**Basic UNIX commands :**

The basic Unix commands are:

**echo**

It displays a line text.

Echo the given string or strings to standard output.

*Syntax:*  
echo [OPTION]... [STRING]...

Example:

echo "Good Morning"

This will display the following in the standard output

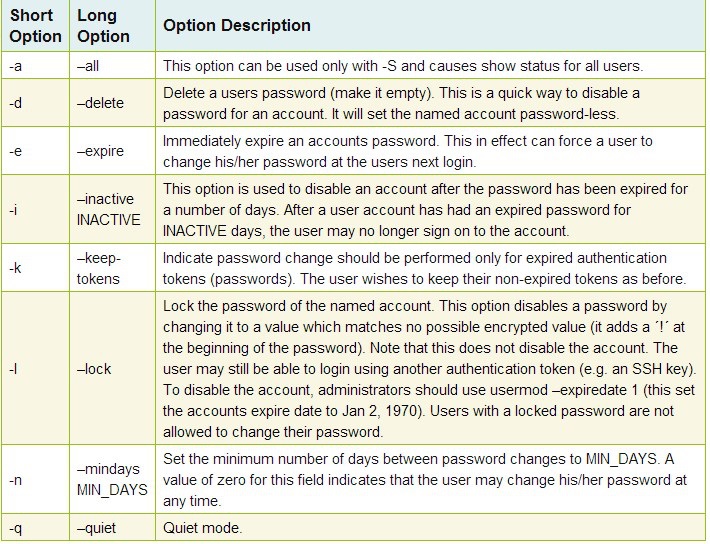
Good Morning

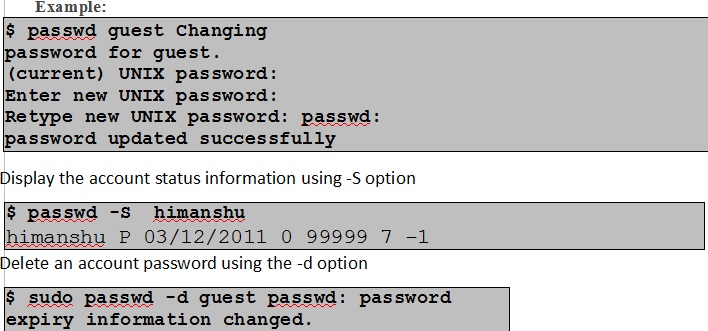
**passwd command**

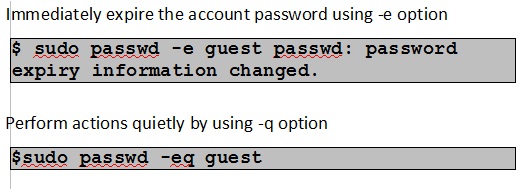
The passwd command is used to change passwords for user accounts. A normal user may only change the password for his or her own account, while the superuser may change the password for any account.

**Syntax :**passwd [options] [LOGIN]

**Options:**

****





**uname command**

Print certain system information.

**Syntax :**uname [OPTION]...

**Options:**



**Examples:**To show the name of the OS

**uname command:**

Used to get the  information about the current system (the name, version and other details)

$uname

To get the kernel name, you can use -s parameter.

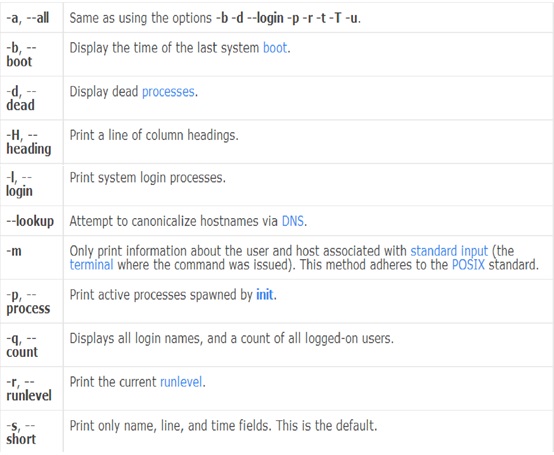
**$ uname -s**

To get the kernel release, -r option is used

**$ uname -r**

**who command:**

who command prints information about all users who are currently logged in.  
Syntax:  
who [OPTION]... [ FILE ] [ am i ]



**find command:**

 find is one of the powerful utility of Unix (or Linux). It can search  
the entire file-system to locate files and directories according to the specific  
search criteria. Other than searching files in can also perform actions(executing  
commands) on searched files.  
**Syntax:**

$ find [option] [action]

**Examples :**  
**1 .  File Type based Search:**  
>> The following command finds the file named Abc in current(.) directory and  
all its sub-directories

|  |
| --- |
| [390119@InGhyUnix ~]$ find . -name Abc ./Abc ./Dir1/Abc |

**2. Wild-Card based Search :** Wild-card characters like \* and ? can be used:  
>> To find all files where filename starts with Abc the command will be:

[390119@InGhyUnix ~]$ find . -name "Abc\*"  
./Abc  
./Dir1/Abc  
./Dir1/Abc.php  
./Abc.txt  
./Abc.dat

**3 . Source Location based Search:**  
>>The command find can search in multiple source location :

|  |
| --- |
| [390119@InGhyUnix ~]$ find Dir1 Dir2 -name "Abc.\*" Dir1/Abc.php Dir2/Abc.htm |

Searches for the file Abc with any extension only in directories Dir1 and Dir2

**4 . Size Based Search :** Using find files can be searched based on its size .  
>> To find the files with size more than 10Mb the command will be

|  |
| --- |
| [390119@InGhyUnix ~]$ find . -size +10M  ./Abc.dat |

**5 . Access/Modification-Time Based Search :** Using find files can be searched  
based on the time of last access/modification :  
>> To find the files which are accessed within last 3 days the command will be

|  |
| --- |
| [390119@InGhyUnix ~]$ find . -atime -3  ./Abc |

**6 . Permission Based Search :**Using find command files can be searched  
based on the access permission they have  
>> To find the files having read write and execute permission only for the owner  
the command will be :

|  |
| --- |
| [390119@InGhyUnix ~]$ find . -perm 700 ./Abc.txt |

**7 . Perform Action on the search result :** Using exec option any action (command)  
can be executed on the search result.  
>> To remove all the files having extension .swp from the current directory the  
command will be :

|  |
| --- |
| [390119@InGhyUnix ~]$ find . -name “\*.swp” -exec rm {} \; |

**bc command:**

bc is the command used for  basic mathematical calculations.

It can be used either a mathematical scripting language or as an interactive mathematical shell .

In interactive mode it accepts input from the terminal and providing calculations on requested arithmatic expression.

That means  typing the command bc on a Unix command prompt and entering a mathematical expression, such as (1 + 4) \* 2 , whereupon 10 will be the output

While used with files , bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is read.

**Syntax:**bc [ -hlwsqv ] [long-options] [ file ... ]

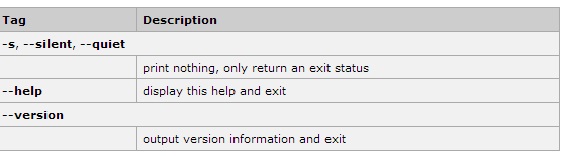
Example:

$ echo "2+5" | bc

Output will be:

7

**tty command:**  
Prints the file name of the terminal connected  in standard input.  
*Syntax :* tty [OPTION]...



**UNIX Text Editors**

To write a program or a paper even to edit a mail message, you need to create, write,store, retrieve, view, and edit files. In unix several text editors available for this .

Editors available unix are  **ed, node,NEdit, emacs and vi** .

One of the first end-user programs hosted on the system and standardized in UNIX-based systems ever since is the vi editor. ed is a line editor for the UNIX operating system.

**vi Editor:**

 vi (pronounced "vee-eye") is short for "vi"sual editor. It is a visual editor used to create and edit text files.

It displays a window into the file being edited that shows 24 lines of text.  vi lets you add, change and delete text, but does not provide  formatting capabilities like  centering lines or indenting paragraphs.   
  
vi works in three different modes:

1. Edit Mode -where any key is entered as text
2. Command Mode -where keys are used as commands
3. Ex Mode -ex commands can be entered in last line to act on text

Invoking vi with/without filename puts it in command mode:

vi []

**Commands to be used in vi:**

For cursor movement within the vi editor, we can not use the mouse . We have to use key commands. To use a key command, user has to be in the command mode. To go to command mode, press the Esc key.

**Commands to Enter and Exit vi:**

|  |  |
| --- | --- |
| command | Purpose of the command |
| vi filename | edit filename starting at line 1 |
| vi –r filename | recover filename that was being edited when system crashed |

The new or modified file is saved when user exits from vi. But it is also possible to exit without saving the file.

To use a command to exit from vi , after pressing the Esc key ,colon : is to be pressed.

The cursor moves to bottom of screen whenever a colon (:) is typed. This type of command is completed by hitting the  (or ) key.

|  |  |
| --- | --- |
| command | Purpose of the command |
| :wq | quit vi, writing out modified file to file named in original invocation |
| :q | quit (or exit) vi |
| :q! | quit vi even though latest changes have not been saved for this vi call |
| :w | write current contents to file named in original vi call |

**Commands to insert or add text:**

Following are the commands to be used to insert or add text.

|  |  |
| --- | --- |
| key | Purpose of the command |
| u | Undo the last command’s action |
| i | insert text before the cursor,until is hit |
| I | insert text at beginning of current line, until  hit |
| a | append text after cursor, until  hit |
| A | append text to end of current line, until  hit |
| o | open and put text in a new line below current line, until  hit |
| O | open and put text in a new line above current line, until  hit |

**Commands to modify texts:**

|  |  |
| --- | --- |
| **key** | **Purpose of the command** |
| **r** | replace single character under cursor (no  needed) |
| **R** | replace characters, starting with current cursor position, until  hit |
| **cw** | change the current word with new text,  starting with the character under cursor, until  hit |
| **cNw** | change N words beginning with character under cursor, until  hit;    e.g., c5w changes 5 words |
| **C** | change (replace) the characters in the current line, until  hit |
| **cc** | change (replace) the entire current line, stopping when  is hit |
| **Ncc or cNc** | change*(*replace) the next N lines, starting with the current line, stopping when  is hit |

**Commands to Delete texts:**

|  |  |
| --- | --- |
| **key** | **Purpose of the command** |
| **x** | delete single character under cursor |
| **Nx** | delete N characters, starting with character under cursor |
| **cw** | delete the single word beginning with character under cursor |
| **dw** | change N words beginning with character under cursor, until  hit;    e.g., c5w changes 5 words |
| **dNw** | delete N words beginning with character under cursor;    e.g., d5w deletes 5 words |
| **D** | delete the remainder of the line, starting with current cursor position |
| **dd** | delete entire current line |
| **Ndd or dNd** | delete N lines, beginning with the current line;    e.g., 5dd deletes 5 lines |

**Commands to copy and paste text:**

|  |  |
| --- | --- |
| **yy** | copy (yank, cut) the current line into the buffer |
| **Nyy** | copy (yank, cut) the next N lines, including the current line, into the buffer |
| **p** | put (paste) the line(s) in the buffer into the text after the current line |

**Commands for searching text**

|  |  |
| --- | --- |
| **/string** | search forward for occurrence of string in text |
| **?string** | search backward for occurrence of string in text |

**UNIX FILE COMMANDS**

In this module following important file handling commands will be discussed:

* touch
* cat
* cat > (single redirection)
* cat >>(double redirection)
* cp
* mv
* cmp
* comm
* diff

**touch command**  
Touch command is used in Unix to create an empty file.

|  |
| --- |
| $ touch file1 file2 $ ls -l -rw-r--r-- 1 user group 0 2014-02-08 7:14 file1 -rw-r--r-- 1 user group 0 2014-02-08 7:14 file2 $ |

Touch command is used to change the timestamps (access time and modification time of a file).

**cat command**  
Use of cat command:  
'cat' command is used to display a file content.  
**Syntax:**

$ cat filename [ filename ]

|  |
| --- |
| $ cat data This is an unix file This is line two $ |

**cat > and cat >> command**  
Redirection operator ( > and >>) can be used with cat command to take input from standard input device(keyboard) and store them in  a file.  
Syntax: $ cat > [filename]

|  |
| --- |
| $ cat > data The file created using cat > filename ^d $ |

cat with single redirection(cat >filename) , creates a new file if no file with the given file name already exists and overrites the file if a file with given file name already exists.

cat with double redirection (cat >>filename) appends the content given through standard input to the file.

**cp command**

cp command is used to copy one file's content to another file.  
Syntax: $ cp [source filename] [destination filename]  
Switches:  
1. cp -i [sourcefilename] [destination filename] This command copies the content of a file interactively.

2.cp -r [source\_directory] [destination\_directory] This command copies the whole file hierarchy.

3. cp -v [sourcefilename] [destination filename] Copies in verbose mode, explains what is being done

**mv command**  
mv command is used for:  
1. for renaming a file in same directory.  
2. for moving a file to a different directory.  
*Syntax:*  
$ mv [sourcefilename] [destinationfilename]  
$ mv [source\_directory] [destination\_directory]

**Directory Commands:**

Every item in the UNIX filesystem tree is either a file, or a directory. A directory is like a file folder. A directory can contain files, and other directories. A directory contained within another is called the child of the other. A directory in the filesystem tree may have many children, but it can only have one parent. A file can hold information, but cannot contain other files, or directories.

The place in the file system tree where an user is located is called the **current working directory**(which can be listed using **pwd** command).

As a user logs in, user is put in his/her**home directory.**

**mkdir(make directory) command:**  
The mkdir command creates a directory with specified name in the present working directory or specified path.  
Syntax: $ mkdir [ ]

**ls Command**  
 Lists the contents of any directory (current directory if no directory specified).  
 Sorts entries without option  
$ ls [option(s)] [filename]

**File comparison commands:**

**cmp** – This command is used to compare two files, and if they differ, tells the first byte and line number where they differ. If input files are same , returns nothing.

Syntax :

cmp options file1 file2

To use cmp commands both the files need to be sorted.

**comm-**This command is used to compare two sorted files.

Syntax:

comm [options] file1 file2

* One set of options allows selection of ‘columns’ to suppress.
  + -1: suppress lines unique to file1 (column 1)
  + -2: suppress lines unique to file2 (column 2)
  + -3: suppress lines common to file1 and file2 (column3)

**Example:** To show only lines common between file1 and file2, command is

$ comm -12 file1 file2

**diff**-  This command is used to compare two files line by line. The output indicates how the lines in each file are different, and the steps that will be involved in changing file1 to file2.

The change commands are in the format [range][acd][range].  The range on the left may be a line number or a comma-separated range of line numbers referring to file1, and the range on the right similarly refers to file2.  The character in the middle indicates the action i.e. add, change or delete.

* ‘LaR’ – Add lines in range ‘R’ from file2 after line ‘L’ in file1.
* ‘FcT’ – Change lines in range ‘F’ of file1 to lines in range ‘T’ of file2.
* ‘RdL’ – Delete lines in range ‘R’ from file1 that would have appeared at line ‘L’ in file2

**Syntax**: diff [options] file1 file2

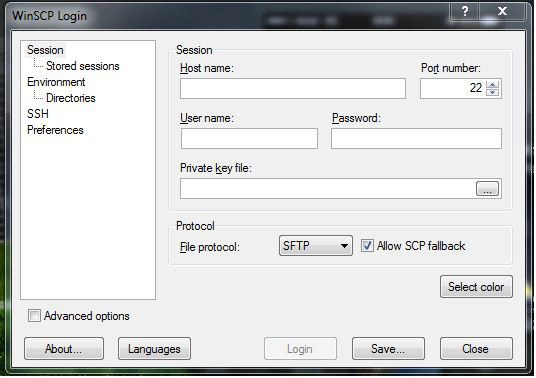
**USAGE OF WinSCP**

**Introduction to WinSCP**

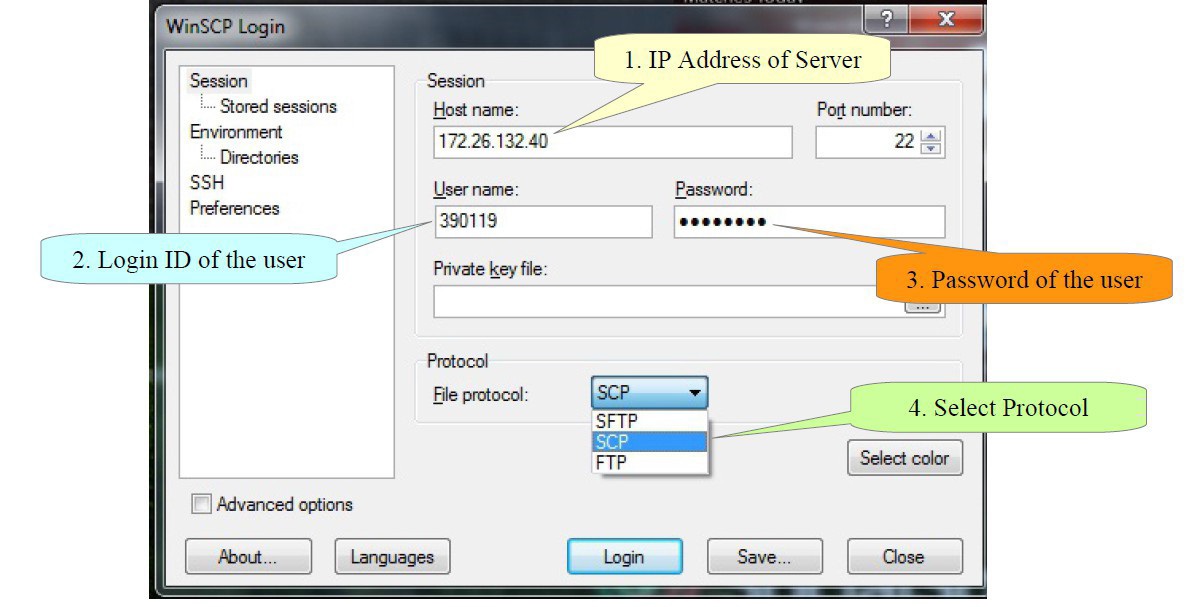
WinSCP (Windows Secure CoPy) is a popular , free and open-source Secure File Transfer application (SFTP) for Windows computers .It is used to transfer files securely between remote server and local computer and vice verse. It includes support for SCP file transfers, and has the ability to create secure tunnels via SSH. Secure FTP (SFTP) is similar to FTP, but with SFTP the entire session is encrypted, so that passwords are never sent in the clear text , and are therefore much less vulnerable to interception. That is ,for  
transferring files securely, we can use Secure copy or SCP protocol, it uses Secure Shell (SSH), which encrypts the content before transferring data over network. It runs over TCP port 22 by default.

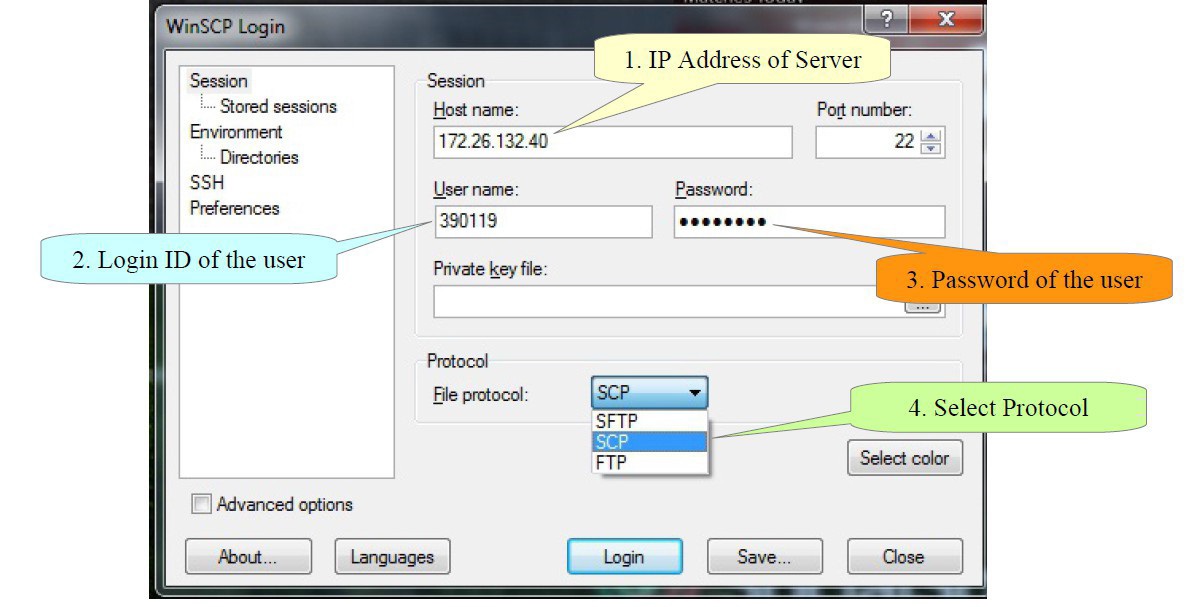
**How to use WinSCP**

Step I : **Start WinSCP**  
To start with double click on WinSCP icon in the desktop or select it from Programs Group . The  
first screen will look as below

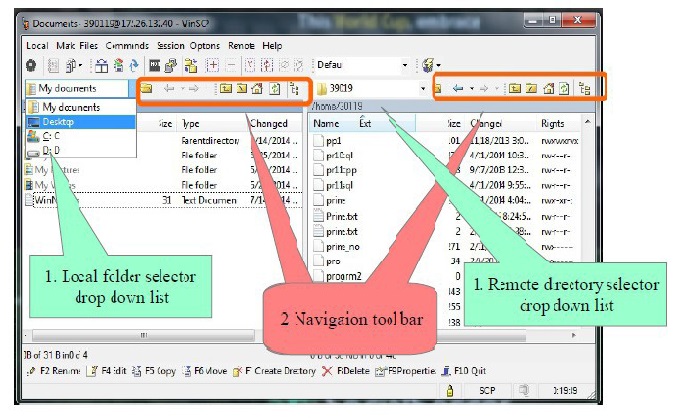


Step II : **Logging in to Server**  
Enter IP address of the server in Host Name field.  
Unix login userid in the User name field  
Unix login Password in Password field  
Choose the protocol ( SFTP/ SCP / FTP)  
For example  
1) Host name used : 172.26.132.40 (For example)  
2) User name used : 390119  
3) Password used : \*\*\*\*\*\*\*\*  
3) Protocol selected : SCP

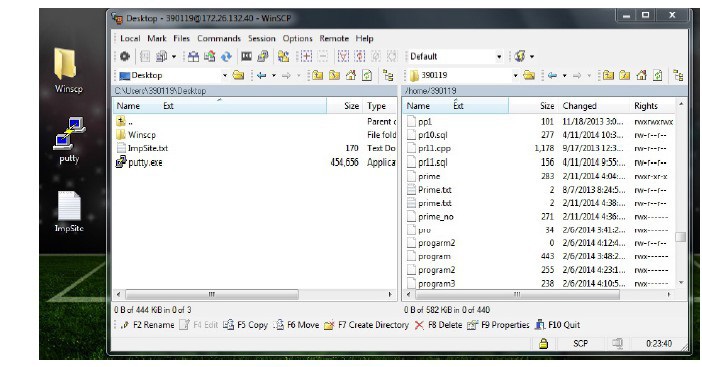




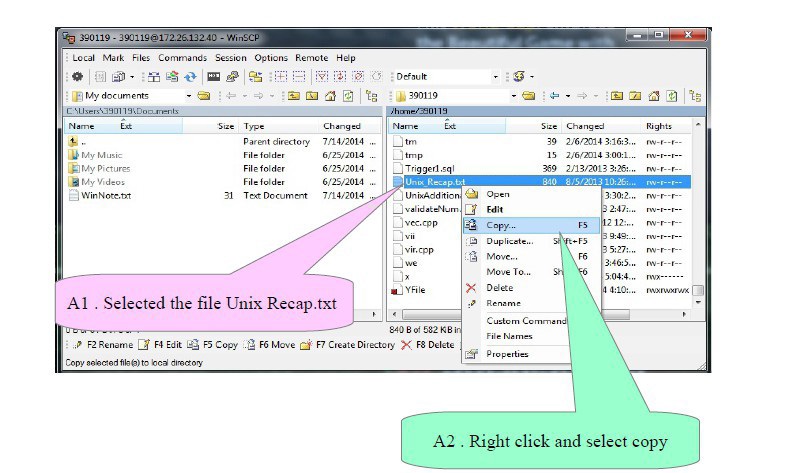
Step III : Select/Navigate to working directory  
1. We can select the working directory in both Local as well as remote machine from the drop  
down menu.  
2. Tool bar can be used to navigate to desired folder



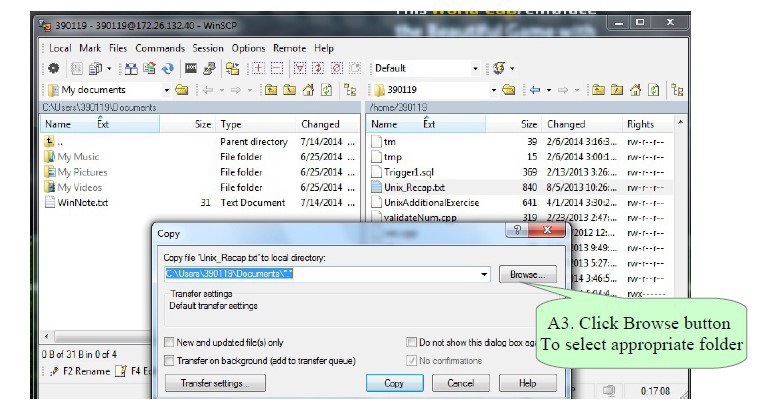
Step IV : Transferring content Content can be transferred in two way  
A . Downloading : Transferring content from Remote server to Local machine  
B . Uploading : Transferring content from Local machine to Remote server



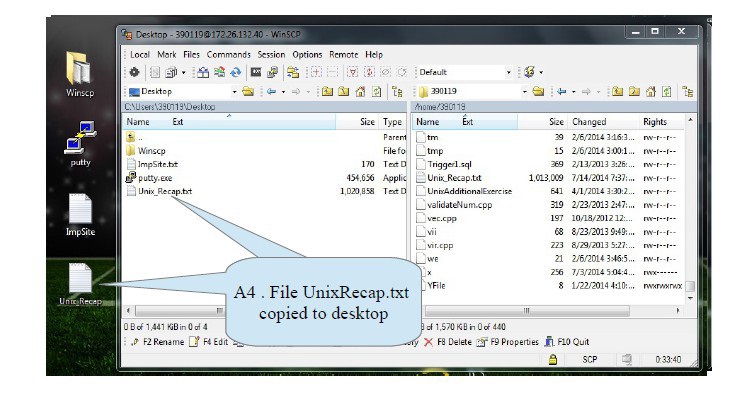
Following Steps to perform for Downloading file/Directory from Server  
Step - A1 : Select the file to transfer in the Remote pane,  
Step - A2 : Right click → Select Copy or move option



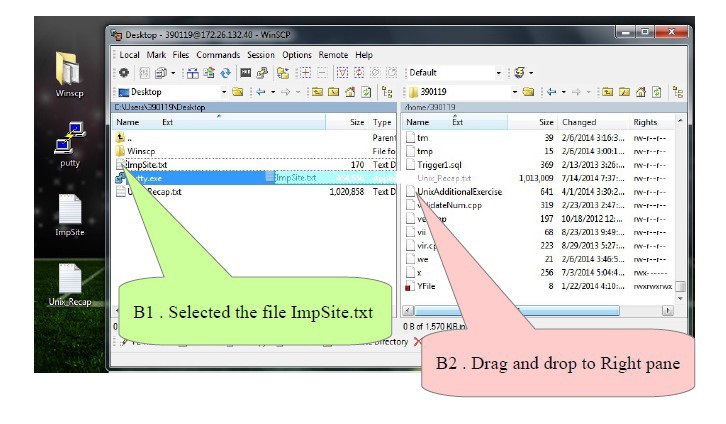
Step - A3 : Select the destination folder in the local machine(can be changed by selecting  
Browse button )



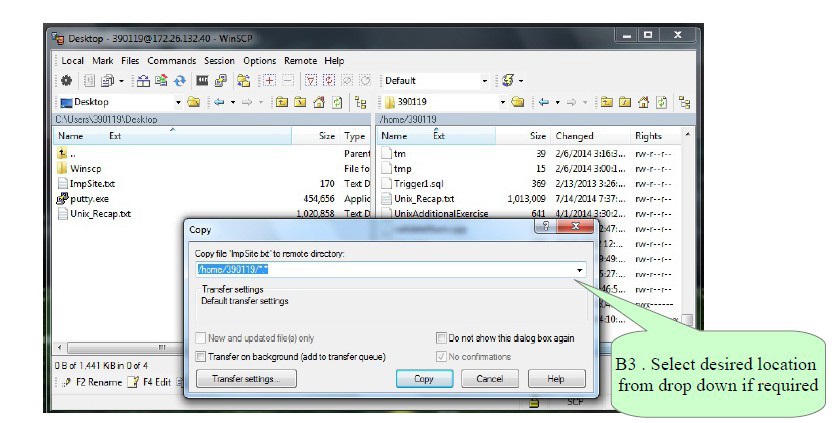
Step - A4 : The file Unix\_Recap.txt copied to desktop.



Uploading files to Remote Server  
Step - B1 : Select the file to copy in Local machine(Left) pane.  
Step - B2 : Drag it and drop it to the remote pane desired location.



Step - B3 : Select the desired location from drop down list and select copy



Saving sessions for future use  
All the session information can be saved for future use and faster access.  
The information can be saved are :  
>> Server IP  
>> User name,  
>> Password(Not recommended)  
>> Communication protocol

