S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT & RESEARCH, NAGPUR

Practical 02

Aim: To understand and demonstrate the use of basic commands in different operating systems (Windows, Linux, and UNIX) for managing files, directories, permissions, and user interactions through a terminal or command-line interface.

Name: Prasad Hore

USN: CM24034

Semester / Year: IV Sem / II Year

Academic Session: 2025-26

Date of Performance: 20-01-26

Date of Submission: 27-01-26

❖ **Aim:** To understand and demonstrate the use of basic commands in different operating systems (Windows, Linux, and UNIX) for managing files, directories, permissions, and user interactions through a terminal or command-line interface.

❖ Objectives:

1. To learn and practice fundamental command-line operations for file and directory management.
2. To explore and utilize user and permission management commands effectively.
3. To enhance system administration skills by working with commands across different operating systems.

❖ Requirements:

Hardware Requirements:

- **Processor**: Multi-core CPU, Intel Core i3 (3.0 GHz) or higher
- **RAM**: Minimum 4 GB (8 GB recommended for optimal performance)
- **Storage**: 100 GB HDD or SSD (Solid State Drive) for faster access
- **Network Interface**: Ethernet or Wi-Fi adapter for connectivity

Software Requirements:

- **Operating System**: Windows 10/11, Linux (Ubuntu 20.04/CentOS 8), UNIX-based OS
- **Command-line Interface**: PowerShell or Command Prompt (Windows), Terminal (Linux/UNIX)
- **Text Editor**: Nano, Vim, or Visual Studio Code for file editing
- **Administrative Privileges**: Superuser (Linux/UNIX) or Administrator (Windows) access

❖ Theory:

Command-line interfaces (CLI) are fundamental tools used in system administration to interact with operating systems efficiently. They enable users to execute tasks by entering text-based commands rather than relying on graphical interfaces. Using CLI commands, administrators can manage files and directories, modify permissions, monitor system performance, and configure system resources. Operating systems such as Windows, Linux, and UNIX offer numerous built-in commands to simplify administrative operations. Commands like `ls`, `cd`, `pwd`, `mkdir`, and `chmod` assist in navigating and organizing the file system. Understanding command structure and syntax ensures accurate execution of tasks. CLI usage helps in performing tasks faster and with greater control. It also supports automation through scripts, improving productivity and consistency. Gaining proficiency in basic commands enhances system management, security, and troubleshooting skills. This practical aims to build a strong foundation in using command-line tools effectively.

❖ **Commands:**

1. **Display User Manual of a Command**
   - Functionality: Shows the manual page with details about a command's usage, options, and arguments.
   - Syntax: man <command>
   - Example: man ls

2. **Change Current Working Directory**.
   - Functionality: Changes the terminal's current working directory.
   - Syntax: cd <directory-path>
   - Example: cd /home/user/Documents.

3. **List Contents of the Current Directory.**
   - Functionality: Lists all files and directories in the current location.
   - Syntax: ls
   - Example: ls

4. **Read/Modify/Concatenate Text Files.**
   - Functionality: Displays or manipulates file content.
   - Syntax:
     - Read: cat <filename>
     - Modify: 'nano <filename>
     - Concatenate: cat <file1> <file2> > <outputfile>

5. **Create a New Directory**.
   - Functionality: Creates a new directory at the specified path.
   - Syntax: mkdir <directory-name
   - Example: mkdir newdir

6. **Display Current Working Directory.**
   - Functionality: Prints the current directory path.
   - Syntax: pwd
   - Example: pwd

7. **Write Arguments to Standard Output.**
   - Functionality: Prints the provided string or variables.
   - Syntax: echo <arguments>
   - Example: echo Hello World

8. **Remove a File**.
   - Functionality: Deletes a specified file.
   - Syntax: rm <filename>
   - Example: rm file.txt

9. **Delete a Directory.**
   - Functionality: Removes an empty directory.
   - Syntax: rmdir <directory-name>
   - Example: rmdir olddir

10. **Copy a File or Directory.**
    - Functionality: Copies a file or directory to a destination.
    - Syntax: cp <source> <destination>
    - Example: cp file.txt backup/

11. **Switch to Root User.**
    - Functionality: Gains root privileges temporarily.
    - Syntax: sudo su
    - Example: sudo s

12. **Move Files or Directories.**
    - Functionality: Moves or renames files and directories.
    -Syntax: mv <source> <destination>
    - Example: mv file.txt newdir/

13. **Search for a String in a File.**
    - Functionality: Searches for a specific word or pattern in a file.
    - Syntax: grep "<string>" <file>
    - Example: grep "error" log.txt

14. **Print Top N Lines of a File.**
    - Functionality: Displays the first N lines of a file.
    - Syntax: head -n <N> <file>
    - Example: ‘head -n 10 file.txt’

15. **Print Last N Lines of a File.**
    - Functionality: Displays the last N lines of a file.
    - Syntax: tail -n <N> <file>
    - Example: ‘tail -n 10 file.txt

16. **Remove Read Permission from Owner.**
   - Functionality: Revokes the owner's read permission for a file.
   - Syntax: chmod u-r <filename>
   - Example: chmod u-r file.txt

17. **Change Specific Permissions.**
   - Functionality: Sets or removes specific file permissions.
   -Syntax: chmod u+r,w-x,g+w <filename>
   - Example: chmod u+r,w-x,g+w file.txt

18. **Add Write Permission to Owner, None to Others**.
   -Functionality: Allows write access for the owner only.
   - Syntax: chmod u+w,o-rwx <filename>
   - Example: chmod u+w,o-rwx file.txt

19. **Assign Permissions to Users.**
   - Functionality: Modifies file access for users, groups, and others.
   - Syntax: chmod u+rwx,g+rx,o+r <filename>
   - Example: 'chmod u+rwx,g+rx,o+r file.txt

20. **Assign R/W/X to Others.**
   - Functionality: Gives read, write, and execute permissions to others.
   - Syntax: chmod o+rwx <filename>
   - Example: chmod o+rwx file.txt

21. **Remove All Permissions from All Users.**
   - Functionality: Clears all permissions on a file.
   - Syntax: 'chmod a-rwx <filename>
   - Example: 'chmod a-rwx file.txt

22. **Remove Read Permission Using Absolute Mode.**
   - Functionality: Uses numeric mode to restrict read access.
   - Syntax: chmod 700 <filename>
   - Example: chmod 700 file.txt

23. **Set R/W for Owner, None for Group/Other.**
   - Functionality: Assigns permissions in numeric mode.
   - Syntax: chmod 600 <filename>
   - Example: chmod 600 file.txt'

24. **Add Execute for Owner, Read for Group/Others.**
   - Functionality: Adds execution and read access.
   - Syntax: chmod u+x,g+r,o+r <filename>

- Example: chmod u+x,g+r,o+r file.txt

25. **Add Execute Permission to All Users.**
    - Functionality: Enables execution by everyone.
    - Syntax: chmod a+x <filename>
    - Example: chmod a+x script.sh

❖ **Conclusion:** In conclusion, understanding and using essential operating system commands like **'ls', 'cd', 'cp', 'mv', and 'chmod'** enables efficient file management, navigation, and permission control. Tools like **'grep', 'head', and 'tail'** enhance data processing. Mastery of these commands improves system administration, task automation, and overall system security and performance.

❖ **Discussion Questions:**

1. **What is the significance of the pwd command in a Linux environment?**
2. **-Explain the function of the cp command and its common options.**
3. **How does chmod 700 affect file permissions, and what does each digit represent?**
4. **Describe the difference between head and tail commands in Linux.**
5. **What is the purpose of the grep command, and how is it used with regular expressions?**

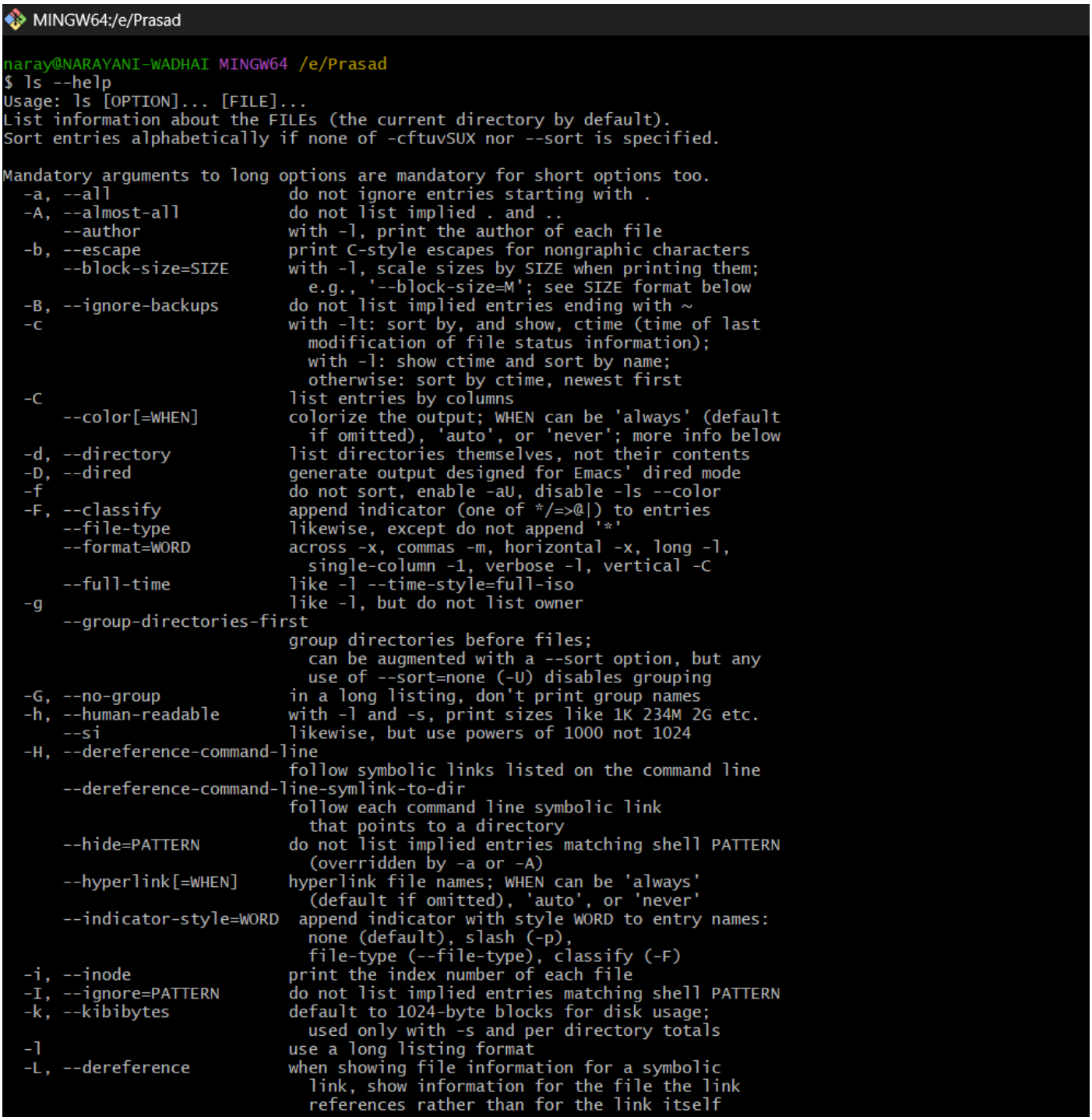
❖ **References**:

https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
https://www.geeksforgeeks.org/25-basic-ubuntu-commands/

Date: 27/01/2026

Signature
Course Coordinator
B.Tech CSE(AIML)
Sem: 4 / 2025-26

- help


MINGW64:/e/Prasad
naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
Mandatory arguments to long options are mandatory for short options too.
-a, --all do not ignore entries starting with .
-A, --almost-all do not list implied . and ..
--author with -l, print the author of each file
-b, --escape print C-style escapes for nongraphic characters
--block-size=SIZE with -l, scale sizes by SIZE when printing them;
e.g., '--block-size=M'; see SIZE format below
-B, --ignore-backups do not list implied entries ending with ~
-c with -lt: sort by, and show, ctime (time of last
modification of file status information);
with -l: show ctime and sort by name;
otherwise: sort by ctime, newest first
-C list entries by columns
--color[=WHEN] colorize the output; WHEN can be 'always' (default
if omitted), 'auto', or 'never'; more info below
-d, --directory list directories themselves, not their contents
-D, --dired generate output designed for Emacs' dired mode
-f do not sort, enable -aU, disable -ls --color
-F, --classify append indicator (one of */=>@|) to entries
--file-type likewise, except do not append '*'
--format=WORD across -x, commas -m, horizontal -x, long -l,
single-column -1, verbose -l, vertical -C
--full-time like -l --time-style=full-iso
-g like -l, but do not list owner
--group-directories-first
group directories before files;
can be augmented with a --sort option, but any
use of --sort=none (-U) disables grouping
-G, --no-group in a long listing, don't print group names
-h, --human-readable with -l and -s, print sizes like 1K 234M 2G etc.
--si likewise, but use powers of 1000 not 1024
-H, --dereference-command-line
follow symbolic links listed on the command line
--dereference-command-line-symlink-to-dir
follow each command line symbolic link
that points to a directory
--hide=PATTERN do not list implied entries matching shell PATTERN
(overridden by -a or -A)
--hyperlink[=WHEN] hyperlink file names; WHEN can be 'always'
(default if omitted), 'auto', or 'never'
--indicator-style=WORD append indicator with style WORD to entry names:
none (default), slash (-p),
file-type (--file-type), classify (-F)
-i, --inode print the index number of each file
-I, --ignore=PATTERN do not list implied entries matching shell PATTERN
-k, --kibibytes default to 1024-byte blocks for disk usage;
used only with -s and per directory totals
-l use a long listing format
-L, --dereference when showing file information for a symbolic
link, show information for the file the link
references rather than for the link itself

```
  -m                         fill width with a comma separated list of entries
  -n, --numeric-uid-gid      like -l, but list numeric user and group IDs
  -N, --literal              print entry names without quoting
  -o                         like -l, but do not list group information
  -p, --indicator-style=slash
                             append / indicator to directories
  -q, --hide-control-chars   print ? instead of nongraphic characters
      --show-control-chars   show nongraphic characters as-is (the default,
                               unless program is 'ls' and output is a terminal)
  -Q, --quote-name           enclose entry names in double quotes
      --quoting-style=WORD   use quoting style WORD for entry names:
                               literal, locale, shell, shell-always,
                               shell-escape, shell-escape-always, c, escape
                               (overrides QUOTING_STYLE environment variable)
  -r, --reverse              reverse order while sorting
  -R, --recursive            list subdirectories recursively
  -s, --size                 print the allocated size of each file, in blocks
  -S                         sort by file size, largest first
      --sort=WORD            sort by WORD instead of name: none (-U), size (-S),
                               time (-t), version (-v), extension (-X)
      --time=WORD            change the default of using modification times;
                               access time (-u): atime, access, use;
                               change time (-c): ctime, status;
                               birth time: birth, creation;
                             with -l, WORD determines which time to show;
                             with --sort=time, sort by WORD (newest first)
      --time-style=TIME_STYLE  time/date format with -l; see TIME_STYLE below
  -t                         sort by time, newest first; see --time
  -T, --tabsize=COLS         assume tab stops at each COLS instead of 8
  -u                         with -lt: sort by, and show, access time;
                               with -l: show access time and sort by name;
                               otherwise: sort by access time, newest first
  -U                         do not sort; list entries in directory order
  -v                         natural sort of (version) numbers within text
  -w, --width=COLS           set output width to COLS.  0 means no limit
  -x                         list entries by lines instead of by columns
  -X                         sort alphabetically by entry extension
  -Z, --context              print any security context of each file
  -1                         list one file per line.  Avoid '\n' with -q or -b
      --append-exe           append .exe if cygwin magic was needed
      --help     display this help and exit
      --version  output version information and exit

The SIZE argument is an integer and optional unit (example: 10K is 10*1024).
Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of 1000).
Binary prefixes can be used, too: KiB=K, MiB=M, and so on.

The TIME_STYLE argument can be full-iso, long-iso, iso, locale, or +FORMAT.
FORMAT is interpreted like in date(1).  If FORMAT is FORMAT1<newline>FORMAT2,
then FORMAT1 applies to non-recent files and FORMAT2 to recent files.
TIME_STYLE prefixed with 'posix-' takes effect only outside the POSIX locale.
Also the TIME_STYLE environment variable sets the default style to use.

Using color to distinguish file types is disabled both by default and
with --color=never.  With --color=auto, ls emits color codes only when
standard output is connected to a terminal.  The LS_COLORS environment
variable can change the settings.  Use the dircolors command to set it.
```

```
Exit status:
 0  if OK,
 1  if minor problems (e.g., cannot access subdirectory),
 2  if serious trouble (e.g., cannot access command-line argument).

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Report any translation bugs to <https://translationproject.org/team/>
Full documentation <https://www.gnu.org/software/coreutils/ls>
or available locally via: info '(coreutils) ls invocation'

naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$
```

- cd

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ cd "/e/N"

naray@NARAYANI-WADHAI MINGW64 /e/N
$
```

- mkdir

```
MINGW64:/e/Prasad/Hore

naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ mkdir Hore

naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ cd Hore

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$
```

- ls

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ ls

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ ls -l
total 0

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$
```

- nano,cat,echo

```
MINGW64:/e/Prasad/Hore
  GNU nano 8.7                                                Prasad.txt
Hi I'm prasad hore of cse ai ml
```

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ nano Prasad.txt

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ cat Prasad.txt
Hi I'm prasad hore of cse ai ml

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ echo "Hello,Os" > Prasad2.txt

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ cat Prasad.txt Prasad2.txt > combi.txt

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ cat combi.txt
Hi I'm prasad hore of cse ai ml
Hello,Os

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$
```

- mkdir

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ mkdir OSCM24034

naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ ls OSCM24034
Prasad.txt  Prasad2.txt  combi.txt
```

- pwd

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ pwd
/e/Prasad
```

- echo

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ echo Hello!
Hello!

naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ echo Os practical 2
Os practical 2
```

- rm

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ rm Prasad2.txt

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/Hore
$ ls
Prasad.txt  combi.txt
```

- rmdir

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ ls
ADAUS/  Hore/  OSCM24034/

naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ rmdir Hore

naray@NARAYANI-WADHAI MINGW64 /e/Prasad
$ ls
ADAUS/  OSCM24034/
```

- cp

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ cp combi.txt bk.txt

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ ls
Prasad.txt  Prasad2.txt  bk.txt  combi.txt
```

- mv

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ mv bk.txt renamed_file.txt

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ ls
Prasad.txt  Prasad2.txt  combi.txt  renamed_file.txt
```

- grep

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ grep "first" Prasad.txt
```

- touch

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ touch trial.txt

naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ ls
Prasad.txt  Prasad2.txt  combi.txt  renamed_file.txt  trial.txt
```

- uname

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ uname -a
MINGW64_NT-10.0-26200 NARAYANI-WADHAI 3.6.6-1cdd4371.x86_64 2026-01-15 22:20 UTC x86_64 Msys
```

- date

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ date
Tue Feb  3 15:58:56 IST 2026
```

- whoami

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ whoami
naray
```

- ps

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ ps
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
      976       1     976      13628  ?        197609 15:16:12 /usr/bin/mintty
      977     976     977       5316  pty0     197609 15:16:12 /usr/bin/bash
     1153     977    1153       7056  pty0     197609 16:02:22 /usr/bin/ps
```

- history

```
naray@NARAYANI-WADHAI MINGW64 /e/Prasad/OSCM24034
$ history
    1  ls --help
    2  cd "/e/N"
    3  cd "e/Prasad"
    4  mkdir Hore
    5  cd Hore
    6  ls
    7  ls -l
    8  nano Prasad.txt
    9  cat Prasad.txt
   10  nano Prasad.txt
   11  cat Prasad.txt
   12  echo "Hello,Os" > Prasad2.txt
   13  cat Prasad.txt Prasad2.txt > combi.txt
   14  cat combi.txt
   15  nano Prasad.txt
   16  cd Prasad
   17  cd "/e/Prasad"
   18  mkdir OSCM24034
   19  ls OSCM24034
   20  pwd
   21  echo Hello!
   22  echo Os practical 2
   23  cd "/e/Prasad/Hore"
   24  rm Prasad2.txt
   25  ls
   26  cd "/e/Prasad"
   27  ls
   28  rmdir Hore
   29  ls
   30  rmdir Hore
   31  ls
   32  cp combi.txt bk.txt
   33  cd "/e/Prasad/OSCM24034"
   34  cp combi.txt bk.txt
   35  ls
   36  mv bk.txt
   37  mv bk.txt renamed_file.txt
   38  ls
   39  grep "first" Prasad.txt
   40  touch trial.txt
   41  ls
   42  uname -a
   43  date
   44  whoami
   45  ps
   46  history
```