Shell Tools and Scripting

Double Quotes (") vs Single Quotes (')

- **Double Quotes**:
  - Expands variables and commands.
  - Preserves whitespace.
  - Allows for string interpolation.
- **Single Quotes**:
  - Preserves the literal content.
  - Does not expand variables or commands.
  - Does not allow for string interpolation.

Dollar Sign ($)

- `$variable`: Value of the variable.
- `$0`: Name of the shell script.
- `$1`, `$2`, ...: Arguments passed to the script.
- `$_`: Last argument of the previous command.
- `$?`: Exit status of the last command.
- `$#`: Number of arguments passed to the script.
- `$@`: All arguments passed to the script.
- `$$`: Process ID of the current shell.
- `$(command)`: Get the output of a command.

Other Signs

- `#`: Comment.
- `;`: Command separator.
- `&&`: Logical AND.
- `||`: Logical OR.
- `!`: Logical NOT.
- `&`: Run command in the background.
- `>`: Redirect output to a file.
- `<`: Redirect input from a file.
- `>>`: Append output to a file.

Useful Commands

- `echo`: Print text to the console.
- `pwd`: Print the current working directory.
- `source`: Source a script.
- `grep`: Search for a pattern in a file.
- `curl`: Transfer data from or to a server.
- `date`: Display or set the system date and time.
- `touch`: Create an empty file or update the timestamp of an existing file.
- `diff`: Compare files line by line.

- `find`: Search for files and directories. E.g., : `find . -name "*.txt"` find all files with the .txt extension in the current directory and its subdirectories. `fin . -name src -type d` find all directories named src in the current directory and its subdirectories. `find . -mtime -1`: Find all files modified in the last 24 hours.
- `fd "string|regex"`: Search for files and directories using the fd utility by regex expression.
- `locate`: Search for files and directories using the locate utility. **On macOS, you need to create a database first. To create the database, run the following command `sudo launchctl load -w /System/Library/LaunchDaemons/com.apple.locate.plist`**
- `history`: Display the history of commands.
- `back-research`: Press **Control + R** to search backwards in the history.

Fun Commands

- `cat <(ls) <(ls ..)` Concatenate the output of two commands.
- `diff <(ls foo) <(ls bar)` Compare the output of two commands.
- `./example.sh mcd.sh script.py example.sh` We can even feed the own scripts to itself.
- `touch foo{,1,2,10}` Create files `foo`, `foo1`, `foo2`, and `foo10`.
- `touch project{1,2}/src/test/test{1..3}.py` Create files `test1.py`, `test2.py`, and `test3.py` in the `src/test` directories of `projects1` and `project2`. `{a..z}` also works.
- `find . -name "*.tmp" -exec rm {} \;`: Remove all files with the.tmp extension.
- `grep -R "foo" *`: Search for the string "foo" in recursive mode.
- `rg "import requests" -t py -C 5 ~/scratch`: Search for the string "import requests" in the `py` files in the `~/scratch` directory and display the matching line and the 5 lines before and after each match. For more about `rg`, look up "Useful External Tools" below.
- `rg -u --files-without-math "^#\!" -t sh`: Find all files in the current directory and its subdirectories that do not start with the string "#!" and have the.sh extension.

Wildcards

- `ls *.sh`: List all files with the .sh extension.
- `ls project?`: List all files starting with project and tail with a single character.

She Bang (#!)

`#!/usr/local/bin/python` is the shebang line. It specifies the interpreter that should be used to run the script. In this case, it specifies that the script should be run using the Python interpreter located at `/usr/local/bin/python`. `#!/usr/bin/env python` is also a shebang line. It specifies the interpreter that should be used to run the script. In this case, it specifies that the script should be run using the Python interpreter that is specified in the `PATH` environment variable. It do help to share your scripts with others if they install their Python interpreter and add it to their `PATH`.

Useful External Tools

- `convert`: Image manipulation. You should manual install it by `brew install imagemagick`.
- `shellcheck`: Check shell scripts for common errors. It will help you do some debugging. You should manual install it by `brew install shellcheck`.
- `rg`: Search for text in files. You should manual install it by `brew install ripgrep`.

- `nnn`: Interactive terminal file manager and disk usage analyzer. You should manual install it by `brew install nnn`.
- `fzf`: Command-line fuzzy finder. You should manual install it by `brew install fzf`.

Construct expressions (Test)

You can get some help from `man test`. Here are some examples:

- `[ $a -eq $b ]`: Compare two numbers.
- `[ $a -gt $b ]`: Check if `$a` is greater than `$b`.
- `[ $a -lt $b ]`: Check if `$a` is less than `$b`.
- `[ $a -ge $b ]`: Check if `$a` is greater than or equal to `$b`.
- `[ $a -le $b ]`: Check if `$a` is less than or equal to `$b`.
- `[ $a -ne $b ]`: Check if `$a` is not equal to `$b`.
- `-b file`: Check if `file` exists and is a block special file.
- `-c file`: Check if `file` exists and is a character special file.
- `-d file`: Check if `file` exists and is a directory.
- `-e file`: Check if `file` exists (regardless of type).
- `-f file`: Check if `file` exists and is a regular file.
- `-g file`: Check if `file` exists and its set group ID flag is set.
- and so on...

Two Manual Pages Checker

Man

`man` is a command-line utility that allows you to view the manual pages of commands. You can use it to get information about a command, its options, and its usage. For example, to get information about the `ls` command, you can run `man ls`. `man` is also work for external tools. For example, to get information about the `convert` command, you can run `man convert`. **Notes that `man` is not work for some tools that are not have a manual page.**

tldr

`tldr` is a command-line utility that allows you to view the simplified and condensed manual pages of commands. You can use it to get a quick overview of a command, its options, and its usage. For example, to get a quick overview of the `ls` command, you can run `tldr ls`. Most of utility installed by `brew` have a manual page. So `man tldr` also works.

Puzzling Sample

```
grep foobar "$file" > /dev/null 2> /dev/null
```

This command searches for the string `"foobar"` in the file specified by `$file`. The output is redirected to /dev/null, which discards the output. The error output is also redirected to /dev/null, which discards the error output. This command is used to suppress the output of the grep command. **`2>` means redirect the error output to the file behind it.**