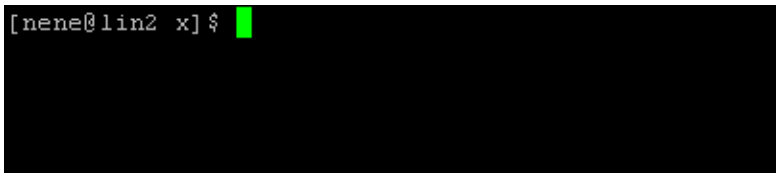
Command line completion



Example of command line completion in Bash.

Command line completion (also **tab completion**) is a common feature of command line interpreters, in which the program automatically fills in partially typed commands.

Command line interpreters are programs that allow a user to interact with the underlying operating system by typing commands at a command prompt using a command line interface (CLI), in contrast to pointing and clicking a mouse in a Graphical User Interface (GUI). Command line completion allows the user to type the first few characters of a command, program, or filename, and press a completion key (normally Tab ↹) to fill in the rest of the item. The user then presses Return or ↵ Enter to run the command or open the file.

Command line completion is useful in several ways, as illustrated by the animated image accompanying this article. Commonly-accessed commands, especially ones with long names, require fewer keystrokes to reach. Commands with long or difficult to spell filenames can be entered by typing the first few characters and pressing a completion key, which completes the command or filename. In the case of multiple possible completions, some command-line interpreters, especially Unix shells, will list all filenames beginning with those few characters. The user can type more characters and press Tab ↹ again to see a new, narrowed-down list if the typed characters are still ambiguous, or else complete the filename. An alternate form of completion rotates through all matching results when the input is ambiguous.

Completable elements may include commands, arguments, file names and other entities, depending on the specific interpreter and its configuration. Command line completion generally only works in interactive mode. That is, it cannot be invoked to complete partially typed commands in scripts or batch files, even if the completion is unambiguous. The name **tab completion** comes from the fact that command line completion is often invoked by pressing the tab key.

History

Tab completion showed up early in computing history; one of the first examples appeared in the Berkeley Timesharing System for the SDS 940, where if a typed string were ambiguous, the interpreter would do nothing, but if the string was *not* ambiguous, it would automatically complete it without any command from the user. This feature did not work well with the all too frequent typos, and so was a mixed blessing. This feature was imitated by Tenex's developers who made an important change: Tenex used "escape recognition", in which the interpreter would not attempt to autocomplete unless the escape key was struck (thus the name) by the user. The domain was also expanded from only program names on the Berkeley system to both program names and files on Tenex.[1] From there it was borrowed by Unix.

Example

To open the file `introduction-to-command-line-completion.html` with Firefox one would type:

```
firefox introduction-to-command-line-completion.html
```

This is a long command to type. Instead we can use command line completion.

Prompting completion

The following example shows how command line completion works in bash. Other command line shells may perform slightly differently.

First we type the first three letters of our command:

```
fir
```

Then we press Tab ↹ and because the only command in our system that starts with "fir" is "firefox", it will be completed to:

```
firefox
```

Then we start typing the file name:

```
firefox i
```

But this time `introduction-to-command-line-completion.html` is not the only file in the current directory that starts with "i". The directory also contains files `introduction-to-bash.html` and `introduction-to-firefox.html`. The system can't decide which of these filenames we wanted to type, but it does know that the file must begin with "introduction-to-", so the command will be completed to:

```
firefox introduction-to-
```

Now we type "c":

```
firefox introduction-to-c
```

After pressing Tab ↹ it will be completed to the whole filename:

```
firefox introduction-to-command-line-completion.html
```

In short we typed:

```
firTab ↹iTab ↹cTab ↹
```

This is just 8 keystrokes, which is considerably less than 52 keystrokes we would have needed to type without using command line completion.

Rotating completion

The following example shows how command line completion works with rotating completion, such as Windows's Command Prompt uses.

We follow the same procedure as for prompting completion until we have:

```
firefox i
```

We press Tab ↹ once, with the result:

```
firefox introduction-to-bash.html
```

We press Tab ↹ again, getting:

```
firefox introduction-to-command-line-completion.html
```

In short we typed:

```
firTab ↹iTab ↹Tab ↹
```

This is just 7 keystrokes, comparable to prompting-style completion. This works best if we know what possibilities the interpreter will rotate through.

Completion in different command line interfaces

- Unix shells, including bash (the default shell in Linux and Mac OS X) and ksh among many others, have a long-standing tradition of advanced and customizable completion capabilities (see the external links section below for some examples).
  - For Korn shell users, file name completion depends on the value of the EDITOR variable. If EDITOR is set to vi, you type part of the name, and then Escape,\. If EDITOR is set to emacs, you type part of the name, and then Escape,Escape.
  - The Z shell (zsh) pioneered the support for fully programmable completion, allowing users to have the shell automatically complete the parameters of various commands unrelated to the shell itself, which is accomplished by priming the shell with definitions of all known switches as well as appropriate parameter types. This allows the user to e.g. type `tar xzf` Tab ↹ and have the shell complete only tarred gzip archives from the actual filesystem, skipping files which are incompatible with the input parameters. A modern zsh installation comes with completion definitions for over five hundred commands.
- Windows PowerShell, the new extensible command shell from Microsoft, which is based on object-oriented programming and the Microsoft .NET framework provides powerful and customizable completion capabilities similar to those of traditional Unix shells.[2] [3]
- The `cmd.exe` command processor of Windows NT-based systems supports basic completion. It is possible to use a separate key-binding for matching directory names only.
- The MS-DOS command processor `COMMAND.COM` did not have command line completion: pressing the tab key would just advance the cursor. Before the release of Windows however, various enhanced shells for MS-DOS, such as 4DOS, or the FreeDOS version of `COMMAND.COM`, featured Unix-style tab completion.

See also

- Autocomplete
- Command line interface
- Comparison of computer shells
- Shell

External links

Unix shells

- A Bash completion overview - "Working more productively with bash 2.x/3.x" by Ian Macdonald [4]
- The zsh completion system, chapter from the Z Shell Manual [5]

Windows command interpreters

(Be sure to check the "Applies to" section in each article)

- Windows Server 2003:
  1. Directory name completion [6]
  2. Filename completion [7]
- Windows XP [8]
- Windows 2000/NT 4 [9]

References

[1] Origins and Development of TOPS-20 (http://www.opost.com/dlm/tenex/hbook.html)
[2] The PowerShell Guy : PowerTab Flash Examples Test (http://thepowershellguy.com/blogs/posh/archive/2007/06/02/powertab-flash-exampes.aspx)
[3] The PowerShell Guy : PowerTab (http://thepowershellguy.com/blogs/posh/pages/powertab.aspx)
[4] http://www.caliban.org/bash/
[5] http://zsh.sourceforge.net/Doc/Release/Completion-System.html
[6] http://technet2.microsoft.com/WindowsServer/en/Library/0a3f5948-84f2-4531-b74a-04dd94227c8d1033.mspx
[7] http://technet2.microsoft.com/WindowsServer/en/Library/f0326a26-8ff7-421c-ba8a-4c9ca945e21f1033.mspx
[8] http://support.microsoft.com/?kbid=310530
[9] http://support.microsoft.com/?kbid=244407

Article Sources and Contributors

Command line completion *Source*: http://en.wikipedia.org/w/index.php?oldid=380115040 *Contributors*: Alerante, Amalas, Angoca, Billposer, Brandenads, Christopherlin, DXBari, Eddpayne, EdgeOfEpsilon, Edward, Endersdouble, Fragglet, Fredrik, Gennaro Prota, Ghettoblaster, IMSoP, Iain.dalton, Incnis Mrsi, Indefatigable, JamesLee, Joy, Lotje, MER-C, Marudubshinki, Mysid, Nil Einne, Pt, R'n'B, Renku, SamSim, Shlomital, Sietse Snel, Tedickey, Theuser, 26 anonymous edits

Image Sources, Licenses and Contributors

Image:Command-line-completion-example.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Command-line-completion-example.gif *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Renku

License