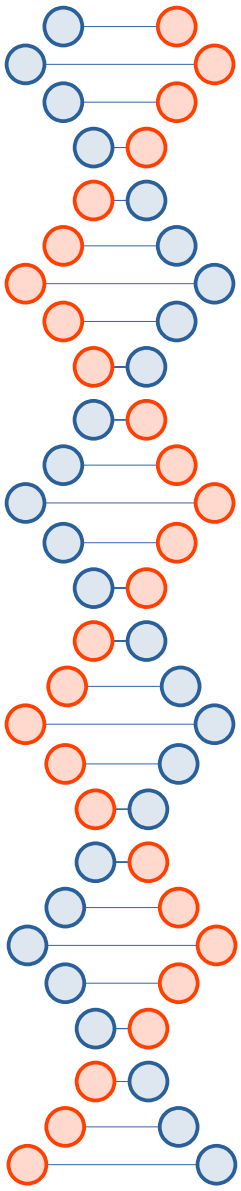


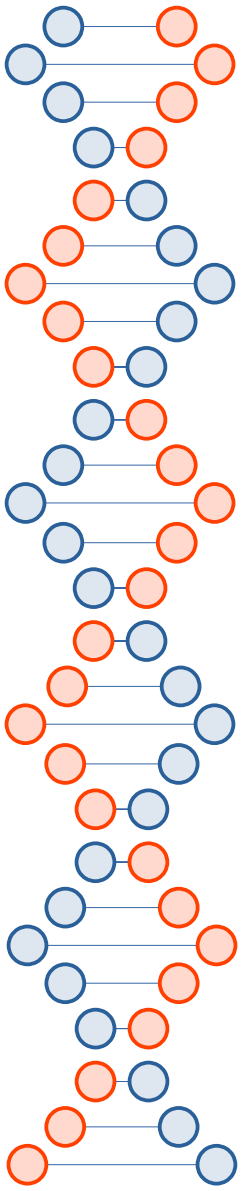
Introduction to Shell Scripting
The end!

Skylar Chan - BSCI238G
UMD
May 6th, 2022

Students will be able to

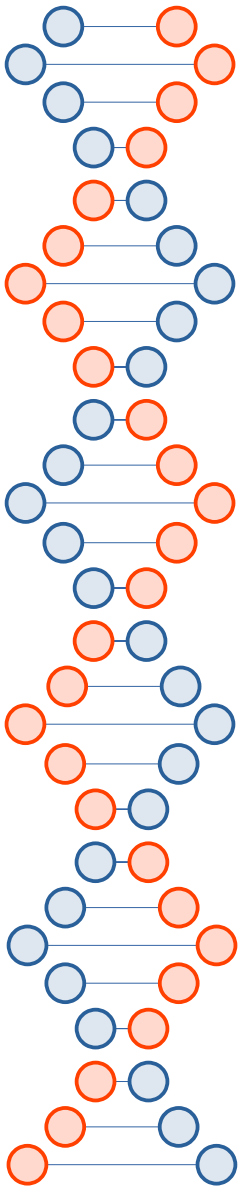
- Find/xargs
- Test, [, [[
- String slicing
- Use shellcheck
- Bash strict mode
- Avoid eval and expr
- Use tar, zip, gzip
- Understand package management

Swoobat



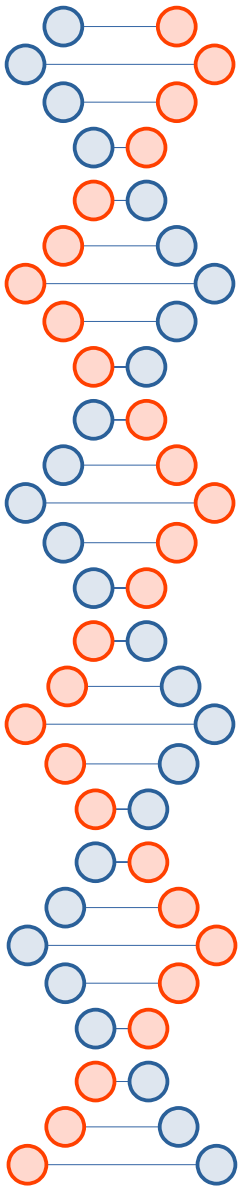
Change your shell to Bash!

- You might be running tcsh…
- **echo $SHELL**
- **chsh**
  - Change to */bin/bash*



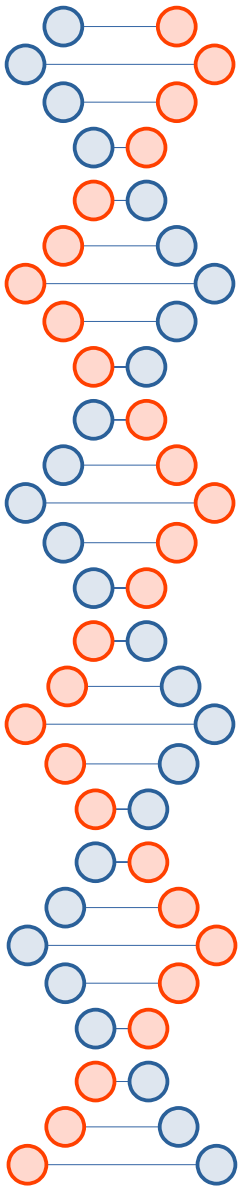
Important dates

- Final project due 5/16
- Include:
  - Documentation
  - Program log
    - **your-program | tee > program.log**
- Tip: you can automate your program's input for testing
  - **Your-program < your-input.txt | tee > program.log**



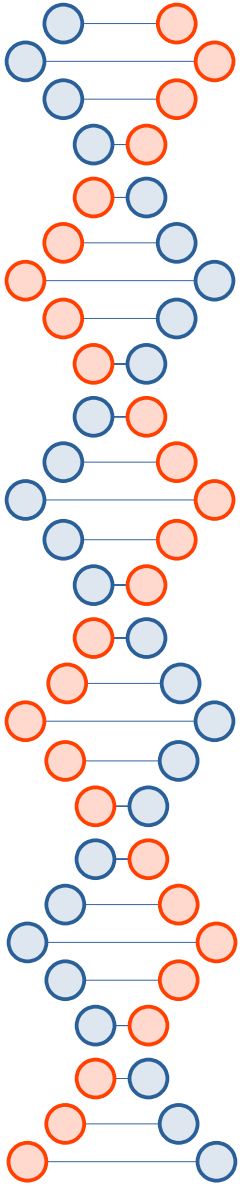
Find/xargs

- Find files
- Run commands with multiple arguments
- These are really hard honestly so looking up examples is the most helpful
- Notes:
  - find -print0 | xargs -0 for files that have spaces



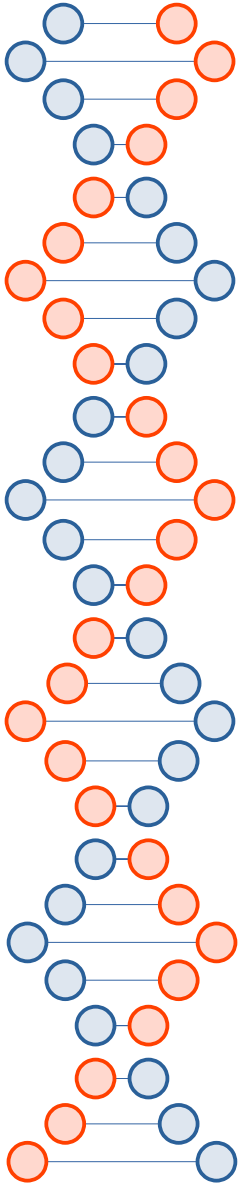
Inside a test statement

- -eq is for numeric equality
- = and == are for string equality
- Single quotes [ ] will exit 1 if a string is given for -eq
- Double quotes [[ ]] will compare ascii if a string is given for -eq



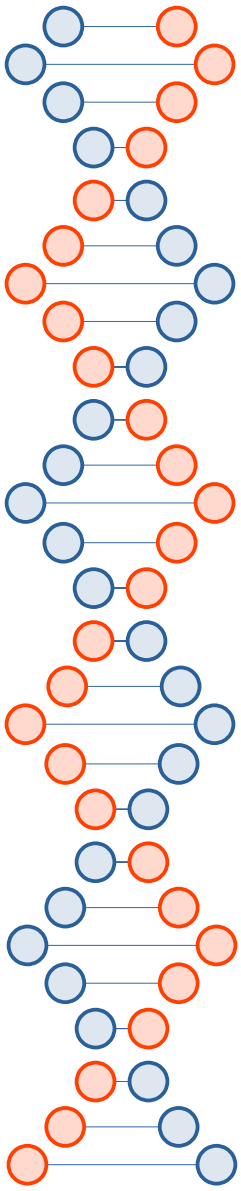
String slicing

- **string=abcdefghj**
- **start=3**
- **length=5**
- **echo “${string:start:length}”**



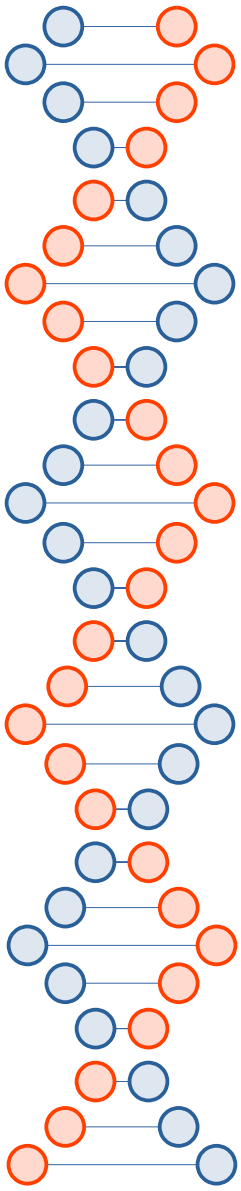
Shellcheck

- https://www.shellcheck.net/
- A shell script static analysis tool
  - Checks for program syntax and correctness
- Many suggestions, while technically correct, are also verbose
- For our class we will not worry about its suggestions



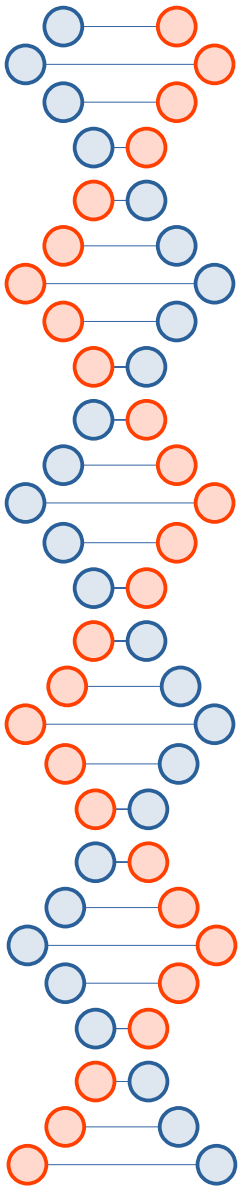
Bash strict mode

- https://olivergondza.github.io/2019/10/01/bash-strict-mode.html
- **#!/usr/bin/env bash**
- **set -euo pipefail**
- -e causes the script to exit when *any* command fails
- -u causes expansions of unassigned variables to fail
- -o pipefail causes a pipeline to fail when any command inside it fails
- This is totally optional but may help with your scripting



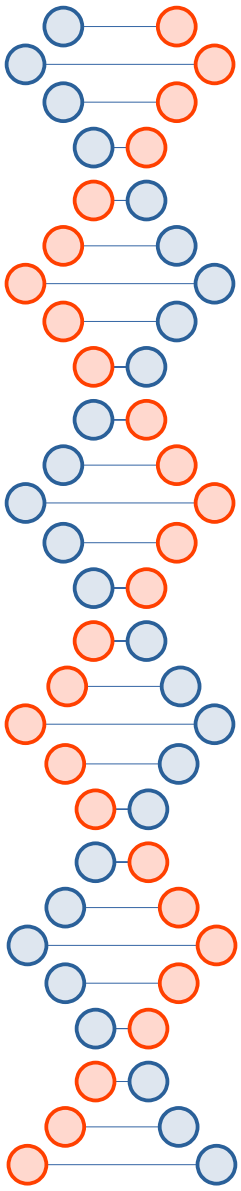
Background processes

- **do_something &**
  - Causes do_something to run in the background
- **disown**
  - Disowns do_something, it now runs freely.
- Fork bomb **(DO NOT RUN THIS!)**
- :(){ :|:& };:
- Make a function that pipes into itself in the background repeatedly and call it. **THIS WILL CRASH YOUR SYSTEM**
- https://www.cyberciti.biz/faq/understanding-bash-fork-bomb/



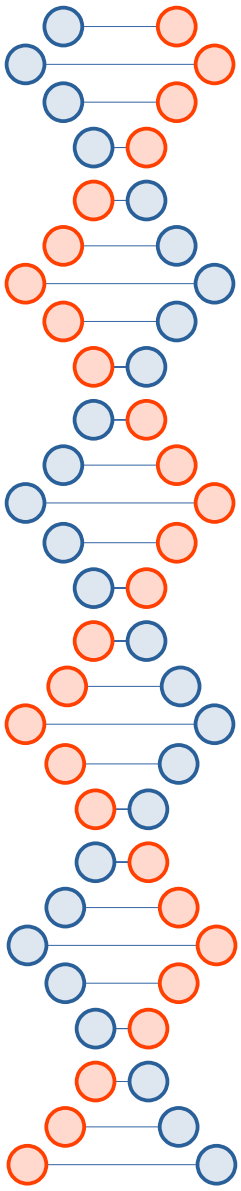
Tar

- Tar = tape archiver
  - That’s why we use -f
- **tar -xfv archive.tar.gz**
  - Extract File Verbosely
- **tar -czfv archive.tar.gz file1 file2**
  - Create (g)Zip file verbosely



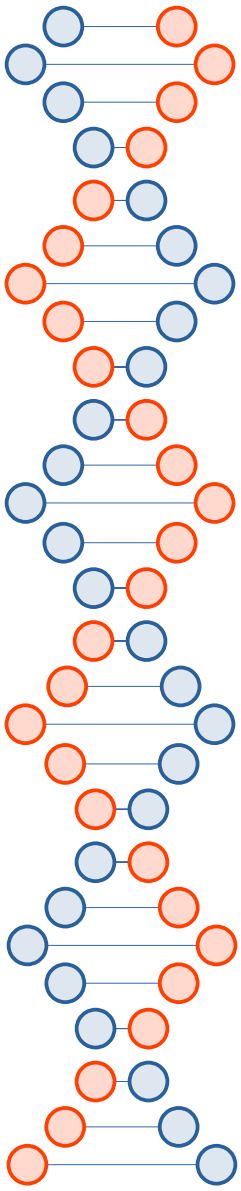
Gzip

- **gzip file.gzip file**
  - Creates file.gzip and deletes file
  - Use -k to keep the file
- **gunzip file.gzip**
  - Extract file from file.gzip and deletes file.gzip
- **zless file.tar.gzip**
  - Read a compressed archive



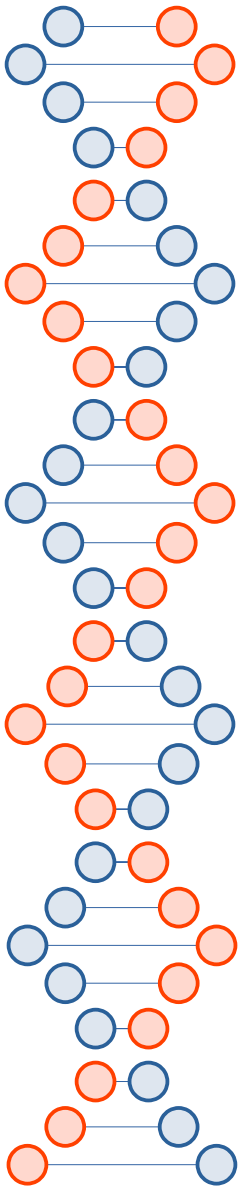
Zip vs Gzip

- tar.gz = compressed archive of files
  - More compression, no random access of files
- zip = archive of compressed files
  - Less compression, random access of files
- Same compression algorithm
  - bz2 has more compression but is slower
  - xz has even more compression and is even slower
- Network/compression tradeoff
  - What’s faster, compression/decompression or network speed?
- Don’t compress compressed files!
- Zip bombs = files that explode when uncompressed



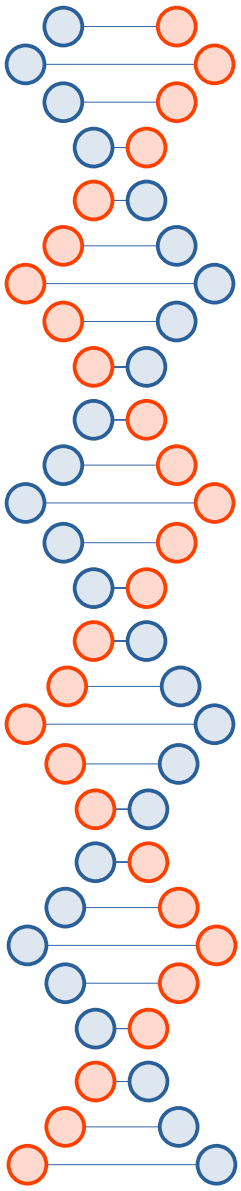
Forbidden keywords

- Avoid these when possible
- **eval**
  - Expands quotes twice, among other unpredictable behaviors
- **expr**
  - Redundant
- **curl | bash**
  - At least read the script first!
- **ls | anything, really**
  - The output of ls is not safe for scripting



Package Management

- Example: Conda
- Repository = place to share (distribute) software
- Conda is run through the terminal, so Bash knowledge is helpful
- Uses hardlinks internally to save space
- Handles dependency resolution
  - Program A depends on Program B, so installing A will also install B



The End!

Today’s Terminal Toy:
notcurses-demo