--The Unix operating system is a set of programs that act as a link between the computer and the user
--Users communicate with the kernel through a program known as the shell.
--The shell is a command line interpreter; it translates commands entered by the user and
converts them into a language that is understood by the kernel
--Several people can use a Unix computer at the same time; hence Unix is called a multiuser system
--A user can also run multiple programs at the same time; hence Unix is a multitasking environment
--Kernel - The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management
--Files and Directories - All the data of Unix is organized into files. All files are then organized into directories.
These directories are further organized into a tree-like structure called the filesystem
--To login, userid and password is case-sensitive

--Who is Logged in?
Sometime you might be interested to know who is logged in to the computer at the same time
you can use any one of the command users, who, and w.

File Mgmt
==========
--Listing Directories and Files
To list the files and directories stored in the current directory
$ls

The command ls supports the -l option which would help you to get more information about the listed files
$ls -l
total 19621
drwxrwxr-x 2 amrood amrood 4096 Dec 25 09:59 uml
-rw-rw-r-- 1 amrood amrood 5341 Dec 25 08:38 uml.jpg

Here entries starting with d..... represent directories
First Column - Represents the file type and the permission given on the file. Below is the description of all type of files
Second Column - Represents the number of memory blocks taken by the file or directory
Third Column - Represents the owner of the file. This is the Unix user who created this file
Fourth Column - Represents the group of the owner. Every Unix user will have an associated group
Fifth Column - Represents the file size in bytes.
Sixth Column - Represents the date and the time when this file was created or modified for the last time
Seventh Column - Represents the file or the directory name

--Metacharacters : Metacharacters have a special meaning in Unix. For example, * and ? are metacharacters.
We use * to match 0 or more characters, a question mark (?) matches with a single character
$ls ch*.doc
--To list the invisible files, specify the -a option to ls
$ ls -a
--Creating Files : You can use the vi editor to create ordinary files on any Unix system
$ vi filename
--Editing Files: You can edit an existing file using the vi editor
$ vi filename
l key to move to the right side.
h key to move to the left side.
k key to move upside in the file.
j key to move downside in the file.
--Display Content of a File
$ cat filename
You can display the line numbers by using the -b option along with the cat command as follows
$ cat -b filename
--Counting Words in a File
You can use the wc command to get a count of the total number of lines, words, and characters contained in a file
$ wc filename
2 19 103 filename
First Column - Represents the total number of lines in the file.
Second Column - Represents the total number of words in the file.
Third Column - Represents the total number of bytes in the file. This is the actual size of the file.
Fourth Column - Represents the file name
-You can give multiple files and get information about those files at a time
$ wc filename1 filename2 filename3
--Copying Files
$ cp source_file destination_file
--Renaming Files
$ mv old_file new_file
--Deleting Files
$ rm filename
You can remove multiple files at a time
$ rm filename1 filename2 filename3

```
Directory Management
====================
--Home Directory
  -The directory in which you find yourself when you first login is called your home directory
   $cd ~
  -Suppose you have to go in any other user's home directory
   $cd ~username
  -To go in your last directory
    $cd -
--To list the files in a directory
  $ls dirname
--Creating Directories
  $mkdir dirname
  $mkdir dirname1, dirname2
  $mkdir /tmp/dirname  (create a directory in tmp)
  -Creating Parent Directories
    $mkdir -p /tmp/dirname  (create a two directories)
--Removing Directories
   $rmdir dirname
   $rmdir dirname1 dirname2 dirname3
--Renaming Directories
   $mv olddir newdir

Sending Email
=============
You can use the Unix mail command to send and receive mail
send an email  : $mail [-s subject] [-c cc-addr] [-b bcc-addr] to-addr
To check incoming email: $mail

Processes Management
====================
$ps  Listing Running Processes
   One of the most commonly used flags for ps is the -f ( f for full) option, which provides more information

$kill -9 <pid> To kill the process

Network Communication Utilities
===============================
$ping hostname or ip-address
$scp -r sourcedir/ user@dest.com:/dest/dir/
-The ftp Utility : ftp stands for File Transfer Protocol.
     This utility helps you upload and download your file from one computer to another computer
    $ftp hostname or ip-address
           then provide login id and password
           >ftp command will shown
        sub commands:- put filename : Uploads filename from the local machine to the remote machine
                              get filename : Downloads filename from the remote machine to the local machine
                                          mget and mput to download and upload more than once

The vi Editor Tutorial
==========================
-Starting the vi Editor
  vi filename : Creates a new file if it already does not exist, otherwise opens an existing file
  vi -R filename : Opens an existing file in the read-only mode
  view filename : Opens an existing file in the read-only mode

-Moving within a File
  k: Moves the cursor up one line
  j: Moves the cursor down one line
  0 or | : Positions the cursor at the beginning of a line
  $ : Positions the cursor at the end of a line
  1G : Moves to the first line of the file
  G : Moves to the last line of the file
  nG : Moves to the nth line of the file
  w : Positions the cursor to the next word
  b : Positions the cursor to the previous word
  M : Moves to the middle of the screen
  L : Move to the bottom of the screen

-Control Commands
  CTRL+d : Moves forward 1/2 screen
  CTRL+f : Moves forward one full screen
  CTRL+u : Moves backward 1/2 screen
  CTRL+b : Moves backward one full screen

-Editing Files
```

```
   -To edit the file, you need to be in the insert mode.
    There are many ways to enter the insert mode from the command mode
   i : Inserts text before the current cursor location
   I : Inserts text at the beginning of the current line
   a : Inserts text after the current cursor location
   A : Inserts text at the end of the current line
   o : Creates a new line for text entry below the cursor location
   O : Creates a new line for text entry above the cursor location

 -Deleting Characters
  x : Deletes the character under the cursor location
  X : Deletes the character before the cursor location
  dw : Deletes from the current cursor location to the next word
  d^ : Deletes from the current cursor position to the beginning of the line
  d$ : Deletes from the current cursor position to the end of the line
  D : Deletes from the cursor position to the end of the current line
  dd : Deletes the line the cursor is on

 -Word and Character Searching
   The / command searches forwards (downwards) in the file.
   The ? command searches backwards (upwards) in the file
    (The n and N commands repeat the previous search command in the same or the opposite direction, respectively)

 Shell Programing
 ================
  Shell Scripts : The first line your script should be #!/bin/sh. Means execute the shell script
  -------------   This is called shebang . # symbol is called a hash, and the ! symbol is called a bang

     -To execute a program available in the current directory, use ./program_name
      Ex : Test.sh
                 #!/bin/sh

                 # Author : Zara Ali
                 # Copyright (c) Tutorialspoint.com
                 # Script follows here:

                 echo "What is your name?"
                 read PERSON
                 echo "Hello, $PERSON"
          o/p $./test.sh
                          What is your name?
                          Zara Ali
                          Hello, Zara Ali

     Shell Variables : The name of a variable can contain only letters (a to z or A to Z), numbers ( 0 to 9) or
 the underscore character ( _).
          --------------    By convention, Unix shell variables will have their names in UPPERCASE
              -Syntax is variable_name=variable_value
                   ex : VAR1="Zara Ali"
                            VAR2=100
           syntax : readonly variable_name (value can't changed)
                    syntax: unset variable_name  (value will be reset with empy)
                    Note : You cannot use the unset command to unset variables that are marked readonly
           -We can access the variable using $ ex : echo $VAR1

          Special Variables : $0 : The filename of the current script.
     -----------------   $n : arguments ex : $1 is first argument and $2 is second argument and so on
                             $# : The number of arguments entered
                             $* : All the arguments are double quoted. If a script receives two arguments, $* is
 equivalent to $1 $2.
                                                  $@ : All the arguments are individually double quoted. If a
 script receives two arguments, $* is equivalent to $1 $2.
                          $$ : The process number PID of the current shell
                          $? : The exit status of the last command executed
                                                  $! : The process number of the last background command
                      Ex : test.sh
                                             #!/bin/sh

                                                  echo "File Name: $0"          o/p is test.sh
                                                  echo "First Parameter : $1"   o/p is sree
                                                  echo "Second Parameter : $2"  o/p is vas
                                                  echo "Quoted Values: $@"      o/p is sree vas
                                                  echo "Quoted Values: $*"      o/p is sree vas
                                                  echo "Total Number of Parameters : $#"   o/p is 2
                                                  o/p : ./test.sh sree vas
```

```
    Shell Arrays :  Array initialization syntax is array_name = (value1 ... valuen)
     ------------       Array element assign syntax is array_name[index]=value
                        Array single element access is ${array_name[index]}
                    Array all element access is ${array_name[*]} or ${array_name[@]}

    Shell Basic Operators :
     Arithmetic Operators :  use either awk or expr
     ---------------------   ex : val=`expr 2 + 2` (There must be spaces between operators and expressions.)
                                                              The complete expression should be enclosed between ` `,
called the backtick
        + : `expr $a + $b`
                - :  `expr $a - $b`
        * (Multiplication) : `expr $a \* $b`
        / (Division) : `expr $b / $a`
        % (Modulus) : `expr $b % $a`
                = (Assignment): a = $b would assign value of b into a
                == : [ $a == $b ]
        != : [ $a != $b ]
                Note : All the arithmetical calculations are done using long integers
    Relational Operators :  -eq : [ $a -eq $b ]
        ---------------------   -ne (not equal) : [ $a -ne $b ]
                            -gt : [ $a -gt $b ]
                            -lt : [ $a -lt $b ]
                            -ge : [ $a -ge $b ]
                            -le : [ $a -le $b ]
        Note : Space should be present  For example, [ $a <= $b ] is correct whereas, [$a <= $b] is incorrect
    Boolean Operators  : !(This is logical negation) : [ ! false ] is true
        ------------------   -o (This is logical OR. If one of the operands is true, then the condition becomes
true)
                                    ex :  $a -lt 20 -o $b -gt 100 ] is true
                                                    -a : This is logical AND. If both the operands are true, then
the condition becomes true otherwise false
                            ex : [ $a -lt 20 -a $b -gt 100 ] is false

        String Operators :  = (Checks if the value of two operands are equal or not; if yes, then the condition
becomes true)
    -----------------    ex : [ $a = $b ] is not true
                            != Checks if the value of two operands are equal or not; if values are not equal then
the condition becomes true
                                                    ex : [ $a != $b ] is true
                         -z Checks if the given string operand size is zero; if it is zero length, then it returns
true
                          ex : [ -z $a ] is not true
                         -n : Checks if the given string operand size is non-zero; if it is nonzero length, then it
returns true
                          ex : [ -n $a ] is not false
                        str : Checks if str is not the empty string; if it is empty, then it returns false
                         ex : [ $a ] is not false
   File Test Operators :  file = '/home/liadmin/test.sh' here file is var name
   ---------------------  (assume the file exists in this location)
                      -e file (Checks if file exists) : [ -e $file ] is true.
                      -s file (Checks if file has size greater than 0): [ -s $file ]
                      -r file (Checks if file is readable) : [ -r $file ]
                      -w file (Checks if file is writable) : [ -w $file ]
                      -x file (Checks if file is executable) : [ -x $file ]
                      -d file (Checks if file is a directory)  : [ -d $file ]
                      -f file (Checks if file is an ordinary file) : [ -f $file ]
                      -u file (Checks if file has its Set User ID (SUID) bit set; if yes, then the condition
becomes true) : [ -u $file ]
                      -g file (Checks if file has its set group ID (SGID) bit set; if yes, then the condition
becomes true) : [ -g $file ]
                                      -c file (Checks if file is a character special file; if yes, then the
condition becomes true) : [ -c $file ] is false
                      -b file (Checks if file is a block special file; if yes, then the condition becomes true) :
[ -b $file ]
        Shell Decision Making :  1)if...fi statement
    ----------------------       syntax : if [ expression ]
                                                                        then
                                                                          Statement(s) to be executed if
expression is true
                                                                        fi
                                                        ex : test.sh
                                                               #!/bin/sh
                                                               a=10
                                                               b=20
                                                               if [ $a == $b ]
```

```
                                                                then
                                                                        echo "a is equal to b"
                                                                fi
                2)if...else...fi statement
         syntax : if [ expression ]
                                  then
                                         Statement(s) to be executed if expression is true
                                  else
                                         Statement(s) to be executed if expression is not true
                                  fi
                3)if...elif...fi statement
        syntax : if [ expression 1 ]
                                  then
                                     Statement(s) to be executed if expression 1 is true
                                  elif [ expression 2 ]
                                  then
                                         Statement(s) to be executed if expression 2 is true
                                  elif [ expression 3 ]
                                  then
                                         Statement(s) to be executed if expression 3 is true
                                  else
                                         Statement(s) to be executed if no expression is true
                                  fi
           4)case...esac Statement  (very similar to the switch...case statement in java)
         case word in
                        pattern1)
                                Statement(s) to be executed if pattern1 matches
                        ;;
                        pattern2)
                                Statement(s) to be executed if pattern2 matches
                        ;;
                        pattern3)
                                Statement(s) to be executed if pattern3 matches
                        ;;
                        *)
                                Default condition to be executed
                        ;;
                esac
       ex : test.sh
       #!/bin/sh
                FRUIT="kiwi"
                case "$FRUIT" in
                        "apple") echo "Apple pie is quite tasty."
                        ;;
                        "banana") echo "I like banana nut bread."
                        ;;
                        "kiwi") echo "New Zealand is famous for kiwi."
                        ;;
                esac
    Shell Loop Types:  1)The while loop
    -----------------  syntax : while condition
                                do
                                  statements
                                done
                2)for Loop
          syntax: for var in word1 word2 ... wordN
                                   do
                                     Statement(s) to be executed for every word.
                                   done
                3)until loop
          until condition
                   do
                         Statement(s) to be executed until condition is true
                   done
        )select loop
         select var in word1 word2 ... wordN
                  do
                         Statement(s) to be executed for every word.
                  done
        Shell Loop Control :  1)break statement
    --------------------  2)continue statement

        Substitution   : The shell performs substitution when it encounters an expression that contains one or
more special characters
        -------------
        \\  : backslash
```

```
                \b  : backspace
                \n  : new line
                \t  : horizontal tab
                \r  : carriage return
                ex : test.sh
                  #!/bin/sh
                        a=10
                        echo -e "Value of a is $a \n"
                        o/p is Value of a is 10
                        With out -e option  o/p is Value of a is 10\n
        Variable Substitution :
        ${var} : Substitute the value of var
                ${var:-word} : If var is null or unset, word is substituted for var. The value of var does not
change
                ${var:=word} : If var is null or unset, var is set to the value of word
        ${var:?message} : If var is null or unset, message is printed to standard error. This checks that
variables are set correctly
                ${var:+word} : If var is set, word is substituted for var. The value of var does not change

        Shell Quoting Mechanisms :
                I want print hello; world i have $120 it's time to enjoy
                         echo "hello; world i have $120 it's time to enjoy"   (will not print)
                         echo "hello\; world i have \$120 it\'s time to enjoy"  Will print
                         \ - used for special characters
                         \` - to enable literal backquotes
                         \" to enable embedded double quotes
                         \\ to enable embedded backslashes

        Shell Input/Output Redirections  :
            Output Redirection : syntax : command > filename (write the out put to the file)
                pgm > file Output of pgm is redirected to file
                ex : ls > test (here ls command out put stored in test file in the current directory)
                         Note : test file contain data it will override
                         pgm >> file Output of pgm is appended to file
             ls >> tset (here ls command out put will be appended if the test file has data)

                Input Redirection : syntax : command < filename (read the input from the file)
        ex : wc -l < test (here reading the input from the file to count number of words)
                pgm < file Program pgm reads its input from file
                Here Document : syntax: command << delimiter (read the input until to find specified delimiter)
                Discard the output : Sometimes you will need to execute a command, but you don't want the output
displayed on the screen
                  syntax : command > /dev/null

   Shell Functions : function_name () {
                                              list of commands
                                         }
                ex : test.sh
                #!/bin/sh
                                # Define your function here
                                Hello () {
                                        echo "Hello World"
                                }
                                # Invoke your function
                                Hello
            Pass Parameters to a Function :  ex : test.sh
            #!/bin/sh
                        # Define your function here
                        Hello () {
                           echo "Hello World $1 $2"
                        }
                        # Invoke your function
                        Hello Zara Ali
                        Returning Values from Functions : ex : test.sh
                        #!/bin/sh
                        # Define your function here
                        Hello () {
                                echo "Hello World $1 $2"
                                return 10
                        }
                        # Invoke your function
                        Hello Zara Ali
                        # Capture value returnd by last command
                        ret=$?
                        echo "Return value is $ret"
                        Function Call from Prompt : You can put definitions for commonly used functions inside
```

your .profile.
These definitions will be available whenever you log in and you can use them at the command prompt.

Alternatively, you can group the definitions in a file, say test.sh, and then execute the file in the current shell by typing –
$. test.sh

ex : $ Hello
To remove the definition of a function from the shell, use the unset command with the .f option. This command is also used to remove the definition of a variable to the shell.
$unset .f function_name

Shell Manpage Help : for any command help we can use syntax $man command (ex is $man pwd)

File System Basics :
Directory Structure : / : This is the root directory
/bin : This is where the executable files are located. These files are available to all users
/dev : These are device drivers
/etc : Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages
/lib : Contains shared library files and sometimes other kernel-related files
/boot: Contains files for booting the system
/home: Contains the home directory for users and other accounts
/mnt : Used to mount other temporary file systems
/proc : Contains all processes marked as a file by process number
/tmp : Holds temporary files used between system boots
/usr : Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others
/var : Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
/sbin : Contains binary (executable) files, usually for system administration. For example, fdisk and ifconfig utlitie
/kernel : Contains kernel file

File System : cat filename : Displays a filename
cd dirname : Moves you to the identified directory
cp file1 file2 : Copies one file/directory to the specified location
file filename : Identifies the file type (binary, text, etc)
find filename/dir : Finds a file/director
head filename : Shows the beginning of a file
less filename : Browses through a file from the end or the beginning
ls dirname : Shows the contents of the directory specified
mkdir dirname : Creates the specified directory
more filename : Browses through a file from the beginning to the end
mv file1 file2 : Moves the location of, or renames a file/directory
pwd : Shows the current directory the user is in
rm filename : Removes a file
rmdir dirname : Removes a directory
tail filename : Shows the end of a file
touch filename : Creates a blank file or modifies an existing file or its attributes
whereis filename : Shows the location of a file
which filename : Shows the location of a file if it is in your PATH
df -k : disk space in kilobytes
df -h : disk space in human readable format
du -h : disak usage in human readable format
mount : to see mounted files

User Administration:
useradd : Adds accounts to the system (same command group instead user use group)
usermod : Modifies account attributes (same command group instead user use group)
userdel : Deletes accounts from the system (same command group instead user use group)

System Performance:
netstat : Prints network connections, routing tables, interface statistics, masquerade connections, and multicast memberships
top : Displays system tasks

Compressed Files:
zip zipfileName file1 file2 ..
unzip zipfileName
tar -xvf tarfileName

->Opended file using sudo vi but what is command to go end of a file
command : G

```
->Command to find a file

  find -name "filename"
  This will be case sensitive, meaning a search for "filename" is different than a search for "Filename"
  Search happens in the current directory

  find -iname "filename"
  This will be case insensitive, i.e igonre the file cases

  find -name "*.java" (To search all the java files in the current directoy)
  find -name "abc*"   (To search all the files start with abc in the current directory)

  search filename from the root directory
   find / -name "filename"  (Here / indicates root directory)

   search filename in specific directory
     find /home/liadmin/webapps -name   "filename"

-> command to fine the File Containing a Particular Text String
    grep "search text" *.java  (Search all the java files contaning search test in the current directory)
    grep -r "search text" /home/liadmin (Search the all files recurisively in the specified directory)


->Navigation Command
   cd .
     Current Directory
   cd ..
     Parent Directory
   cd
     Home Directory
   cd -
     To go back to previous directory
   pwd
     Present working directory

-> Create/Edit/Remove command
   vi <File_Name_To Create>
   Note: If the file not exists,It will open VI editor to write the content in File.
         If the file exists, It will open VI editor to write the content with existing File

   cat <File-name>
     It will display the content and return to command prompt.

    rm  <File-name>
      Remove the file name

->      Display the list of Command Fired in Current Session
    history
        It will display the last 10 commands fired
    history 100
        It will display the last 100 command fired
->  Current logged in user command is whoami
->  All logged user command is who

->  command to creat a zip
      zip filename.zip file1 file2 folder2 (and so on)
    To zip folder and its all contens and its sub folders
      zip -r filename.zip folder/*
        To zip specified directory and folder is not current directory
        zip -r /tmp/abc.zip /home/test/*

->  command to unzip
     unzip filename.zip
         List all the files stored in a zip file
         unzip -l abc.zip or less abc.zip

->  command to create a tar file
    tar -cvf filename.tar /foldername
    command to extract a tar file
    tar -xvf filename.tar

-> command to create a Jar file
    jar -cf myfile.jar *.class
    command to extract Jar file
    jar -xf myfile.jar
```

```
     command to execute Jar file
     java -jar myfile.jar

 ->  command to find all running process
     top or ps
         command to find the process by name
     psg <processname> example psg tomcat
     command to kill the process by id
     sudo kill -9 <processid>    (The -9 tells the kill command that you want to send signal #9, which is called
SIGKILL)
     command to kill the process by name
     sudo pkill -9 <processname>
     command to kill all running the processes by name
         sudo killall -9 <processname>
```