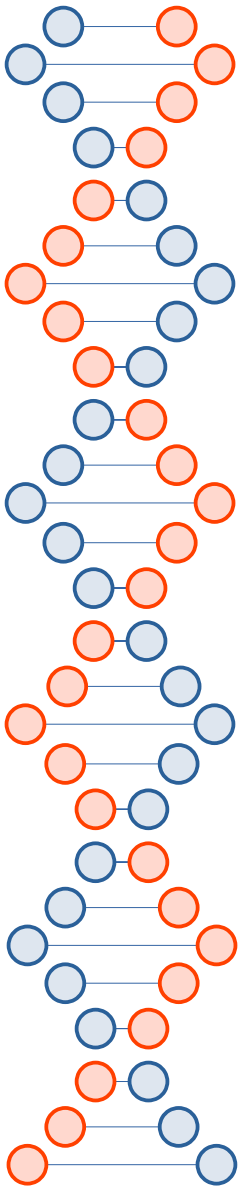
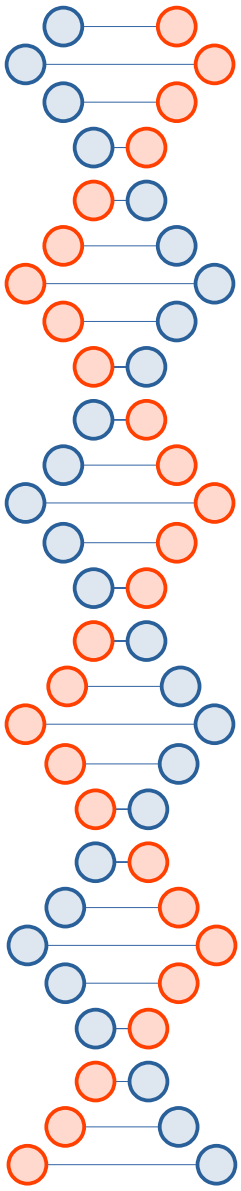


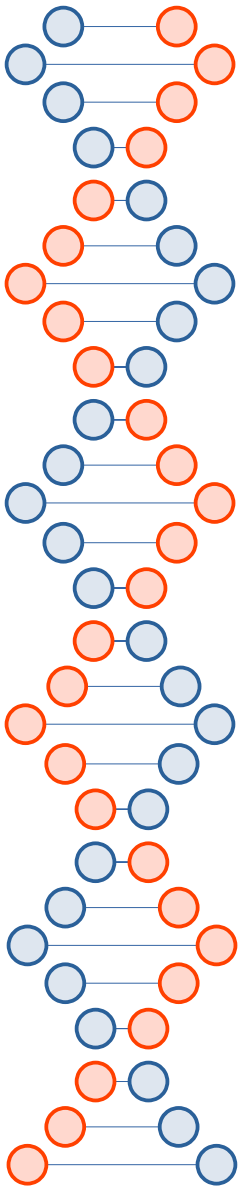
Introduction to Shell Scripting (finally!)

Skylar Chan - BSCI238G
UMD
April 1st, 2022



April Fools!

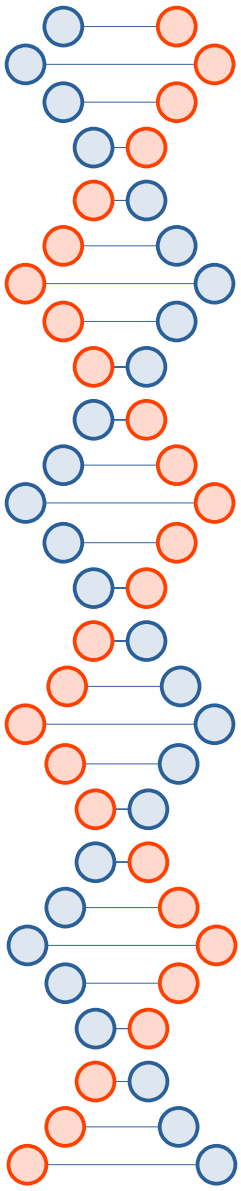
- http://error418.net/
- https://google.com/teapot
- https://httpbin.org/status/418
- https://datatracker.ietf.org/doc/html/rfc2324



Students will be able to

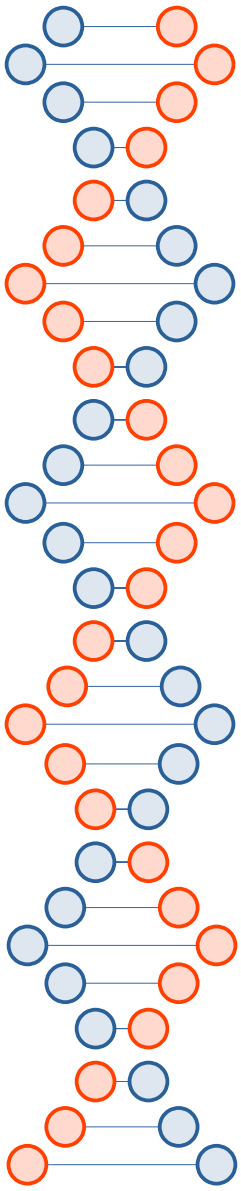
- Use printf
- Bash order of operations
- Variables
- Word splitting/quoting
- Command substitution

Swoobat



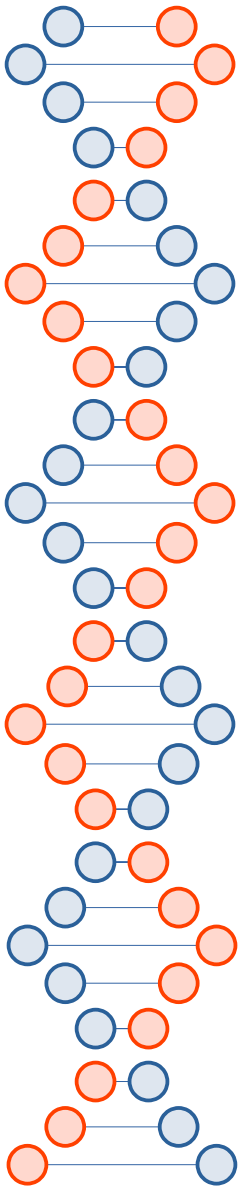
printf

- Has more POSIX-defined behaviors than echo does
- Generally recommended for scripting
- printf format-string words
- %s is substituted with the next word
- **printf ‘%s\n’ 1 2 3 4**
- **printf ‘%s\n%s’ 1 2 3 4**



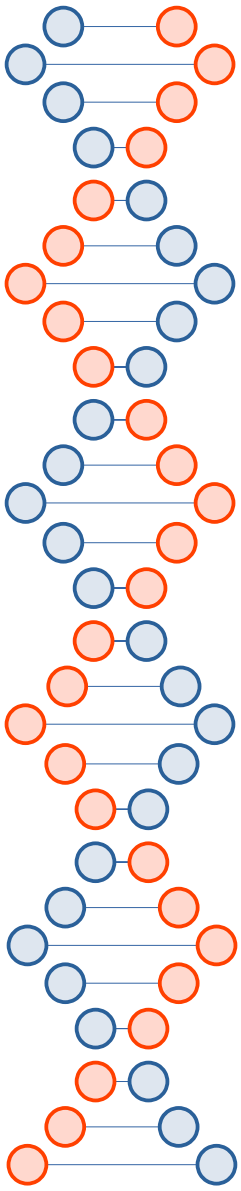
Variables

- Temporary storage in the shell
- **VAR=1 ; echo “$VAR”**
- Variables are cleared with every new shell session
  - Typically lowercase
- *Environment variables* persist through shell session and subshells
  - Typically uppercase
- **bash -c ‘echo $var’**
- **export VAR && bash -c ‘echo $var’**
- Quotes will be explained soon



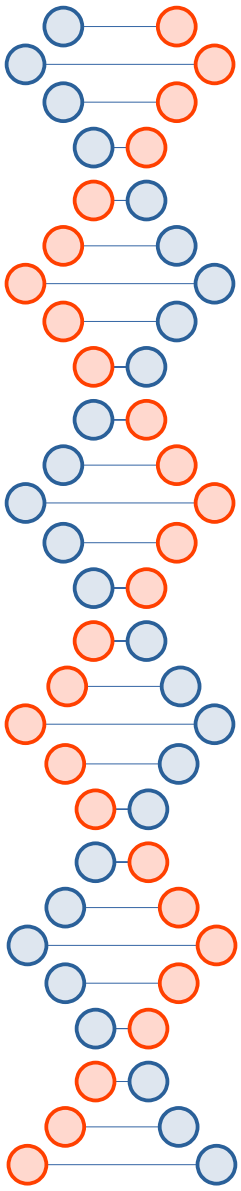
Environment variables

- **env | less**
- Important
  - $IFS
  - $SHELL
  - $HOME
  - $USER
- Less important but nice to know
  - $PAGER
  - $EDITOR
  - $TERMINAL



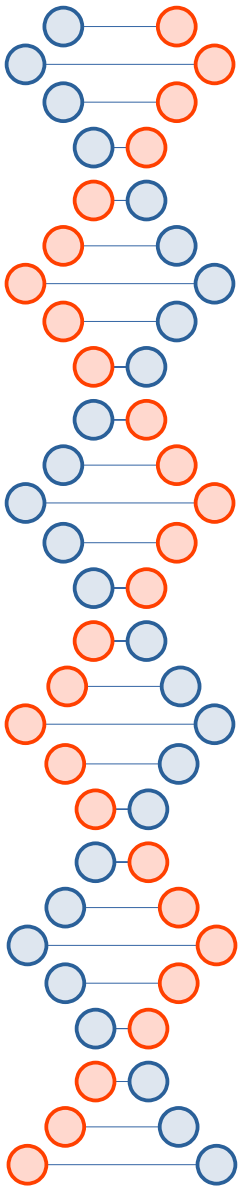
Order of expansions

- Brace expansion
- From left to right
  - Tilde expansion
  - Parameter and variable expansion
  - Arithmetic expansion
  - Command substitution
- Word splitting
- Filename expansion
- Quote removal



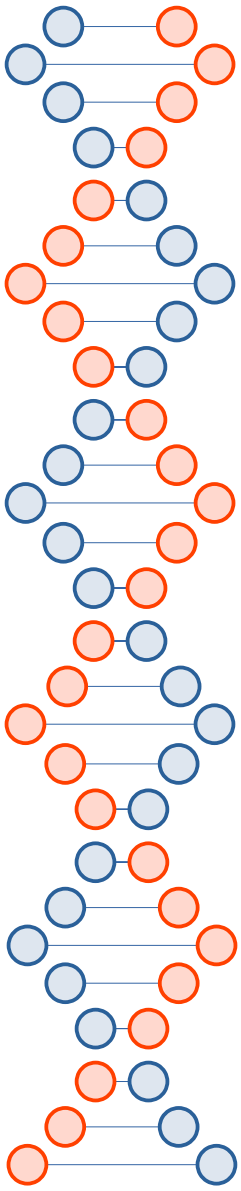
Brace expansion

- Patterns in braces are expanded into multiple words
- **printf ‘%s%s\n’ {1..4}**
- **printf ‘%s\n’ {200,404,418}**
- Useful for interactive purposes
- **cp file{,.copy}**
- **mv file{.txt,.sh}**



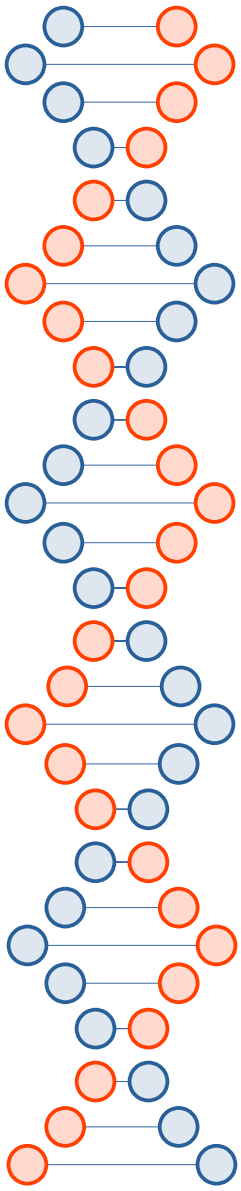
Tilde expansion

- ~ → $HOME
- A bunch of other stuff I have never used
- That’s all you need to know



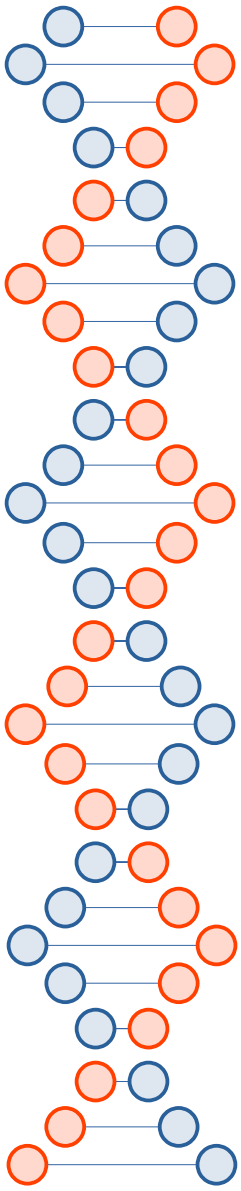
Parameter and variable expansion

- Variables are substituted with their contents
- **printf ‘%s\n’ “$var”**
- **printf ‘%s\n’ “${var}”** is exactly the same
- **printf ‘%s\n’ “${stuff var stuff}”** can be wildly different!
- Basically a bunch of *built-in* string utilities are provided (length, text substitution, etc)
- Faster than using external commands but hard to read and remember
- That’s all you need to know



Arithmetic expansion

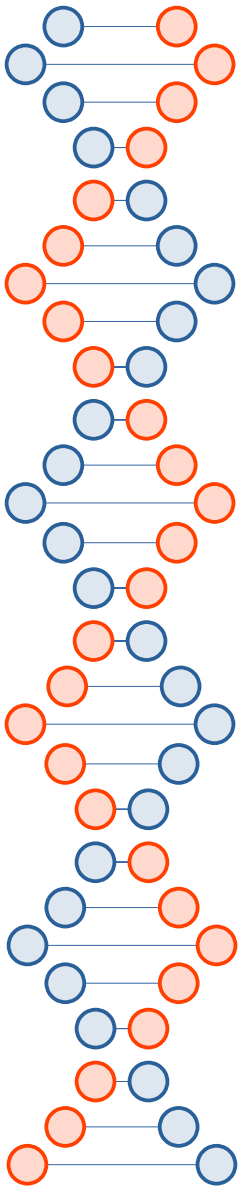
- ((math expression))
- **i=1 ; u=2**
- **printf ‘%s\n’ “$((3 * u))”**
- **((i < 3 * u)) && echo ‘I love you’**



Command substitution

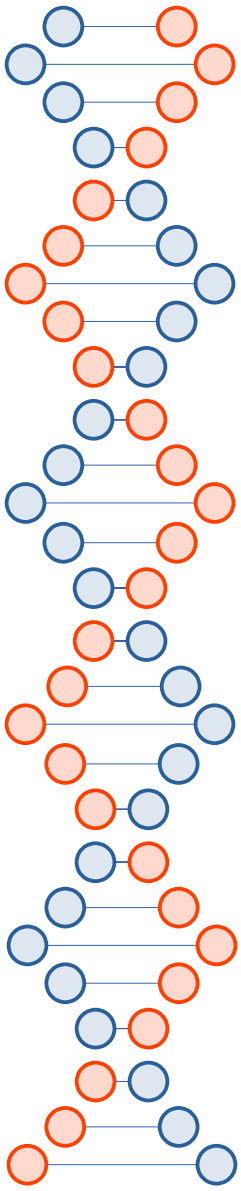
- $(cmd) → output of that command
- `cmd` is the same but is not recommended
- Application: print the result of commands
- **thedate=“$(date)”**
- **printf 'The date is %s\n' "$thedate"**

https://unix.stackexchange.com/questions/440088/what-is-command-substitution-in-a-shell



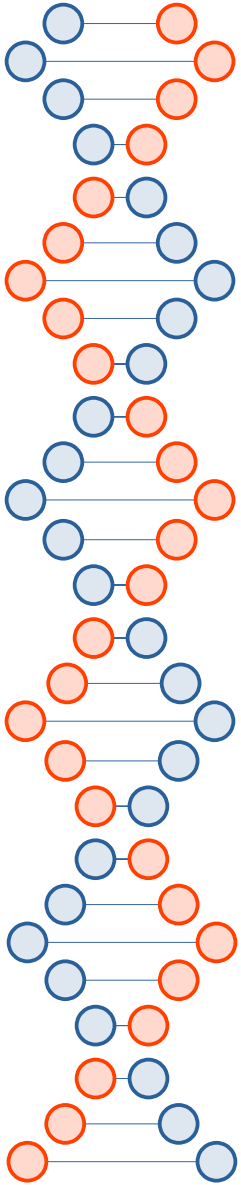
Single/Double quotes

- Expansions do not happen in ‘$single quotes’
- Expansions happen in $double quotes
- Expansions in “$quoted variables” and are not word split
- **Escape quotes and dollar signs or use single quotes**
  - **echo “\$\””**
  - **echo ‘$\’**
- Combine quoted strings which are concatenated
  - **echo “BSCI”‘238G’**



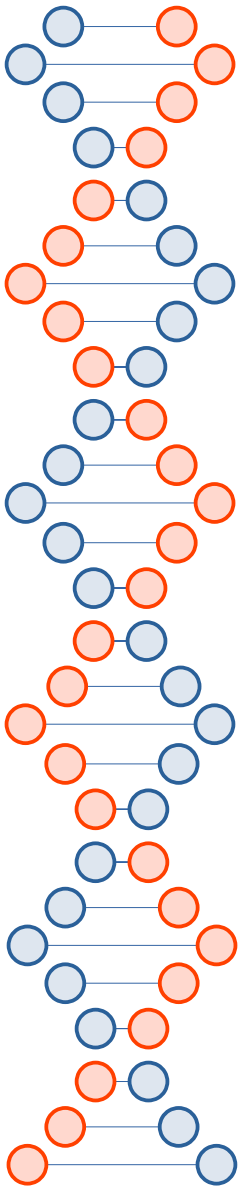
Word splitting

- Occurs when expanded $variables are unquoted
- Typically words are split by spaces
  - Use IFS= to control word splitting
- Use with caution!
- **file='file with spaces in it'**
- **rm $file → rm file with spaces in it**
- Now you've just removed 5 files, none of which were your intended file…
- This is why we (almost) always (double-)quote variables



Quote removal

- Quotes are removed at the end
- **echo a**
- **echo ‘a’**
- **echo “a”**



The End

Today’s Terminal Toy:
nyancat