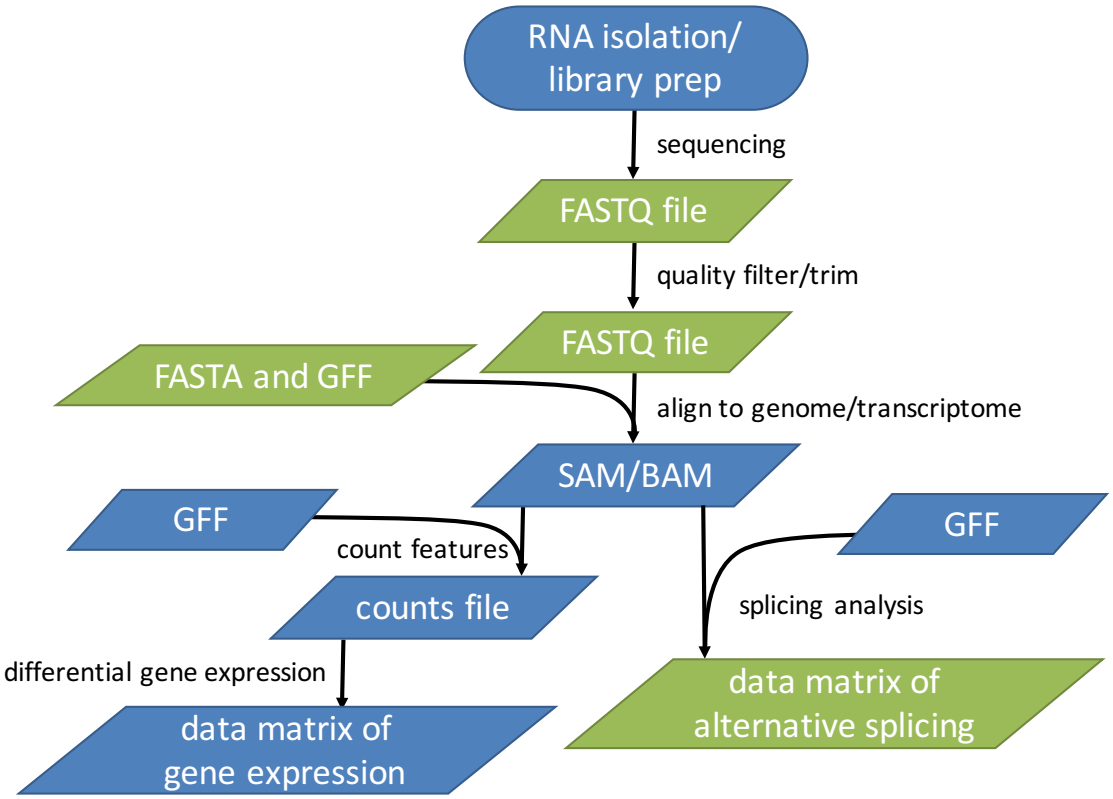
Intro to Unix and the Command Line, Part 2

File management programs, wc, cut, grep, pipes

Copy example files

- Download the following files from Canvas if you don't already have them
  - FASTA: `contigs.fa`
  - FASTQ: `seq_2.fastq`
  - `wt_vs_DM1.miso_bf_ens`
- Move them to `~/shell` (on your own computer) and rename `contigs.fa` to `contigs.fasta`



Working with text files: Count raw reads

- Consider the following file formats. What is your strategy for counting the number of reads they contain?
  - FASTA
  - FASTQ

```
>NODE_1140748_length_208_cov_4.298077
GTATATTAGAAGGGGCCGCGCGGATGAGATGGGTGACAGTACACTTTCCATGCAAGAACG
GGCGGGTTTGTAATATTCCTTAAATTATTGTCAGAAACTCTGTGATGGAGACATTGACCT
```

```
@NS500451:154:HWKTMBGXX:1:11101:10065:1121 1:N:0:TAGAACAC
AGGTTGCTATGAAATTTTAGTTGTCGTAGTAGGCAAACAATAAGGAATGTTGATCCAATAATTACATGGAGTCCATGGAA
+
AAAAAEEEA6EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEAEEEEEE
@NS500451:154:HWKTMBGXX:1:11101:8541:1044 1:N:0:TAGAACAC
CCAAGNTTTCCGCCAGCTCCATCAGCGCTCGTGCTCTGGCATCCCGGCTACACCTCCATTTCTCTCAACTGTNACGNNNN
+
AAAAA#EE/6AEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE#EEE####
@NS500451:154:HWKTMBGXX:1:11101:8274:1044 1:N:0:TAGAACAC
CAGTANTGTCTCGTATTCATCCTTGCAAATAATGGTTCCTGTGCTCTCTGATGGAGCACTAATAGCACCCGCTGTANNAN
+
6AAAA#EE/6AEEEEEE6EEEEEE/AEEEEEEEEAEAEEEEEA<EEEEEEEEE/EEEEEEEEEEEEA6EAEEEEEE##E#
```

Working with text files: Count raw reads

```
$ cd ~/shell
$ wc -l seq_2.fastq  #will this give the actual # of reads?
4000000 seq_2.fastq

$ grep “@” seq_2.fastq    #prints every line containing “@”
                          #ctrl+c will abort
$ grep -c “@” seq_2.fastq
1000000
$ grep -c “^@” seq_2.fastq
1000000
$ grep -v “@” seq_2.fastq
$ grep -v -c “@” seq_2.fastq
3000000

#count reads with in-line barcode
$ grep -c “^CGATA” s_1_seq.fastq
19501
#what is the difference between this command and
$ grep -c “CGATA” s_1_seq.fastq
```

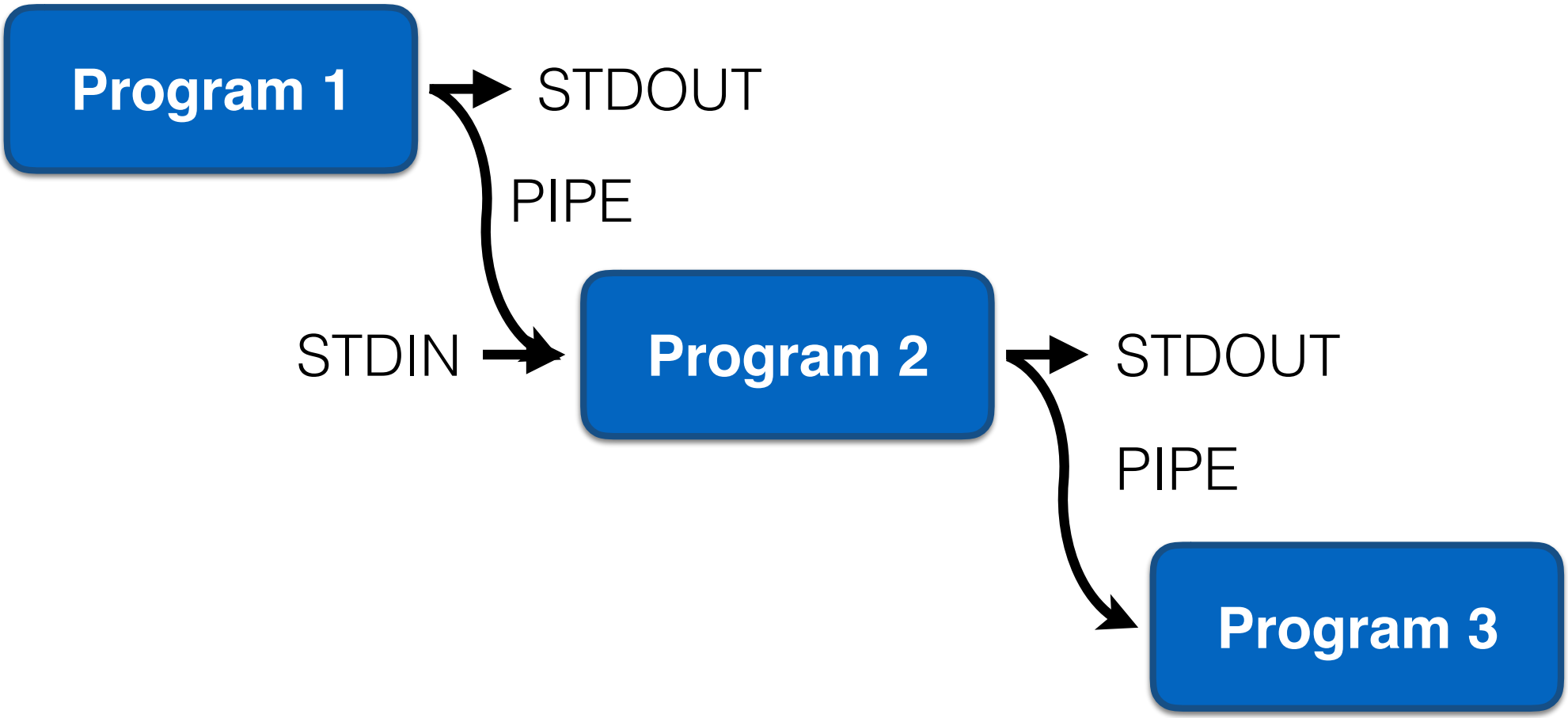
Special files: STDIN, STDOUT, STDERR

- STDIN
  - The input stream going into a program
  - Ex: `wc`
  - End-of-transmission character: `ctrl+d`
  - `wc -l < ~/contigs.fasta`
- STDOUT
  - The stream where a program writes it's output data
  - STDOUT is the terminal, unless redirected
  - `wc -l ~/contigs.fasta > wc.txt`
- STDERR
  - Stream containing error messages/diagnostics
  - Also the terminal, unless redirected
  - `wc -l ~/nonexistant_file.txt 2> errors.txt`

PIPES – the shell's killer app

Pipes

Pipes



What is the purpose of the program `cat`?

Some examples

```
#cut
$ cut -f 2

#cut, capture the output
$ cut -f 2,5,19-21 wt_vs_DM1.miso_bf_ens > events.miso

#cut, pipe the output to grep
$ cut -f 2,5,19-21 wt_vs_DM1.miso_bf_ens | grep “Mbnl”
$ cut -f 2,5,19-21 wt_vs_DM1.miso_bf_ens | grep “Mbnl” | less -S
$ cut -f 2,5,19-21 wt_vs_DM1.miso_bf_ens | grep “Mbnl” >
Mbnl.txt
```

ICA2 – Parsing a text file

`ls`
`man`
`more`
`less`
`cat`
`wc`
`head`
`cut`
`grep`
`sort`
`uniq`
`>`
`|`

`s_1_seq.fastq`

1. Count the number of raw reads (250,000)
2. Count the number of reads with barcode CGATA (19,501)
3. Capture all FASTQ records for the ACCAT barcode into a file called `sample_1.fq` (you should get 18,352 records and 73,408 lines)
4. Determine the count of all barcodes in the file

Some hints:

1. Use `head` when building a command, `cat` once the command is working
2. Look at the `-n` option for the head command, the `-l` option for `wc`
3. Checkout the `grep` options: `-c`, `-v`, `-A`, `-B`
4. Read the `man` pages for `sort` and `uniq` to learn how to use/combine them

PS1 – Parsing an output file from MISO