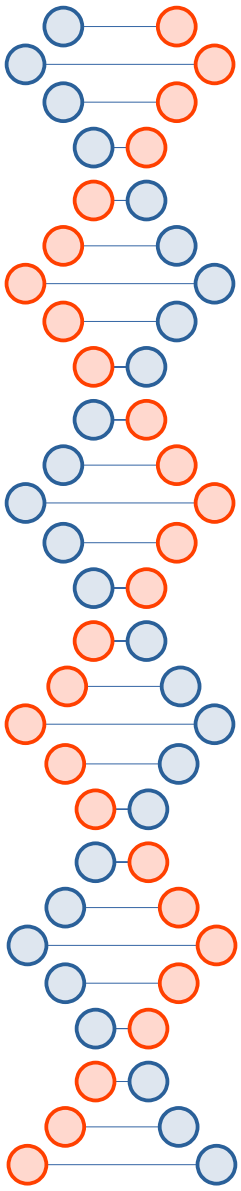


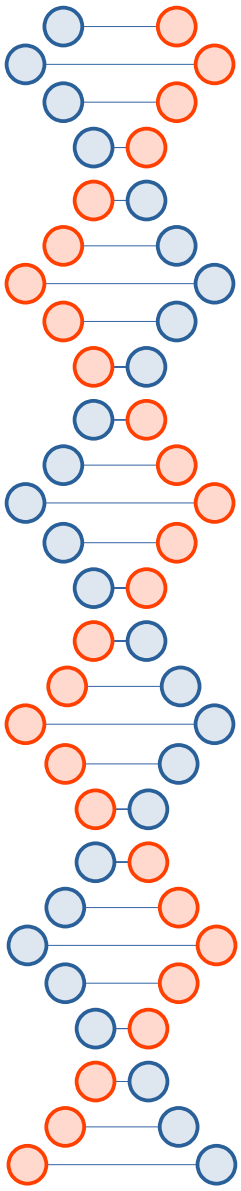
Introduction to Shell Scripting
Error Redirection, Scripts, Regex

Skylar Chan - BSCI238G
UMD
February 17th, 2023

Students will be able to

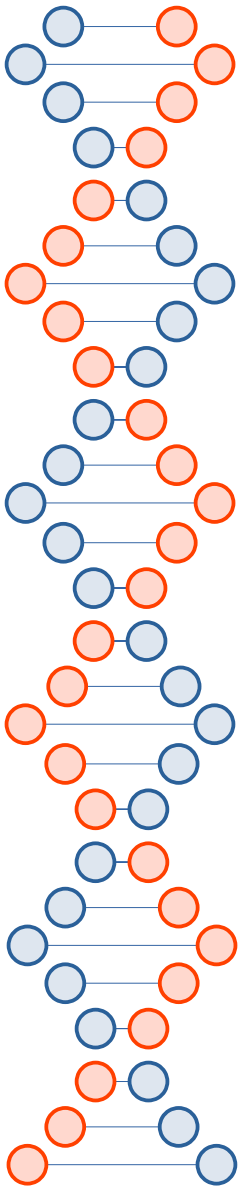
- Redirect standard error
- Write a basic script
- Understand regex expressions

Swoobat



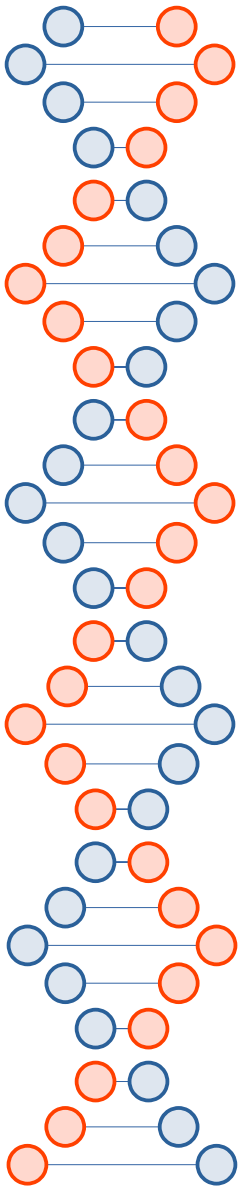
Redirecting standard error

- Recall that standard error is file descriptor 2
- To redirect standard error, redirect file descriptor 2
- **some_command 2> /dev/null**
  - Hides standard error
- **some_command > /dev/null 2>&1**
  - Hide standard error and standard output
- **some_command &> /dev/null**
  - Same as above (Bashism!)



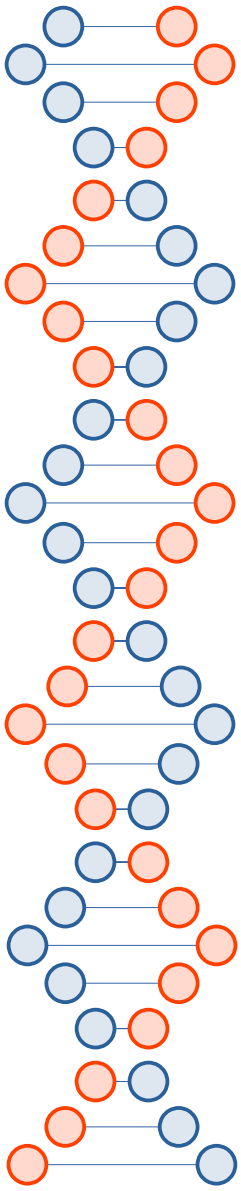
Introduction to the Script

- Text files that can be run (interpreted) by the operating system
- The first line is the shebang, which is the command to run the script with
- Each line after the shebang is run as if it was typed interactively (for the most part)



Example shebangs

- #!/bin/bash
- **#!/usr/bin/env bash**
  - Posix compliant
- #!/usr/bin/sed
- #!/usr/bin/awk



Your first Bash script

```
#!/usr/bin/env bash
pwd
ls
cd ~
pwd
ls
cd /
pwd
ls
```

Slides slightly modified

CMSC 330: Organization of Programming Languages

Regular Expressions

What if we want to find more complicated patterns?

- E.g.,
  - Either Steve, Stephen, Steven, Stefan, or Esteve
  - All words that have even number vowels

We need Regular Expressions

Regular Expressions

- A regular expression is a pattern that describes a set of strings. It is useful for
  - Searching and matching
  - Formally describing strings

    The symbols (lexemes or tokens) that make up a language
- Common to lots of languages and tools
  - Syntax for them in **sed**, **grep**, **awk**, Perl, Python, Ruby, …

    Popularized (and made fast) as a language feature in Perl
- Based on some elegant theory

Example Regular Expressions

- Ruby
  - Strings are matched exactly; here, the string "Ruby"
- Ruby|OCaml
  - *e1*|*e2* means to match either *e1* or *e2*
  - Here, matches either "Ruby" or "OCaml"
- (ab)*
  - 0 or more occurrences of “ab”: matches “”, “ab”, “abab”,”ababab”,…

Repetition in Regular Expressions

The following are suffixes on a regular expression *e*

*e** *zero or more occurrences of e*

e+ *one or more occurrences of e*

so *e*+ is the same as *ee**

a* “”, “a”, “aa”, “aaa”, …

a+ “a”, “aa”, “aaa”, …

bc* “b”, “bc”, “bcc”, …

a+b* “a”, “ab”, “aa”, “aab”, “aabb”, “aabbb”, “aaa”, …

Repetition in Regular Expressions

The following are suffixes on a regular expression *e*

*e** *zero or more occurrences of e*

e+ *one or more occurrences of e*

so *e*+ is the same as *ee**

e? *exactly zero or one e*

e{*x*} *exactly x occurrences of e*

e{*x*,} *at least x occurrences of e*

e{*x*,*y*} *at least x and at most y occurrences of e*

Watch Out for Precedence

- (Ruby)* means {"", "Ruby", "RubyRuby", ...}
- Ruby* means {"Rub", "Ruby", "Rubyy", ...}
- Best to use parentheses to disambiguate
  - Note that parentheses have another use, to extract matches, as we'll see later

Character Classes

- [abcd]
  - {"a", "b", "c", "d"} (Can you write this another way?)
- [a-zA-Z0-9]
  - Any upper- or lower-case letter or digit (range of ASCII characters)
- [^0-9]
  - Any character except 0-9 (the ^ means *not,* and must come first)
- [\t\n]
  - Tab, newline or space
- [a-zA-Z_\$][a-zA-Z_\$0-9]*
  - Java identifiers ($ escaped...see next slide)

Special Characters

.	any character
^	beginning of line
$	end of line
\$	just a $
\d	digit, [0-9]
\s	whitespace, [\t\r\n\f]
\w	word character, [A-Za-z0-9_]
\D	non-digit, [^0-9]
\S	non-space, [^\t\r\n\f]
\W	non-word, [^A-Za-z0-9_]
\\	literal backslash
\(	literal open parenthesis

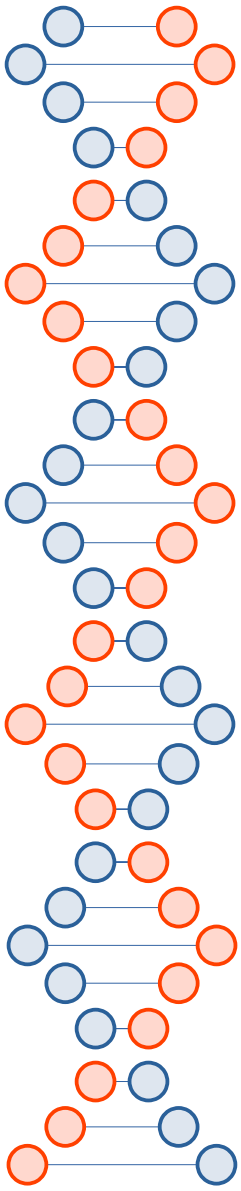
Using ^pattern$ ensures entire string/line must match pattern

Potential Syntax Confusions

- [ ]
  - Inside regular expressions: character class
  - Outside regular expressions: test expression (in Bash)
- ^
  - Inside regex character class: *not*
  - Outside regex character class: beginning of line
- ( )
  - Inside character classes: literal characters ( )
  - Outside character classes in regex: used for grouping
- –
  - Inside regex character classes: range (e.g., a to z given by [a-z])
  - Outside regex character classes: literal -

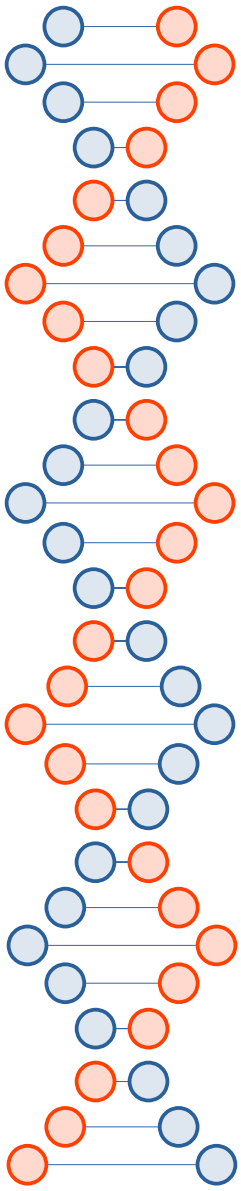
Summary

- Let *re* represents an arbitrary pattern; then:
  - *re* – matches regexp *re*
  - (*re* $_1$|*re* $_2$) – match either *re* $_1$ or *re* $_2$
  - (*re*)* – match 0 or more occurrences of *re*
  - (*re*)+ – match 1 or more occurrences of *re*
  - (*re*)? – match 0 or 1 occurrences of *re*
  - (*re*){2} – match exactly two occurrences of *re*
  - [a-z] – same as (a|b|c|...|z)
  - [^0-9] – match any character that is not 0, 1, etc.
  - ^, $ – match start or end of string



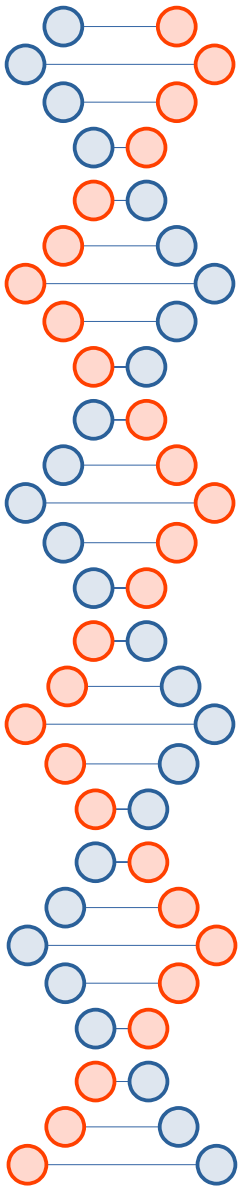
Example regexs

- ACTG
- [ACTG]
- A*C*T*G*
- [ACUG]+
- (G|C)+



Play with Regex

- Ascii table (link)
- regexer.com



The End

Today’s Terminal Toy:
asciiquarium