* **graphical user interfaces** (GUI): windows, icons and pointers.
* shell provides - a simple language and a **command-line interface (CLI)**
* The heart of a command-line interface is a **read-evaluate-print loop** (REPL).

type a command - press Return - reads command – executes - output - loops back – waits

* The first line shows only a **prompt**, indicating that the shell is waiting for input.
* *do not type the prompt*
* structure: a **command**, some **flags** (also called **options** or **switches**) and an **argument**.
* **Flags** start with a single dash (-) or two dashes (--)

change the behavior of a command.

* Arguments tell the command what to operate on (e.g. files and directories).
* capitalization matters: LS is different to ls
* Each part is separated by spaces
* A **shell is a program** whose primary purpose is to read commands and run other programs.
* The shell’s main advantages are its **high action-to-keystroke ratio**, its support for **automating repetitive tasks**, and its capacity to **access networked machines**.
* The shell’s main disadvantages are its **primarily textual nature** and how **cryptic** **its commands and operation** can be.
* The part of the operating system responsible for managing files and directories is called the **file system**
* **Current working directory =\ home directory**
* Two meanings for the / character (**Slash**).
* When it appears at the front of a file or directory name, it’s the root directory.
* When it appears inside a name, it’s just a separator.
* About “..”, For example, if we are in /Users/nelle/data, the command ls .. will give us a listing of /Users/nelle
* **absolute path** to a directory by including its entire path from the root directory, which is indicated by a leading slash.
* the character ~ (tilde) at the start of a path to mean “the current user’s home directory”. For example, if Nelle’s home directory is /Users/nelle, then ~/data is equivalent to /Users/nelle/data.
* cd will translate - into the previous directory I was in
* cd path changes the current working directory.
* ls path prints a listing of a specific file or directory; ls on its own lists the current working directory.
* A **relative path** specifies a location starting from the current location.
* An **absolute path** specifies a location from the root of the file system.
* rm -i : the -i option will prompt before every removal. **To remove safely**
* rm -r direc 可以整个删除direc 或者 先rm files 然后 rmdir direc
* ls can be given multiple paths at once
* \* is a **wildcard**. It matches zero or more characters
* ? is also a wildcard, but it only matches a single character
* the command less, which you use with $ less lengths.txt. This displays a screenful of the file, and then stops.
* **Pipe** tells the shell that we want to use the output of the command on the left as the input to the command on the right 连接两个cmd
* use < to redirect its input, i.e., to read from a file instead of from standard input.
* In a loop, the $ tells the shell interpreter to treat the **variable** as a variable name and substitute its value in its place.
* echo, just prints its command-line arguments to standard output.

echo $filename just prints the name of the file

* to **stop** the loop procession, typing Ctrl-C
* move to the beginning of a line in the shell by typing Ctrl-a and to the end using Ctrl-e.
* !$ retrieves the last word of the last command.
* Use Ctrl-R to search through the previously entered commands.
* Use history to display recent commands, and !number to repeat a command by number.
* grep **-E** '^.o' haiku.txt the -E flag and put the pattern in quotes to prevent the shell from trying to interpret it. The ^ in the pattern anchors the match to the start of the line. The . matches a single character
* find finds files with specific properties that match patterns.