**PWD**

**Description**

**pwd** prints the full pathname of the current working [directory](https://www.computerhope.com/jargon/d/director.htm).

**pwd syntax**

**pwd** [*OPTION*]...

**Options**

|  |  |
| --- | --- |
| **-L**, **--logical** | If the **PWD** environment variable contains an absolute name of the current directory with no "**.**" or "**..**" components, then output those contents, even if they contain symbolic links Otherwise, fall back to default (**-P**) behavior. |
| **-P**, **--physical** | Print a fully resolved name for the current directory, in which all components of the name are actual directory names, and not symbolic links. |
| **--help** | Display a help message, and exit. |
| **--version** | Display version information, and exit. |

**Screenshot**



Vi

**Description**

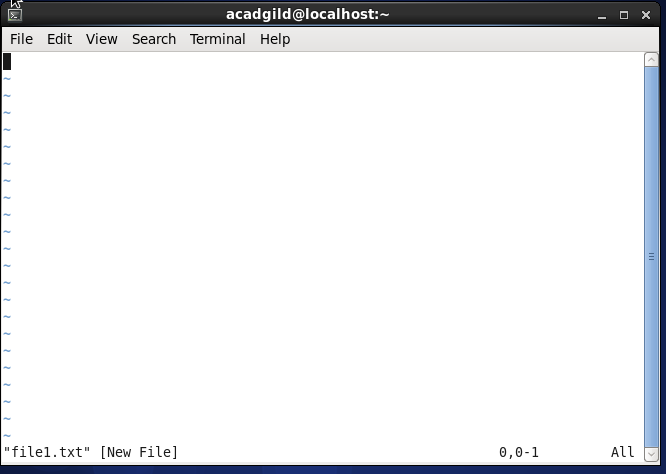
Vi is a command line text editor. Vi is intended as a plain text editor (similar to Notepad on Windows, or Textedit on Mac) as opposed to a word processing suite such as Word or Pages. It does, however have a lot more power compared to Notepad or Textedit.

**vi syntax**

vi filename

**To exit vi and save changes:** ZZ or :wq  
**To exit vi without saving changes:** :q!  
**To enter vi command mode:** *[esc]*

**Screenshot**



**Touch**

**Description**

The touch [command](http://www.linfo.org/command.html) is the easiest way to create new, empty [files](http://www.linfo.org/file.html). It is also used to change the timestamps (i.e., dates and times of the most recent access and modification) on existing files and [directories](http://www.linfo.org/directory.html).

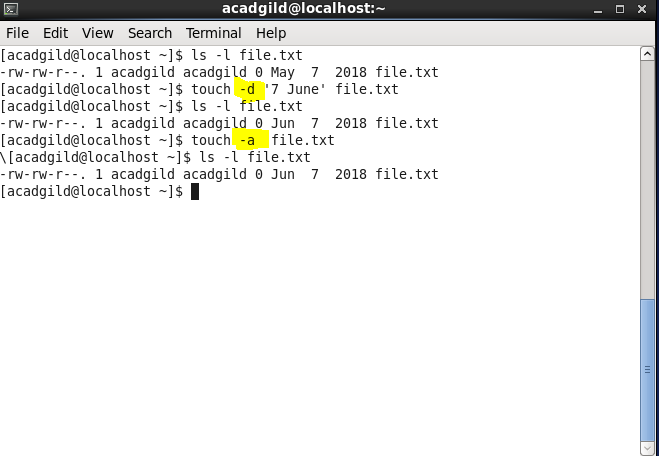
**Touch syntax**

touch [option] file\_name(s)

**Options**

|  |  |
| --- | --- |
| **am** | the *-a* option changes only the access time, while the *-m* option changes only the modification time. The use of both of these options together changes both the access and modification times to the current time, |
| **-r** | The *-r* (i.e., *reference*) option followed directly by a space and then by a file name tells touch to use that file's time stamps instead of current time.. |
| **--B** | The *-B* option modifies the timestamps by going back the specified number of seconds. |
| **-F** | *-F* option modifies the time by going forward the specified number of seconds. |
| **-d -t** | The *-d* and *-t* options allow the user to add a specific last access time. |

**Screenshot**

****

**mkdir**

**Description**

The mkdir command is to create new directories.

**mkdir syntax**

mkdir [option] directory\_name(s)

**Options**

|  |  |
| --- | --- |
| **-m** | mkdir's *-m* option is used to control the permissions of new directories. New directories are by default created with the read, write and *execute* (i.e., run as a program if a program) permissions enabled for the *owner* (i.e., the creator of the directory by default)  mkdir -m 777 dir\_4  The first digit represents the owner, the second represents the group and the third represents other users. The number 7 represents all three types of permission (i.e., read, write and execute), 6 stands for read and write only, 5 stands for read and execute, 4 is read only, 3 is write and execute, 2 is write only, 1 is execute only and 0 is no permissions. |
| **-p** | The *-p* (i.e., *parents*) option creates the specified intermediate directories for a new directory if they do not already exist. |

**Screenshot**

****

**rm**

**Description**

he *rm* (i.e., *remove*) command is used to delete files and directories on Linuxand other Unix-like operating systems.

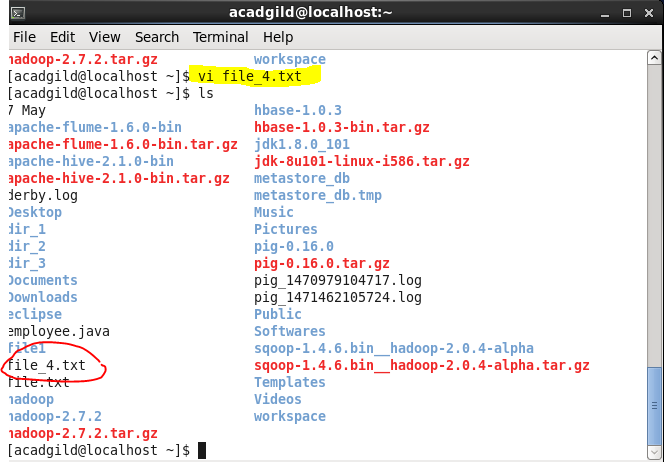
**rm syntax**

rm [options] [-r directories] filenames

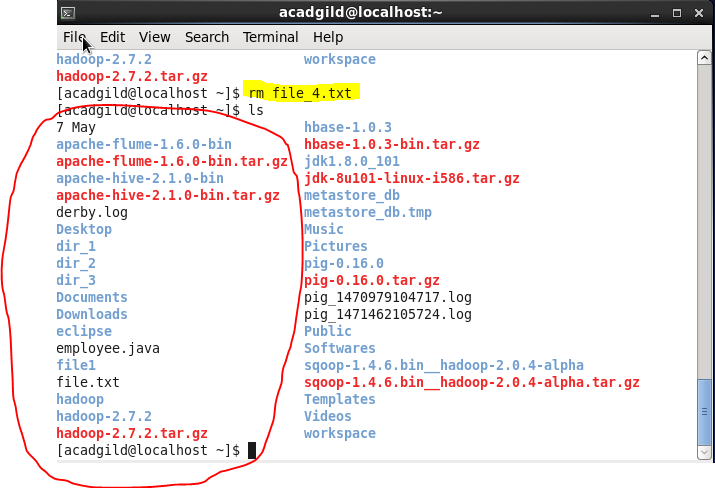
**Options**

|  |  |
| --- | --- |
| **-f** | The *-f* (i.e., *force*) [option](http://www.linfo.org/option.html) tells rm to remove all specified files, whether write-protected or not, without prompting the user. It does not display an error message or return error status if a specified file does not exist. However, if an attempt is made to remove files in a write-protected directory, this option will not suppress an error message.. |
| **-i** | The *-i* (i.e., *interactive*) option tells rm to prompt the user for confirmation before removing each file and directory. If both the -f and -i options are specified, the last one specified takes affect. |

**Screenshot**

****

**File\_4.txt does not exist after executing rm**

****

**ls**

**Description**

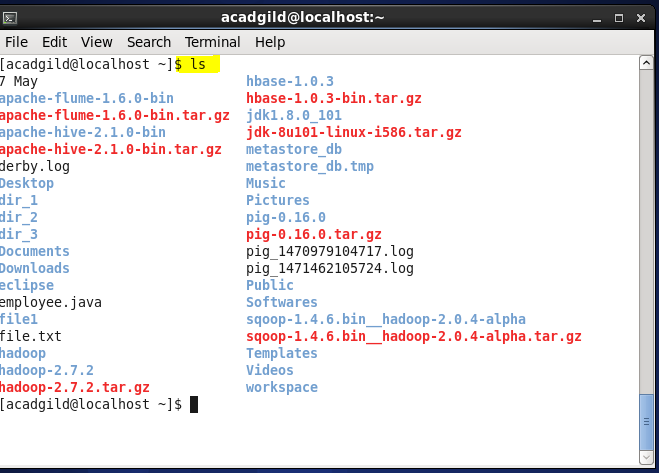
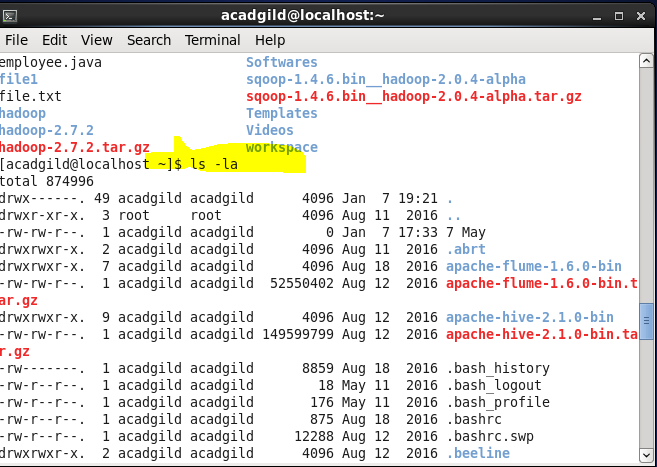
*ls* is a Linux shell command that lists directory contents of files and directories.

**ls syntax**

rm [options] [-r directories] filenames

**Options**

|  |  |  |
| --- | --- | --- |
| [**ls -a**](https://www.rapidtables.com/code/linux/ls/ls-a.html) | list all files including hidden file starting with '.' | |
| **ls --color** | colored list [=always/never/auto] | |
| **ls -d** | list directories - with ' \*/' | |
| **ls -F** | add one char of \*/=>@| to enteries | |
| **ls -i** | list file's inode index number | |
| [**ls -l**](https://www.rapidtables.com/code/linux/ls/ls-l.html) | list with long format - show permissions | |
| [**ls -la**](https://www.rapidtables.com/code/linux/ls/ls-l.html) | list long format including hidden files | |
| [**ls -lh**](https://www.rapidtables.com/code/linux/ls/ls-l.html) | list long format with readable file size | |
| [**ls -ls**](https://www.rapidtables.com/code/linux/ls/ls-l.html) | list with long format with file size | |
| [**ls -r**](https://www.rapidtables.com/code/linux/ls/ls-r.html#reverse) | list in reverse order | |
| [**ls -R**](https://www.rapidtables.com/code/linux/ls/ls-r.html#recursive) | list recursively directory tree | |
| [**ls -s**](https://www.rapidtables.com/code/linux/ls/ls-s.html#size) | list file size | |
| [**ls -S**](https://www.rapidtables.com/code/linux/ls/ls-s.html#sort-size) | sort by file size | |
| [**ls -t**](https://www.rapidtables.com/code/linux/ls/ls-t.html) | sort by time & date | |
| **ls -X** | sort by extension name | |
|  | |  | |
|  | |  | |

**Screenshot **

**echo**

**Description**

echo is a built-in command in the bash and C shells that writes its arguments to standard output.

A shell is a program that provides the command line (i.e., the all-text display user interface) on Linux and other Unix-like operating systems It also executes (i.e., runs) commands that are typed into it and displays the results. bash is the default shell on Linux.

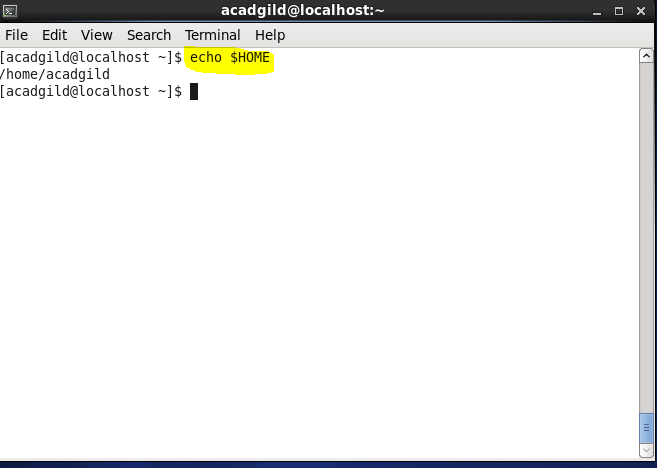
**echo syntax**

echo [option(s)] [string(s)]

**Options**

|  |  |
| --- | --- |
| **$HOME** | the environmental value that shows the current user's [home directory](http://www.linfo.org/home_directory.html), |
| **$PATH** | environmental variable, which contains a colon-separated list of the directories that the system searches to find the executable program corresponding to a command issued by the user:. |
| **-e** | The *-e* option is used to enable echo's interpretation of additional instances of the newline character as well as the interpretation of other special characters, such as a horizontal tab, which is represented by *\t*. |
|  |  |

**Screenshot**

****

**cat**

**Description**

*cat* is one of the most frequently used commands on Unix-like operating systems It has three related functions with regard to text files: displaying them, combining copies of them and creating new ones.

**cat syntax**

cat [options] [filenames] [-] [filenames]

**Options**

cat file1 > file2

the standard output of cat is redirected using the *output redirection operator* (which is represented by a rightward pointing angular bracket) to *file2*:

cat > felines  
This is not about a feline.

Typing the command *cat* followed by the output redirection operator and a file name on the same line, pressing ENTER to move to the next line, then typing some text and finally pressing ENTER again

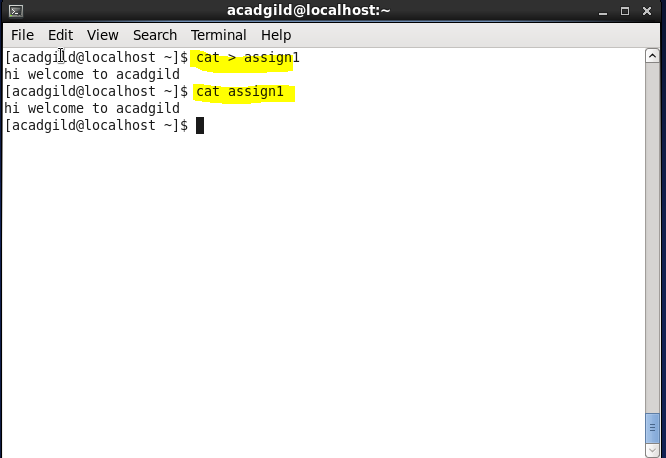
cat > file1

The third use for cat is file creation. For small files this is often easier than using [*vi*](http://www.linfo.org/vi/index.html), *gedit* or other text editors. It is accomplished by typing *cat* followed by the output redirection operator and the name of the file to be created, then pressing ENTER and finally simultaneously pressing the CONTROL and *d* keys.

cat >> file1

If a file named *file1* already exists, it will be *overwritten* (i.e., all of its contents will be erased) by the new, empty file with the same name. Thus the cautious user might prefer to instead use the *append operator* (represented by two successive rightward pointing angular brackets) in order to prevent unintended erasure

**Screenshot**

****

**who**

**Description**

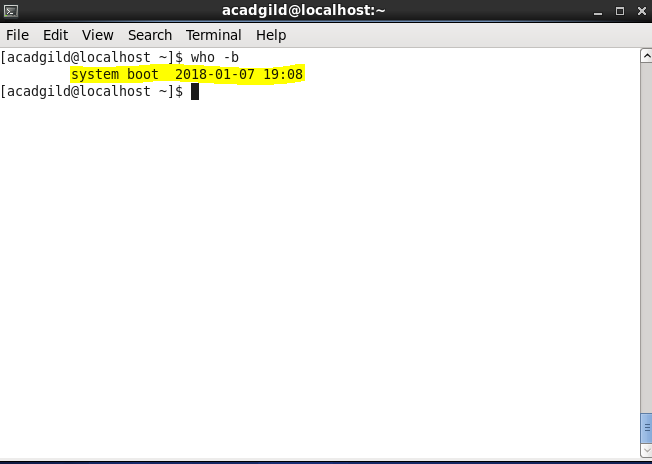
Displays who is loggedinto system

**who syntax**

who [OPTION]... [ FILE | ARG1 ARG2 ]

**Options**

|  |  |
| --- | --- |
| **-b** | Get the time of last system boot |
| **-l** | Get information on system login processes |
| **-m** | Get the hostname and user associated with stdin |
| **-u** | Get the list of user logged in |
| **-r** | Get the current run level |
| **-q** | Get number of users logged-in and their user names |
| **-a** | Get all the information |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | **Screenshot** |

****