I'm not robot

reCAPTCHA

Continue

Command line user interface pdf

Speak to us to find how you can achieve cloud elasticity with a serverless messaging queue and background task solution with free handeld support.If you're new to programming, CLI is essentially a command-line program that utilizes text to execute functions within an operating system. Commonly used in the 1960s, 1970s, and into the1980s, this approach would allow users to interact with an operating system or application via a keyboard. Users would type commands in the CLI to achieve a specific output or response (i.e. a table, messages, list, etc.). Prior to the mouse, this was the only way to communicate with a computer. Today, most users no longer use command-line interfaces due to the development of GUI. However, CLI is still used among developers and admins when installing software, configuring computers, and accessing features that aren't otherwise available with GUI. The MS-DOS operating system and the command shell in Windows are examples of CLIs. Programming languages such as Python also support CLIs. When using a command line, you can perform nearly all of the same tasks that can be completed with a GUI. However, the majority of tasks can be performed more easily and more rapidly. As a user, you would enter a command, press enter, and then wait for a response. Since there are hundreds of different commands, it can take time to learn and remember the most important commands.One of the main advantages of a command line interface is that it allows users to type in commands that can produce immediate results. Those familiar with command-lines find this method quick and simple. Tasks can also be automated using a batch file, for example. A CLI can also enhance security. In 2015, the cost of malware around the globe sat at $500 billion. However, at the current trajectory, the total global cost is expected to reach $6 trillion by 2021. When it comes to adware, viruses, and malware, a command-line interface is a highly effective tool. Security professionals often overlook this simple method, opting for more complex GUI-based tools. However, the Windows command interface is easy-to-use and far less complicated, which is a huge pro.To summarize, the main advantages of a command-line interface include:If you know the commands, a CLI can be a lot faster and efficient than any other type of interface. It can also handle repetitive tasks easily. A CLI requires less memory to use in comparison to other interfaces. It also does not use as much CPU processing time as other interfaces. A CLI doesn't require Windows and a low-resolution monitor can be used. Meaning, it needs fewer resources, yet is highly precise. In contrast, the main disadvantage of a command-line interface is that users must be familiar with the commands in order to maximize the above advantages. In that sense, there can be a fairly steep learning curve if you're new to programming. Since a command-line interface is strictly text-based, knowing the language is crucial. There is also virtually zero room for error. You will be required to memorize many different commands in order to get the most value out of using a command-line interface. If you type the wrong command, you could accidentally delete files or close the program before you saved your work. This is considered to be the main disadvantage of using CLI. To summarize, some of the main disadvantages include:If you are new to programming or have never used a CLI, this approach can be confusing. There are also a lot of commands that need to be learned (i.e. in this case of Unix, there are hundreds of commands). Accuracy is of the utmost importance. If there is a spelling error, a command will fail. Also, if an instruction is mistyped, you'll often need to start from scratch. Bottom line: If you need to automate commonly performed tasks, manage files on a web or file server, work with computers remotely, or need ways to manage files more easily and rapidly, you'll want to consider the advantages associated with the command line. Find out how IronWorker and IronMQ can help your application obtain the cloud with fanatical customer support, reliable performance, and competitive pricing. A Command line Interface (or CLI) is a text-based interface used for entering commands. In the early days of computing, before the mouse, it was the standard way to interact with a computer. While the graphic user interface (GUI) has largely replaced CLIs, they are still included with several major operating systems, such as Windows and OS X.[1] CLI is an older method for interacting with applications and operating systems and is used to perform specific tasks required by users. CLI is a text-based interface, unlike the GUI, which uses graphical options that enable the user to interact with the operating system and applications. CLI allows a user to perform tasks by entering commands. Its working mechanism is very easy, but it is not user friendly. Users enter the specific command, press "Enter", and then wait for a response. After receiving the command, the CLI processes it accordingly and shows the output/result on the same screen; command line interpreter is used for this purpose. CLI was introduced with the teletypewriter machine. This system was based on batched processing. Modern computers support CLI, batch processing and GUI in one interface. In order to best make use of CLI, a user must be able to enter a bundle of commands (one by one) quickly. There are many applications (mono-processing systems) that still use CLI for their operators. In addition, some programming languages, such as Forth, Python and BASIC, offer CLI. Command line interpreter is used to implement the text-based interface. Another feature of CLI is command prompt, which is employed as a sequence of characters used in the user interface, or shell. Command prompt is used to notify users that CLI is ready to accept commands. MS-DOS is the best example of CLI.[2] Shell – The Foundation Behind CLI[3] If we dive from CLI into the deeper part of an operating system, we will find the shell. Shell is a user interface responsible for processing all commands typed on CLI. It reads and interprets the commands and instructs the operating system to perform tasks as requested. In other words, a shell is a user interface that manages CLI and acts as the man-in-the-middle, connecting users with the operating system. In practice, there are many things that shell can process, such as: Among many types of shell, the most popular ones are Windows shell (for Windows) and bash (for Linux and MacOS). History of Command-line Interface[4] The command-line interface evolved from a form of dialog once conducted by humans to interact with a computer. In which human operators remotely exchanged information, usually one line of text at a time. Early computer systems often used teleprinter machines as the means of interaction with a human operator. The computer became one end of the human-to-human teleprinter model. So instead of a human communicating with another human over a teleprinter, a human communicated with a computer. The mechanical teleprinter was replaced by a "glass tty", a keyboard and screen emulating the teleprinter. "Smart" terminals permitted additional functions, such as cursor movement over the entire screen, or local editing of data on the terminal for transmission to the computer. As the microcomputer revolution replaced the traditional – minicomputer + terminals – time sharing architecture, hardware terminals were replaced by terminal emulators — PC software that interpreted terminal signals sent through the PC's serial ports. These were typically used to interface an organization's new PC's with their existing mini- or mainframe computers, or to connect PC to PC. Some of these PCs were running Bulletin Board System software. Early operating system CLIs were implemented as part of resident monitor programs, and could not easily be replaced. The first implementation of the shell as a replaceable component was part of the Multics time-sharing system. In 1964, MIT Computation Center staff member Louis Pouzin developed the RUNCOM tool for executing command scripts while allowing argument substitution. Pouzin coined the term "shell" to describe the technique of using commands like a programming language, and wrote a paper about how to implement the idea in the Multics operating system. Pouzin returned to his native France in 1965, and the first Multics shell was developed by Glenda Schroeder. The first Unix shell, the V6 shell, was developed by Ken Thompson in 1971 at Bell Labs and was modeled after Schroeder's Multics shell. The Bourne shell was introduced in 1977 as a replacement for the V6 shell. Although it is used as an interactive command interpreter, it was also intended as a scripting language and contains most of the features that are commonly considered to produce structured programs. The Bourne shell led to the development of the Korn shell (ksh), Almquist shell (ash), and the popular Bourne-again shell (or Bash). Early microcomputers themselves were based on a command-line interface such as CP/M, DOS or AppleSoft BASIC. During the 1980s and 1990s, the introduction of the Apple Macintosh and of Microsoft Windows on PCs saw the command line interface as the primary user interface supplanted by the Graphical User Interface. The command line remained available as an alternative user interface, often used by system administrators and other advanced users for system administration, computer programming and batch processing. In November 2006, Microsoft released version 1.0 of Windows PowerShell (formerly codenamed Monad), which combined features of traditional Unix shells with their proprietary object-oriented .NET Framework. MinGW and Cygwin are open-source packages for Windows that offer a Unix-like CLI. Microsoft provides MKS Inc.'s ksh implementation MKS Korn shell for Windows through their Services for UNIX add-on. Since 2001, the Macintosh operating system macOS has been based on a Unix-like operating system called Darwin. On these computers, users can access a Unix-like command-line interface by running the terminal emulator program called Terminal, which is found in the Utilities sub-folder of the Applications folder, or by remotely logging into the machine using ssh. Z shell is the default shell for macOS; bash, tcsh, and the Korn shell are also provided. Before macOS Catalina, bash was the default. CLI Advantages and Disadvantages[5] The advantages of a command line interface are: granular control of an OS or application; faster management of a large number of operating systems; ability to store scripts to automate regular tasks; and basic command line interface knowledge to help with troubleshooting, such as network connection issues. The disadvantages of a command line interface are: GUI is more user-friendly; steeper learning curve associated with memorizing commands and complex syntax/arguments; and different commands in different shells. See Also References ↑ Definition - What is Meant by Command-line Interface (CLI) Techterms ↑ Explaining Command-line Interface (CLI) Techopedia ↑ Shell – The Foundation Behind CLI Hostinger ↑ History of Command-line Interface Wikipedia ↑ CLI Advantages and Disadvantages Techtarget command line user interface is used in operating system like. command line user interface examples. command line user interface advantages and disadvantages. command line user interface operating system. command line user interface definition. command line user interface java. command line user interface python. command line user interface pdf