**Experiment 5**

**Types of Editors and Shells in Linux**

**Aim:**To study various editors and shells in linux

**# Editors in UNIX**

Many files in Linux are in plain text format such as:

1. Configuration files.
2. Shell programs.
3. Manual pages (although sometimes stored in compressed format).

So we need a utility which to let us change text files. These Utilities are known as Text Editors. Some popular text editors are:

* emacs, xemacs
* vi, vim
* gedit, pico, nano

Linux editors for plain text can be divided into two categories, graphical GUI editors and console text editors. The advantage of the GUI editor is intuitive user friendly interface while the benefit of the console text editor is the suitability over long distance network connections which may or may not provide suitable bandwidth or reliability which would both be required by the GUI editors for remote operation.

Some popular editors are:

1. **VI Editor:** The vi editor (short for visual editor) is a screen editor which is available on almost all Unix systems. Once you have learned vi, you will find that it is a fast and powerful editor. vi has no menus but instead uses combinations of keystrokes in order to accomplish commands.
   1. Features of VI Editor
      1. It is present in almost every Linux Unix system, even the most minimal.
      2. It is very small. In fact, some versions have a total code size of less than 100KB. This makes it easy to include vi on even the tiniest versions of Linux, such as those in embedded systems and those that run from a single floppy disk.
      3. It is typist-friendly, at least once you get used to it. For example, the commands are very short, usually just a few keystrokes. And because vi does not use the mouse, there is never any need to remove one's hands from the keyboard. This can speed up editing substantially.
      4. It is very powerful, as just a few very short commands can make sweeping changes to large documents. In fact, vi is more powerful than most of its users realize, and few of them know more than just fraction of all the commands.
   2. VI has a total of approximately 150 basic commands. Some of them being,
      1. Command usage: vim [arguments] filename1 [filename2 ...]
      2. Use command: ZZ

Save changes to current file and quit.

* + 1. Use command line: ":wq"

Save (write) changes to current file and quit.

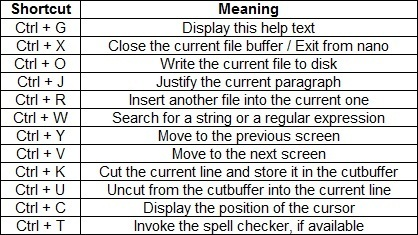
* + 1. Use command line: ":w"

Save (write) changes to current file without quitting.

* + 1. Use command line: ":w!"

Save (write) changes to current file overriding the file permissions if the user has the privileges to change the file permissions. For example this will save a file with read only privileges if the user is the owner or has the ability to modify the privileges to allow a write. This will not permanently modify the file privileges. Note that there is no space between the two characters. A space will infer that the output is streamed to a Unix command following the "!".

1. **GEDIT:** gedit is the default text editor of the GNOME desktop environment and part of the GNOME Core Applications. Designed as a general-purpose text editor, gedit emphasizes simplicity and ease of use, with a clean and simple GUI, according to the philosophy of the GNOME project. It includes tools for editing source code and structured text such as markup languages. It is free and open-source software subject to the requirements of the GNU General Public License version 2 or later.gedit is also available for Mac OS X and Microsoft Windows.
   1. Notable Features of GEDIT
      1. Full support for internationalized text (UTF-8)
      2. Configurable syntax highlighting for various languages (C, C++, Java, HTML, XML, Python, Perl and many others)
      3. Undo/Redo
      4. Editing files from remote locations
      5. File reverting
      6. Print and print preview support
      7. Clipboard support (cut/copy/paste)
      8. Search and replace with support of regular expressions
      9. Go to specific line
      10. Auto indentation
      11. Text wrapping
2. **NANO Editor:** Nano is a modeless editor so you can start typing immediately to insert text. If you are editing a configuration file like /etc/fstab use the -w switch to disable wrapping on long lines as it might render the configuration file unparseable by whatever tools depend on it.
   1. GNU nano, like Pico, is [keyboard](https://en.wikipedia.org/wiki/Computer_keyboard)-oriented, controlled with [control keys](https://en.wikipedia.org/wiki/Control_key). For example, Ctrl+O saves the current file; Ctrl+W goes to the search menu. GNU nano puts a two-line "shortcut bar" at the bottom of the screen, listing many of the commands available in the current context. For a complete list, Ctrl+G gets the help screen.
   2. Unlike Pico, nano uses [meta keys](https://en.wikipedia.org/wiki/Meta_key) to toggle its behavior. For example, Meta+S toggles smooth scrolling mode on and off. Almost all features that can be selected from the [command line](https://en.wikipedia.org/wiki/Command_line) can be dynamically toggled. On keyboards without the meta key it is often mapped to the escape key, Esc, such that in order to simulate, say, Meta+S one has to press the Esc key, then release it, and then press the S key.
   3. GNU nano can also use pointer devices, such as a mouse, to activate functions that are on the shortcut bar, as well as position the cursor.
   4. Shortcuts for Nano Editors are as shown



**# Shells in UNIX**

In addition to graphical user interfaces like Gnome, the Linux operating system also offers several shells. These command-line interfaces provide powerful environments for software development and system maintenance. Though shells have many commands in common, each type has unique features.A shell is what’s known as an interpreter. Similarly to a compiler, an interpreter translates the human code into machine code. One of the differences is that an interpreter does this one statement at a time as opposed to a compiler which scans the entire program and translates it as a whole into machine code.

The following are some of the important shells in UNIX:

|  |  |  |  |
| --- | --- | --- | --- |
| **Shell Name** | **Developed by** | **Where** | **Remark** |
| BASH ( Bourne-Again SHell ) | Brian Fox and Chet Ramey | Free Software Foundation | Most common shell in Linux. It's Freeware shell. |
| CSH (C SHell) | Bill Joy | University of California (For BSD) | The C shell's syntax and usage are very similar to  the C programming language. |
| KSH (Korn SHell) | David Korn | AT & T Bell Labs | -- |
| TCSH | See the man page.  Type $ man tcsh | -- | TCSH is an enhanced but completely compatible version of the Berkeley UNIX C shell (CSH). |

1. **sh shell**
   1. The Bourne shell (sh), written by Steve Bourne at AT&T Bell Labs, is the original UNIX shell. It is the preferred shell for shell programming because of its compactness and speed. A Bourne shell drawback is that it lacks features for interactive use, such as the ability to recall previous commands (history). The Bourne shell also lacks built-in arithmetic and logical expression handling.
   2. The Bourne shell is the Solaris OS default shell. It is the standard shell for Solaris system administration scripts.
   3. For the Bourne shell the:
      1. Command full-path name is /bin/sh and /sbin/sh.
      2. Non-root user default prompt is $.
      3. Root user default prompt is #.
2. **The C Shell**
   1. The C shell (csh):
      1. Is a UNIX enhancement written by Bill Joy at the University of California at Berkeley.
      2. Incorporated features for interactive use, such as aliases and command history.
      3. Includes convenient programming features, such as built-in arithmetic and a C-like expression syntax.
   2. For the C shell the:
      1. Command full-path name is /bin/csh.
      2. Non-root user default prompt is hostname %.
      3. Root user default prompt is hostname #.
3. **The Korn Shell**
   1. The Korn shell (ksh):
      1. Was written by David Korn at AT&T Bell Labs
      2. Is a superset of the Bourne shell.
      3. Supports everything in the Bourne shell.
      4. Has interactive features comparable to those in the C shell.
      5. Includes convenient programming features like built-in arithmetic and C-like arrays, functions, and string-manipulation facilities.
      6. Is faster than the C shell.
      7. Runs scripts written for the Bourne shell.
   2. For the Korn shell the:
      1. Command full-path name is /bin/ksh.
      2. Non-root user default prompt is $.
      3. Root user default prompt is #.
4. **The GNU Bourne Shell**
   1. The GNU Bourne-Again shell (bash):
      1. Is compatible to the Bourne shell.
      2. Incorporates useful features from the Korn and C shells.
      3. Has arrow keys that are automatically mapped for command recall and editing.
   2. For the GNU Bourne-Again shell the:
      1. Command full-path name is /bin/bash.
      2. Default prompt for a non-root user is bash-x.xx$. (Where x.xx indicates the shell version number. For example, bash-3.50$)
      3. Root user default prompt is bash-x.xx#. (Where x.xx indicates the shell version number. For example, bash-3.50$#)

Here is a short comparison between the four types of shells:

|  |  |  |  |
| --- | --- | --- | --- |
| Shell | Path | Default Prompt (Non-Root user) | Default Prompt (Root user) |
| The Bourne Shell (sh) | /bin/sh and /sbin/sh | $ | # |
| The C Shell (csh) | /bin/csh | % | # |
| The Korn Shell (ksh) | /bin/ksh | $ | # |
| The GNU Bourne-Again Shell (Bash) | /bin/bash | bash-x.xx$ | bash-x.xx# |

**Conclusion:**We have studied various editors and shells in linux.