

Lecture 2B:

Strings, redirects, and pipes



Lecture 2B outline

1. Strings
2. More commands
3. Redirects and pipes
4. Lab 2B overview

Unix design principle #2

// Expect the output of every program to become the input to another, as yet unknown, program.

How to choose a format that can be written by any program, read by any program, and potentially read and written by any user?

Strings

A ***string*** is a sequence of ***standard text characters*** that can be interpreted (or read) by humans

Binary string

- 010010010101

Hexidecimal string

- bf2741f09ce03ede19231

Text string

- "Hello, world!"
- 'top_secret_password.txt'

Text strings

Text strings can include any standard character (*e.g. letters, numbers, symbols, spaces, enters*)

A string is **constructed** as a sequence of characters that is **delimited** by a matching pair of single quotes or double quotes

```
# valid single-quote construction
'Hello, world!'
# valid double-quote construction
"Hello, world!"
# not valid, due to mismatched quotes
'Hello, world!"
```

Escaped characters

Certain characters have special meanings, such as the string delimiter tokens -- ' and "

Special characters can be ***escaped*** when preceded by the backslash (\)

When constructing a string, an escaped character will always print its apparent (or ***literal***) value rather than convey its special meaning

Escaped strings

string construction	string literal
"my friend"	my friend
"my 'friend'"	my 'friend'
"my "friend""	<i>equivalent to 'my ';</i> then problem
"my \"friend\""	my "friend"

single-quote strings automatically escapes all characters;
double-quote strings require manual escapes

Special characters we'll use

escaped character	string literal	special meaning
\"	"	string delimiter
\'	'	string delimiter
\\$	\$	shell variable identifier
*	*	wildcard
\?	?	wildcard
\\	\	escape character
\n	<newline>	enters newline
\t	<tab>	enters tab

Typical special characters recognized across operating systems and programming languages

Wildcards

Wildcards match patterns across many strings
(useful with filesystems)

Each wildcard character in a string can match

- any *single* character to against the ? wildcard
- any *string* of characters against the * wildcard

```
> ls
cow  crab  crow
> ls cr*
crab  crow
> ls c?ow
crow
> ls cr??
crab  crow
```

More shell commands

man	display manual page
wc	word, line, character counts
head	display first lines of file
tail	display last lines of file
diff	compare files line-by-line
grep	file pattern searcher

man

display manual page

```
> man echo
```

```
ECHO(1)                                BSD General Commands Manual                                ECHO(1)
```

NAME

echo -- write arguments to the standard output

SYNOPSIS

```
echo [-n] [string ...]
```

DESCRIPTION

The echo utility writes any specified operands, separated by single blank (` `) characters and followed by a newline (` \n `) character, to the standard output.

The following option is available:

-n Do not print the trailing newline character. This may also be achieved by appending ` \c ` to the end of the string, as is done by iBCS2 compatible systems. Note that this option as well as the

arrow keys to navigate; 'q' to exit

WC

count words, lines, characters for file

```
# print file contents
> cat lyrics.txt
I am the very model of a modern Major-General,
I've information vegetable, animal, and mineral,
I know the kings of England, and I quote the fights historical
From Marathon to Waterloo, in order categorical;
I'm very well acquainted, too, with matters mathematical,
I understand equations, both the simple and quadratical,
About binomial theorem I'm teeming with a lot o' news,
With many cheerful facts about the square of the hypotenuse.
I'm very good at integral and differential calculus;
I know the scientific names of beings animalculous:
In short, in matters vegetable, animal, and mineral,
I am the very model of a modern Major-General.

# count lines, words, and characters
> wc lyrics.txt
    12     108    669 lyrics.txt
```

line
count

word
count


character
count

target
file

head


print first
10 lines
(default)

display first lines of file



```
# print first ten lines (default)
> head lyrics.txt
I am the very model of a modern Major-General,
I've information vegetable, animal, and mineral,
I know the kings of England, and I quote the fights historical
From Marathon to Waterloo, in order categorical;
I'm very well acquainted, too, with matters mathematical,
I understand equations, both the simple and quadratical,
About binomial theorem I'm teeming with a lot o' news,
With many cheerful facts about the square of the hypotenuse.
I'm very good at integral and differential calculus.
I know the scientific names of beings animalculous:

# print first two lines
> head -n2 lyrics.txt
I am the very model of a modern Major-General,
I've information vegetable, animal, and mineral,
```




print only
first 2 lines
(-n2)

tail


print last
10 lines
(default)

display last lines of file



```
# print last ten lines (default)
> tail lyrics.txt
I know the kings of England, and I quote the fights historical
From Marathon to Waterloo, in order categorical; a
I'm very well acquainted, too, with matters mathematical,
I understand equations, both the simple and quadratical,
About binomial theorem I'm teeming with a lot o' news,
With many cheerful facts about the square of the hypotenuse.
I'm very good at integral and differential calculus;
I know the scientific names of beings animalculous:
In short, in matters vegetable, animal, and mineral,
I am the very model of a modern Major-General.

# print last two lines
> tail -n2 lyrics.txt
In short, in matters vegetable, animal, and mineral,
I am the very model of a modern Major-General.
```



print only
last 2 lines
(-n2)

diff

compare files line-by-line

```
# view end of first file
> tail -n4 lyrics.txt
I'm very good at integral and differential calculus;
I know the scientific names of beings animalculous:
In short, in matters vegetable, animal, and mineral,
I am the very model of a modern Major-General.

# view end of second file
> tail -n4 or_were_these_the_lyrics.txt
I'm very good at integral and differential calculus;
I know the scientific names of beings animalculous:
In short, in matters vegetable, animal, and mineral,
I am the hairy model of a modern Major-General.

# show differences in files
> diff lyric.txt or_were_these_the_lyrics.txt
10c10
< I know the scientific names of beings animalculous:
---
> I know the scientific manes of beings animalculous:
12c12
< I am the very model of a modern Major-General.
---
> I am the hairy model of a modern Major-General.
```

difference on
line 10



difference on
line 12



grep

file pattern searcher

print lines
that contain
pattern "animal"



```
# print lines containing "animal"
> grep animal lyrics.txt
I've information vegetable, animal, and mineral,
I know the scientific names of beings animalculous:
In short, in matters vegetable, animal, and mineral,
```

print lines
that *do not* contain
pattern "animal"
(-v, inverted grep)

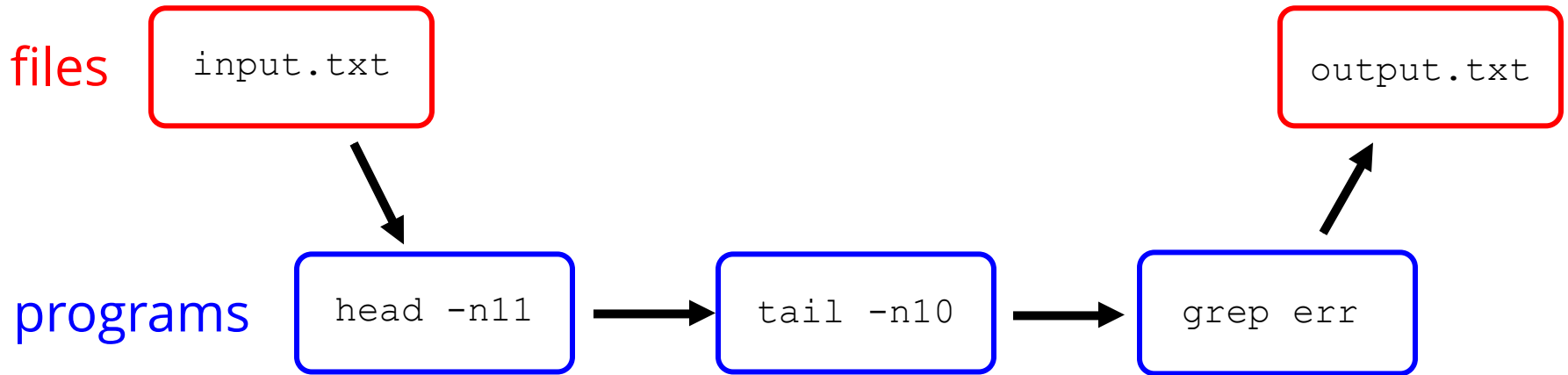


```
# print lines that do _not_ contain "animal"
> grep -v animal lyrics.txt
I am the very model of a modern Major-General,
I know the kings of England, and I quote the fights historical
From Marathon to Waterloo, in order categorical;
I'm very well acquainted, too, with matters mathematical,
I understand equations, both the simple and quadratical,
About binomial theorem I'm teeming with a lot o' news,
With many cheerful facts about the square of the hypotenuse.
I'm very good at integral and differential calculus;
I am the very model of a modern Major-General.
```

We'll learn more about this powerful tool
when we learn about regular expressions

Redirects and pipes

example of simple pipeline



Redirects transmit info between `files` and `programs`

Pipes transmit info directly between `programs`

Use > to **redirect** program output into a file

```
$ echo "Hello, world!"  
Hello, world!  
# redirect echo output into file.txt  
$ echo "Hello, world!" > file.txt  
$ ls  
file.txt  
$ cat file.txt  
Hello, world!
```

Use >> to **append** program output into a file

```
# append echo output into file.txt  
$ echo "...um, hello?" >> file.txt  
$ cat file.txt  
Hello, world!  
...um, hello?
```

Use < to **redirect** a file as input into a program

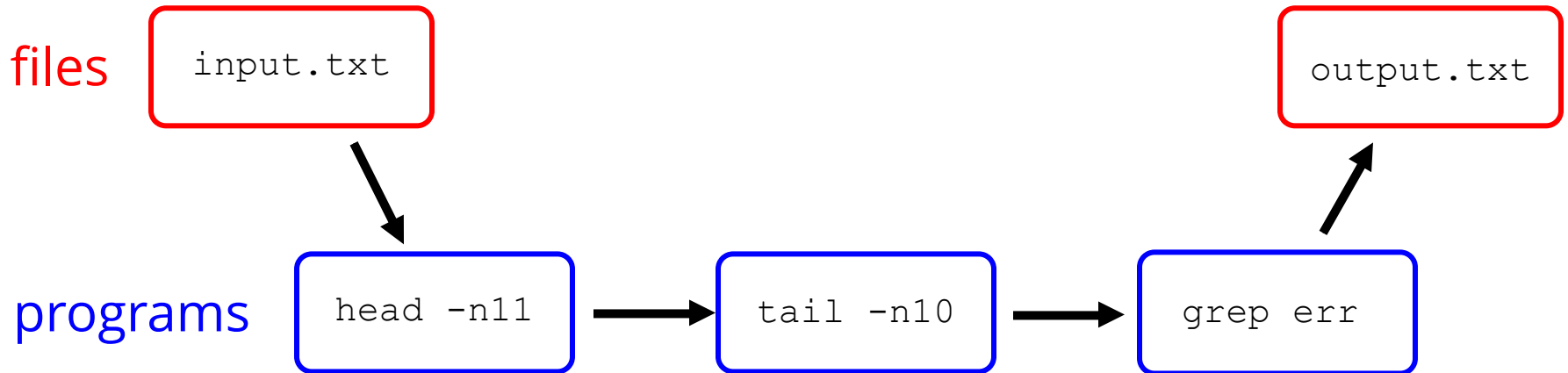
```
# redirect file.txt as input into cat
$ tail -n2 < lyrics.txt
In short, in matters vegetable, animal, and mineral,
I am the very model of a modern Major-General.
```

Use | to transmit (or **pipe**) the output of one program as the input of a second program

```
# pipe the output of the ls command
# as input for the wc command
$ ls
eenie.txt  meenie.txt  minie.txt  mo.txt
$ ls | wc
      4      4     38
$ ls *eenie.txt | wc
      2      2     21
```

Redirects and pipes

example of simple pipeline



example command

```
$ head -n11 < input.txt | tail -n10 | grep err > output.txt
```

More pipeline examples

```
# create a new file with text
$ echo "I made a file for you" > new_file.txt
$ cat new_file.txt
I made a file for you
```

```
# count how many lines contain "animal";
# print that count to the file "num_animal.txt"
$ grep animal lyrics.txt | wc -l > num_animal.txt
$ cat num_animal.txt
3
```

```
# quickly scan man page for option to number lines
$ man cat | grep " -" | grep Number
-b      Number the non-blank output lines, starting at 1.
-n      Number the output lines, starting at 1.
```

More pipeline examples

```
# redirect file.txt as input into tail;  
# then redirect out from cat into lyrics_tail.txt  
$ tail -n2 < lyrics.txt > lyrics_tail.txt  
$ cat lyrics_tail.txt  
In short, in matters vegetable, animal, and mineral,  
I am the very model of a modern Major-General.
```

```
# Suppose you fit a model to the same dataset twice;  
# this is often done to ensure that the inference  
# method succeeded to find the best estimate. Find  
# instances where the two output files contain  
# _different_ values for parameter x2, but at least  
# one model-fitting method claims success.  
$ cat output1.txt  
10.321,x1,failure  
36.331,x2,success  
91.585,x3,success  
$ cat output2.txt  
10.321,x1,failure  
35.268,x2,failure  
96.521,x3,success  
# find differences in x2 where an entry claims success  
$ diff output1.txt output2.txt | grep x2 | grep success  
< 36.331,x2,success
```

Lab 2B

github.com/WUSTL-Biol4220/home/labs/lab_02B.md