "$sf" ]**

**then**

**echo Source file is not allowed to Read.**

**elif [ -s "$sf" ]**

**then**

**if [ -s "$df" ]**

**then**

**echo Destination File has already some content which is below**

**echo -------**

**cat $df**

**echo -------**

**echo "Want to Overwrite (y/n):"**

**read inp**

**if [ "$inp" = "y" ]**

**then**

**cp $sf $df**

**echo coppied Successfilly.**

**else**

**echo Not coppied**

**fi**

**else**

**echo Something went wrong , Try again.**

**fi**

**else**

**echo Sorry ! Source file Doesnot exist or have no data.**

**fi ;;**

**d) echo Enter Source file**

**read sf**

**if [ ! -f "$sf" ]**

**then**

**echo Source file doesnot Exixts**

**else**

**echo Enter destination file name.**

**read df**

**mv $sf $df**

**echo file moved successfully.**

**fi ;;**

**\*) echo wrong choise !! ;;**

**esac**

12: Write a function yesno() to display question to user and accept answer as y/n. If answer to the question is y the function should return 0 otherwise 1.

Use yesno functions for asking different questions. Question will be passed as parameter to the function.

Accept filename from user check whether it is file or directory. Use yesno() function to display question do you really want to delete file? If the ans is y, then delete the file or directory.

**function yesno()**

**{**

**echo "$1 [y/n]:"**

**read input**

**case $input in**

**y)**

**return 1**

**;;**

**Y)**

**return 1**

**;;**

**n)**

**return 0**

**;;**

**N)**

**return 0**

**;;**

**\*)**

**yesno "y/n?"**

**return $?**

**esac**

**}**

13: Write a shell script to store names of four employees and check whether those employees are currently logged in or not. Display appropriate message.

========= ======== =========

$ vi lab6-13.sh

**echo Enter Four Employee Name**

**echo 1.**

**read emp1**

**echo 2.**

**read emp2**

**echo 3.**

**read emp3**

**echo 4.**

**read emp4**

**st=`who | cut -d " " -f 1 | grep "$emp1" | wc -l`**

**if [ $st -eq 1 ]**

**then**

**echo $emp1 is logedin.**

**else**

**echo $emp1 is not logedin.**

**fi**

**st=`who | cut -d " " -f 1 | grep "$emp2" | wc -l`**

**if [ $st -eq 1 ]**

**then**

**echo $emp2 is logedin.**

**else**

**echo $emp2 is not logedin.**

**fi**

**st=`who | cut -d " " -f 1 | grep "$emp3" | wc -l`**

**if [ $st -eq 1 ]**

**then**

**echo $emp3 is logedin.**

**else**

**echo $emp3 is not logedin.**

**fi**

**st=`who | cut -d " " -f 1 | grep "$emp4" | wc -l`**

**if [ $st -eq 1 ]**

**then**

**echo $emp4 is logedin.**

**else**

**echo $emp4 is not logedin.**

**fi**

14: Accept the user's first and last name and the echo the entire name along with some suitable comment.

$ vi lab6-14.sh

**echo Enter First Name**

**read fn**

**echo Enter Last Name**

**read ln**

**echo HI $fn $ln , Welcome to have you in Unix lab.**

15: List all files that have been modified today.

$ vi lab6-15.sh

**echo List of files which modified today.**

**d=`date | cut -d' ' -f2- | cut -c-6`**

**f=`ls -l | grep "$d"`**

**echo $f**

16: Display long listing of only the regular files in the current directory.

$ vi lab6-16.sh

**var=$(ls -lp | grep -v '[/]$')**

**echo $var**

17: Display details of all files in the 2 “paths” accepted from user. The display should be screen by screen.

**if [ $# -ne 2 ]**

**then**

**echo "Enter the paths as command line argument"**

**exit;**

**fi**

**echo “all the files in the first path:”**

**ls –l $1**

**echo “all the files from second path:”**

**ls –l $2**

18: Let the script display its name and its PID.

$ vi lab6-18.sh

**echo Name of Script is - $0**

**echo PID is - $$**

19: Get the concatenated o/p of 2 files into a third file: Take 3 command line arguments: The first argument is the name of a destination file, and the other two arguments are names of files whose contents are to be placed in the destination file.

$ vi lab6-19.sh

**echo Your Destination File is - $1**

**echo Two Source Files are $2 , $3**

**echo Two continue Press (y):**

**read ans**

**if [ "$ans" = "y" ]**

**then**

**if [ ! -f "$2" ]**

**then**

**echo Source File $2 does not exixts.**

**elif [ ! -f "$3" ]**

**then**

**echo Source File $3 does not exixts.**

**elif [ -s "$3" ]**

**then**

**echo Destination File $1 has already some content.**

**echo "Want to Overwrite and Paste (y/n):"**

**read inp**

**if [ "$inp" = "y" ]**

**then**

**cat $2 $3 > $1**

**echo Coppied Successfilly.**

**else**

**echo Not coppied**

**fi**

**else**

**cat $2 $3 > $1**

**echo File $2 and $3 are copied into File $1.**

**else**

**echo wrong choise**

**fi**

**Stretched Assignments:**

20: Write a menu driven shell program to:

a. Display calendar of current month

1. Search for a pattern in all the files/subdirectories from current directory.
2. Count the no. of directories / sub directories in current directory

========= ======== =========

**$ vi lab6-20.sh**

**echo Hi user, What you want from this script.**

**echo a) Display calendar of current month.**

**echo b) Search for a pattern in all the files/subdirectories from current directory.**

**echo c) Count the no. of directories / sub directories in current directory.**

**echo “Enter Your Option (a,b,c) : \c”**

**read choise**

**case $choise in**

1. **cal ;;**
2. **Enter Pattern**

**read ptrn**

**grep –n “$ptrn” \* ;;**

1. **ls | wc –l ;;**

**\*) echo Wrong Input !! ;;**

**esac**

21: Display day of week for a given date. (ddmmyyyy)

If day is Monday, display message “Monday Blues”

Friday display message “yeh! It’s week end.”

Similarly display different messages for each day of the week.

22: Display the contents of all .lst files in the current directory.

23: Design a simple calculator, which will add/subtract/multiply/divide 2 numbers.

eg. cal 10 20 + will give o/p as 30.

$ vi lab6-23.sh

**case $3 in**

**+) let ans=$1+$2**

**echo $ans ;;**

**-) let ans=$1-$2**

**echo $ans ;;**

**/) let ans=$1/$2**

**echo $ans ;;**

**\*) let ans=$1\*$2**

**echo $ans ;;**

**esac**