

Shell (computing)

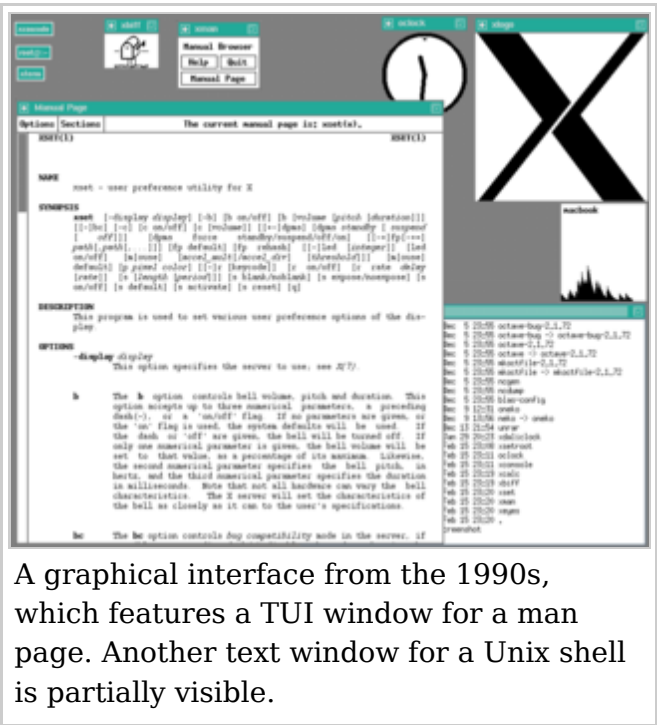
From Wikipedia, the free encyclopedia

In computing, a **shell** is a user interface for access to an operating system's services. In general, operating system shells use either a command-line interface (CLI) or graphical user interface (GUI), depending on a computer's role and particular operation.

The design of a shell is guided by cognitive ergonomics and the goal is to achieve the best workflow possible for the intended tasks; the design can be constricted by the available computing power (for example, of the GPU) or the available amount of graphics memory. The design of a shell is also dictated by the employed computer periphery, such as computer keyboard, pointing device (a mouse with one button, or one with five buttons, or a 3D mouse) or touchscreen, which is the direct human-machine interface.

CLI shells allow some operations to be performed faster, for example rearranging large blocks of data. However, they require the user to memorize all commands and their calling syntax, and also to learn the shell-specific scripting language, for example bash script. CLIs are also easier to be operated via refreshable braille display and provide certain advantages to screen readers.

Graphical shells have a low burden to start using a computer, and they are characterized as being simple and easy to use. With the widespread adoption of programs with GUIs, the use of graphical shells has gained greater adoption. Since graphical shells come with certain disadvantages (for example, lack of support for easy automation of operation sequences), most GUI-enabled operating systems also provide additional CLI shells.



A graphical interface from the 1990s, which features a TUI window for a man page. Another text window for a Unix shell is partially visible.

Contents

- 1 Overview
- 2 Text (CLI) shells

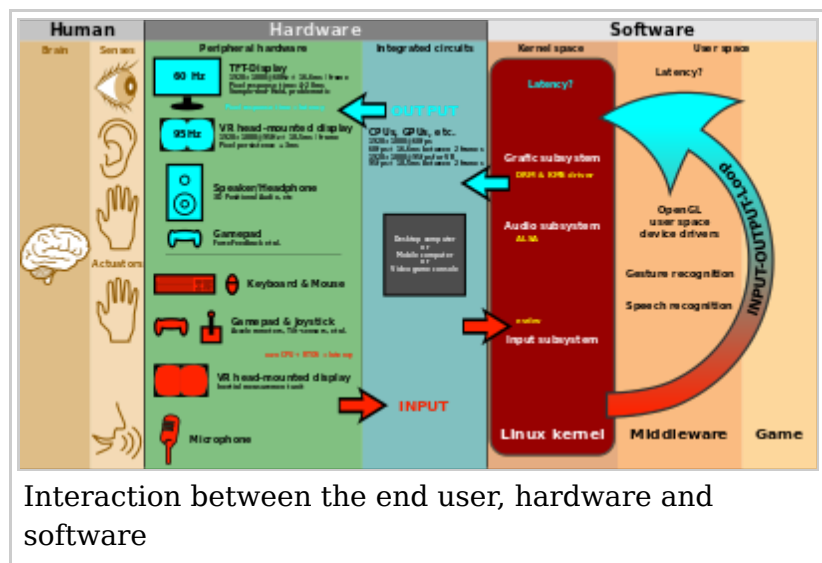
- 3 Graphical shells
 - 3.1 Microsoft Windows
 - 3.2 Unix-like systems
- 4 Other uses
- 5 See also
- 6 References

Overview

Operating systems provide various services to their users, including file management, process management (running and terminating applications), batch processing, and operating system monitoring and configuration.

Most operating system shells are not *direct* interfaces to the underlying kernel, even if a shell communicates with the user via peripheral devices attached to the computer directly. Shells are actually special applications that use the kernel API in just the same way as it is used by other application programs. A shell manages the user-system interaction by prompting users for input, interpreting their input, and then handling an output from the underlying operating system.^[1] Since the operating system shell is actually an application, it may easily be replaced with another similar application, for most operating systems.

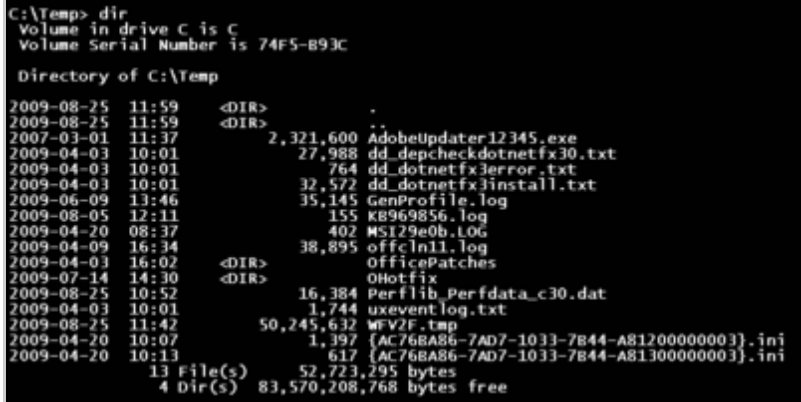
In addition to shells running on local systems, there are different ways to make remote systems available to local users; such approaches are usually referred to as remote access or remote administration. Initially available on multi-user mainframes, which provided text-based UIs for each active user *simultaneously* by means of a text terminal connected to the mainframe via serial line or modem, remote access has extended to Unix-like systems and Microsoft Windows. On Unix-like systems, Secure Shell protocol is usually used for text-based shells, while SSH tunnelling can be used for X Window System-based graphical user interfaces (GUIs). On Microsoft Windows, Remote Desktop Protocol can be used to provide GUI remote access.



Most operating system shells fall into one of two categories – command-line and graphical. Command line shells provide a command-line interface (CLI) to the operating system, while graphical shells provide a graphical user interface (GUI). Other possibilities, although not so common, include voice user interface and various implementations of a text-based user interface (TUI) that are not CLI. The relative merits of CLI- and GUI-based shells are often debated.

Text (CLI) shells

A command-line interface (CLI) is an operating system shell that uses alphanumeric characters typed on a keyboard to provide instructions and data to the operating system, interactively. For example, a teletypewriter can send codes representing keystrokes to a command interpreter program running on the computer; the



```

C:\Temp> dir
Volume in drive C is C
Volume Serial Number is 74F5-B93C

Directory of C:\Temp

2009-08-25  11:59    <DIR>          .
2009-08-25  11:59    <DIR>          ..
2007-03-01  11:37    2,321,600  AdobeUpdater12345.exe
2009-04-03  10:01    27,988    dd_depcheckdotnetfx30.txt
2009-04-03  10:01     764     dd_dotnetfx3error.txt
2009-04-03  10:01    32,572    dd_dotnetfx3install.txt
2009-06-09  13:46    35,145    GenProfile.log
2009-08-05  12:11     155     KB969856.log
2009-04-20  08:37     402     MSI29e0b.LOG
2009-04-09  16:34    38,895    office1n11.log
2009-04-03  16:02    <DIR>          OfficePatches
2009-07-14  14:30    <DIR>          OHotfix
2009-08-25  10:52    16,384    PerfLib_Perfdata_c30.dat
2009-04-03  10:01     1,744    uxevent.log.txt
2009-08-25  11:42    50,245,632  WFV2F.tmp
2009-04-20  10:07     1,397    {AC76BA86-7AD7-1033-7B44-A81200000003}.ini
2009-04-20  10:13     617     {AC76BA86-7AD7-1033-7B44-A81300000003}.ini
                13 File(s)      52,723,295 bytes
                4 Dir(s)  83,570,208,768 bytes free
  
```

Command Prompt, a CLI shell in Windows

command interpreter parses the sequence of keystrokes and responds with an error message if it cannot recognize the sequence of characters, or it may carry out some other program action such as loading an application program, listing files, logging in a user and many others. Operating systems such as UNIX have a large variety of shell programs with different commands, syntax and capabilities. Some operating systems had only a single style of command interface; commodity operating systems such as MS-DOS came with a standard command interface but third-party interfaces were also often available, providing additional features or functions such as menuing or remote program execution.

Application programs may also implement a command-line interface. For example, in Unix-like systems, the telnet program has a number of commands for controlling a link to a remote computer system. Since the commands to the program are made of the same keystrokes as the data being sent to a remote computer, some means of distinguishing the two are required. An escape sequence can be defined, using either a special local keystroke that is never passed on but always interpreted by the local system. The program becomes modal, switching between interpreting commands from the keyboard or passing keystrokes on as data to be processed.

A feature of many command-line shells is the ability to save sequences of commands for re-use. A data file can contain sequences of commands which the

CLI can be made to follow as if typed in by a user. Special features in the CLI may apply when it is carrying out these stored instructions. Such batch files (script files) can be used repeatedly to automate routine operations such as initializing a set of programs when a system is restarted. Batch mode use of shells usually involves structures, conditionals, variables, and other elements of programming languages; some have the bare essentials needed for such a purpose, others are very sophisticated programming languages in and of themselves. Conversely, some programming languages can be used interactively from an operating system shell or in a purpose-built program.

```
chealer@vinci:/usr/share/doc/bash$ export LC_ALL=C
chealer@vinci:/usr/share/doc/bash$ cd ~chealer/
chealer@vinci:~$ ls
Cloutier  Ido      Musique  logs      skolo      sources
Desktop  Mes images boston   ncix.png  smb4k     vieux
chealer@vinci:~$ #Why is there color when calling ls without arguments?
chealer@vinci:~$ which ls
/bin/ls
chealer@vinci:~$ $(!!)
$(which ls)
Cloutier  Ido      Musique  logs      skolo      sources
Desktop  Mes images boston   ncix.png  smb4k     vieux
chealer@vinci:~$ type ls #ls doesn't just run /bin/ls
ls is aliased to `ls --color=auto'
chealer@vinci:~$ echo $PS1
${debian_chroot:+($debian_chroot)}\u@\h:\w\$
chealer@vinci:~$ sh
sh-3.1$ echo $PS1
\s-\v\$
sh-3.1$ echo $BASH_VERSION
3.1.17(1)-release
sh-3.1$ ls
Cloutier  Ido      Musique  logs      skolo      sources
Desktop  Mes images boston   ncix.png  smb4k     vieux
sh-3.1$ echo $SHELLOPTS # ls isn't an alias in POSIX mode
braceexpand:emacs:hashall:histexpand:history:interactive-comments:monitor:posix
sh-3.1$ kill
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill
-l [sigspec]
sh-3.1$ /bin/kill 6> killerror # collect stdout and stderr of $ /bin/kill; in ki
llerror
sh-3.1$ wc -l !$
wc -l killerror
7 killerror
sh-3.1$ type kill # kill doesn't just run /bin/kill, even in POSIX mode.
kill is a shell builtin
sh-3.1$ !$ -n 9 $$ # OK, kill self
kill -n 9 $$ # OK, kill self
Killed
chealer@vinci:~$
```

Bash, a widely adopted Unix shell

The command-line shell may offer features such as command-line completion, where the interpreter expands commands based on a few characters input by the user. A command-line interpreter may offer a history function, so that the user can recall earlier commands issued to the system and repeat them, possibly with some editing. Since all commands to the operating system had to be typed by the user, short command names and compact systems for representing program options were common. Short names were sometimes hard for a user to recall, and early systems lacked the storage resources to provide a detailed on-line user instruction guide.

The first Unix shell, Ken Thompson's *sh*,^[2] was modeled after the Multics shell,^[3] itself modeled after the RUNCOM^[4] program Louis Pouzin showed to the Multics Team. The "rc" suffix on some Unix configuration files (for example, ".vimrc"), is a remnant of the RUNCOM ancestry of Unix shells.

Graphical shells

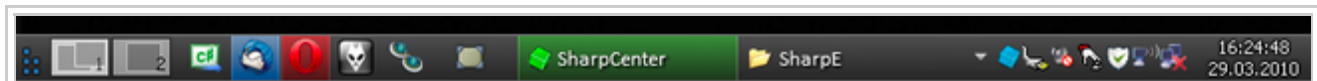
Graphical shells provide means for manipulating programs based on graphical user interface (GUI), by allowing for operations such as opening, closing, moving

and resizing windows, as well as switching focus between windows. Graphical shells may be included with desktop environments or come separately, even as a set of loosely coupled utilities.

Most graphical user interfaces develop the metaphor of an "electronic desktop", where data files are represented as if they were paper documents on a desk, and application programs similarly have graphical representations instead of being invoked by command names.

Microsoft Windows

Modern versions of the Microsoft Windows operating system use the Windows shell as their shell. Windows Shell provides the familiar desktop environment, start menu, and task bar, as well as a graphical user interface for accessing the file management functions of the operating system. Older versions also include Program Manager, which was the shell for the 3.x series of Microsoft Windows, and which in fact ships with later versions of Windows of both the 95 and NT types at least through Windows XP. The interfaces of Windows versions 1 and 2 were markedly different.



The taskbar of SharpEnviro, which is a shell replacement for Windows

Desktop applications are also considered shells, as long as they use a third-party engine. Likewise, many individuals and developers dissatisfied with the interface of Windows Explorer have developed software that either alters the functioning and appearance of the shell or replaces it entirely. WindowBlinds by Stardock is a good example of the former sort of application. LiteStep, SharpE and Emerge Desktop are good examples of the latter.

Interoperability programmes and purpose-designed software lets Windows users use equivalents of many of the various Unix-based GUIs discussed below, as well as Macintosh. An equivalent of the OS/2 Presentation Manager for version 3.0 can run some OS/2 programmes under some conditions using the OS/2 environmental subsystem in versions of Windows NT. For an example of the first, X Window-type environments can be run using combinations of Windows/Unix interoperability packages, communications suites such as Hummingbird Connectivity, and/or X server programmes for Windows such as WinaXe and others.

Unix-like systems

Graphical shells typically build on top of a windowing system. In the case of X Window System or Wayland, the shell consists of the X window manager or the Wayland compositor, as well as of one or multiple programs providing the

functionality to start installed applications, to manage open windows and virtual desktops, and often to support a widget engine.

In the case of OS X, Quartz could be thought of as the windowing system, and the shell consists of the Finder,^[5] the Dock,^[5] SystemUIServer,^[5] and Mission Control.^[6]

Other uses

"Shell" is also used loosely to describe application software that is "built around" a particular component, such as web browsers and email clients, in analogy to the shells found in nature.

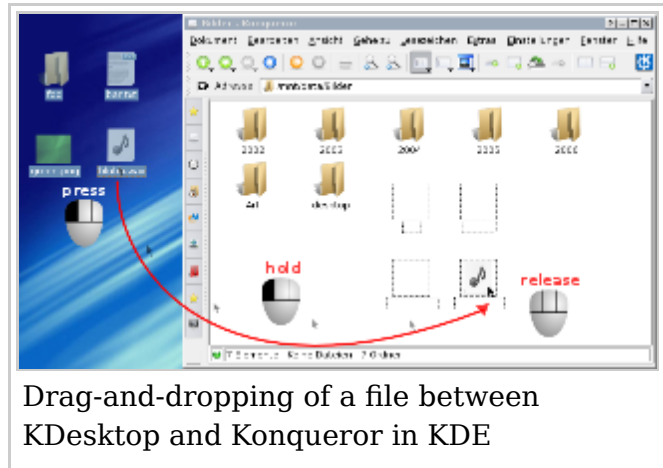
In expert systems, a shell is a piece of software that is an "empty" expert system without the knowledge base for any particular application.^[7]

See also

- Comparison of command shells
- DOS Shell
- Human-computer interaction
- Internet Explorer shell
- Shell account
- Shell builtin
- Window manager – provides a rudimentary process management interface

References

1. ^ "Operating system shells" (<http://pic.dhe.ibm.com/infocenter/aix/v6r1/index.jsp?topic=%2Fcom.ibm.aix.baseadm%2Fdoc%2Fbaseadmdita%2Fshells.htm>) . *AIX 6.1 Information Center*. IBM Corp. Retrieved September 16, 2012.
2. ^ "V6 Thompson Shell Port - History" (<http://v6shell.org/history/>). V6shell.org. Retrieved 2012-08-14.
3. ^ Tom Van Vleck (1995-02-05). "Unix and Multics" (<http://www.multicians.org/unix.html>). Multicians.org. Retrieved 2012-08-14.
4. ^ Louis Pouzin (2000-11-25). "The Origin of the Shell" (<http://www.multicians.org/shell.html>). Multicians.org. Retrieved 2012-08-14.



Drag-and-dropping of a file between KDesktop and Konqueror in KDE

5. ^ **a b c** "The Life Cycle of a Daemon" (<https://developer.apple.com/library/mac/documentation/macosx/conceptual/bpsystemstartup/chapters/Lifecycle.html>). Apple Inc.
6. ^ "Restart Mission Control in OS X Lion" (<http://osxdaily.com/2011/11/23/restart-mission-control-in-os-x-lion/>). OSXDaily. Nov 23, 2011.
7. ^ *British Computer Society: The BCS glossary of ICT and computing terms* (<http://books.google.com/books?id=g8Bds8ssYYgC&pg=PA135&dq=%22shell+is+a+piece%22+%22expert+system%22&hl=sv#v=onepage&q=%22shell%20is%20a%20piece%22%20%22expert%20system%22&f=false>). Pearson Education. 2005. p. 135. ISBN 978-0-13-147957-9.

Retrieved from "[http://en.wikipedia.org/w/index.php?title=Shell_\(computing\)&oldid=619946655](http://en.wikipedia.org/w/index.php?title=Shell_(computing)&oldid=619946655)"

Categories: Command shells | Desktop environments

- This page was last modified on 5 August 2014 at 11:46.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.