

Part 1.3:

Introduction to Coreutils

Manipulating text data on command line

Dana L Carper and Travis J Lawrence

Quantitative and Systems Biology

University of California, Merced



What are Coreutils?

- The basic file, shell and text manipulation utilities of the GNU/Linux operating system.
- Core utilities which are expected to exist on every operating system

How does a Computer see a file?

- Example File example.1.3.txt:

How does a computer process this file?
Another line of text that is very exciting.

How does a Computer see a file?

- Example File `example.1.3.txt`:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Word Count command: `wc`

```
$ wc example.1.3.txt  
2      15     83  example.1.3.txt
```

How does a Computer see a file?

- Example File example.1.3.txt:

Line 1 → How does a computer process this file?\nLine 2 → Another line of text that is very exciting.\n

- Word Count command: wc

```
$ wc example.1.3.txt
    2      15     83  example.1.3.txt
```

↑
Number of lines

How does a Computer see a file?

- Example File example.1.3.txt:

Line 1 → How does a computer process this file?\n

Line 2 → Another line of text that is very exciting.\n

- Word Count command: wc

```
$ wc example.1.3.txt
```

2

Number of lines

```
example.1.3.txt
```

What is the structure of a line?

- Ends with a newline character ("\n")
 - "\n" – newline character
 - Indicates end of line
 - Invisible
 - Different on Windows

How does a Computer see a file?

- Example File example.1.3.txt:

Line 1 → How|does|a|computer|process|this|file?\nLine 2 → Another|line|of|text|that|is|very|exciting.\n

- Word Count command: `wc`

```
$ wc example.1.3.txt  
2 15 83 example.1.3.txt
```

Number of lines

Number of words
(fields)

How does a Computer see a file?

- Example File example.1.3.txt:

Line 1 → How|does|a|computer|process|this|file?\nLine 2 → Another|line|of|text|that|is|very|exciting.\n

- Word Count command: `wc`

```
$ wc example.1.3.txt
      2      15     83     example.1.3.txt
```

Number of lines Number of words
(fields)

What separates (delimits) fields?

- Whitespace (default)
 - spaces and tabs
- Optionally set to any character
 - Commas (csv)
 - Tabs (tsv)

How does a Computer see a file?

- Example File example.1.3.txt:

Line 1 → How|does|a|computer|process|this|file?\nLine 2 → Another|line|of|text|that|is|very|exciting.\n

- Word Count command: `wc`

```
$ wc example.1.3.txt
```

2 15 83 example.1.3.txt

Number of lines Number of words Number of characters
(fields)

head

- Example File `example.1.3.txt`:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- # of lines from beginning of file or input: `head`

```
$ head example.1.3.txt  
How does a computer process this file?  
Another line of text that is very exciting.
```

head

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- # of lines from beginning of file or input: head

```
$ head example.1.3.txt  
How does a computer process this file?  
Another line of text that is very exciting.
```

- Default number of lines is 10. Can we change this?

```
$ man head
```

man head

HEAD(1)

BSD General Commands Manual

HEAD(1)

NAME

head -- display first lines of a file

SYNOPSIS

head [-n count | -c bytes] [file ...]

DESCRIPTION

This filter displays the first count lines or bytes of each of the specified files, or of the standard input if no files are specified. If count is omitted it defaults to 10.

If more than a single file is specified, each file is preceded by a header consisting of the string ``==> XXX <=='' where ``XXX'' is the name of the file.

EXIT STATUS

The **head** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

tail(1)

HISTORY

The **head** command appeared in PWB UNIX.

man head

HEAD(1)

BSD General Commands Manual

HEAD(1)

NAME

head -- display first lines of a file

SYNOPSIS

head [-n count | -c bytes] [file ...]

DESCRIPTION

This filter displays the first count lines or bytes of each of the specified files, or of the standard input if no files are specified. If count is omitted it defaults to 10.

If more than a single file is specified, each file is preceded by a header consisting of the string ``==> XXX <=='' where ``XXX'' is the name of the file.

EXIT STATUS

The **head** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

tail(1)

HISTORY

The **head** command appeared in PWB UNIX.



Provides name of command and short description

man head

HEAD(1)

BSD General Commands Manual

HEAD(1)

NAME

head -- display first lines of a file

SYNOPSIS

head [-n count | -c bytes] [file ...]

DESCRIPTION

This filter displays the first count lines or bytes of each of the specified files, or of the standard input if no files are specified. If count is omitted it defaults to 10.

If more than a single file is specified, each file is preceded by a header consisting of the string ``==> XXX <=='' where ``XXX'' is the name of the file.

EXIT STATUS

The **head** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

tail(1)

HISTORY

The **head** command appeared in PWB UNIX.



Provides concise overview of possible options

man head

HEAD(1)

BSD General Commands Manual

HEAD(1)

NAME

head -- display first lines of a file

SYNOPSIS

head [-n count | -c bytes] [file ...]

DESCRIPTION

This filter displays the first count lines or bytes of each of the specified files, or of the standard input if no files are specified. If count is omitted it defaults to 10.

If more than a single file is specified, each file is preceded by a header consisting of the string ``==> XXX <=='' where ``XXX'' is the name of the file.

EXIT STATUS

The **head** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

tail(1)

HISTORY

The **head** command appeared in PWB UNIX.



Provides long description

man head

HEAD(1)

BSD General Commands Manual

HEAD(1)

NAME

head -- display first lines of a file

SYNOPSIS

head [-n count | -c bytes] [file ...]

DESCRIPTION

This filter displays the first count lines or bytes of each of the specified files, or of the standard input if no files are specified. If count is omitted it defaults to 10.

If more than a single file is specified, each file is preceded by a header consisting of the string ``==> XXX <=='' where ``XXX'' is the name of the file.

EXIT STATUS

The **head** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

tail(1)

HISTORY

The **head** command appeared in PWB UNIX.

The synopsis and description indicate the -n option controls how many lines are displayed

BSD

June 6, 1993

BSD

head

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- # of lines from beginning of file or input: head

```
$ head example.1.3.txt  
How does a computer process this file?  
Another line of text that is very exciting.
```

- Default number of lines is 10. Can we change this?

```
$ head -n 1 example.1.3.txt  
How does a computer process this file?
```



-n 1

Is an argument saying how many lines we want

tail

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- # of lines from end of file or input: tail

```
$ tail example.1.3.txt  
How does a computer process this file?  
Another line of text that is very exciting.
```

tail

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- # of lines from beginning of file or input: tail

```
$ tail example.1.3.txt  
How does a computer process this file?  
Another line of text that is very exciting.
```

- Default number of lines is 10. Can we change this?

```
tail -n 1 example.1.3.txt  
Another line of text that is very exciting.
```

cut – single character

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c 1 example.1.3.txt
```

cut – single character

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c 1 example.1.3.txt  
H  
A
```

cut – multiple characters

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c 1 example.1.3.txt  
H  
A
```

```
$ cut -c 1,3 example.1.3.txt  
Hw  
Ao
```

cut – ranges

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c -3,5,7-9,12- example.1.3.txt
```

cut – ranges

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c 3,5,7-9,12- example.1.3.txt  
Howdes computer process this file?  
Anohr le of text that is very exciting.
```

cut – ranges

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c -3,5,7-9,12- example.1.3.txt
```

```
Howdes computer process this file?  
Anohr le of text that is very exciting.
```

Open Range:

Beginning of line to character 3



```
-3
```

cut – ranges

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c -3,5,7-9,12- example.1.3.txt  
Howdes computer process this file?  
Anohr le of text that is very exciting.
```

Single Character:
character 5

cut – ranges

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c -3,5,7-9,12- example.1.3.txt  
Howdes computer process this file?  
Anohr le of text that is very exciting.
```



Closed Range:
character 7 through 9

cut – ranges

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select characters from each line of file or input: `cut -c`

```
$ cut -c -3,5,7-9,12- example.1.3.txt  
Howdes computer process this file?  
Anohr le of text that is very exciting.
```



Open Range:
character 12 until end of line

cut – ranges

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select fields from each line of file or input: `cut -f`

```
$ cut -f -3,5,6- example.1.3.txt
```

cut - fields

- Example File `example.1.3.txt`:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select fields from each line of file or input: `cut -f`

```
$ cut -f -3,5,6- example.1.3.txt  
How does a computer process this file?  
Another line of text that is very exciting.
```

- This didn't work. What is the default delimiter?: `tab`
 - How can we change it? `man cut`

cut - fields

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Select fields from each line of file or input: `cut -f`

```
$ cut -f -3,5,6- example.1.3.txt  
How does a computer process this file?  
Another line of text that is very exciting.
```

- This didn't work. What is the default delimiter?: `tab`

- How can we change it? `man cut`

```
$ cut -d " " -f -3,5,7- example.1.3.txt  
How does a process file?  
Another line of that very exciting.
```

grep

- Example File `example.1.3.txt`:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Returns lines that match the pattern

```
$ grep "text" example.1.3.txt  
Another line of text that is very exciting.
```

```
$ grep "er" example.1.3.txt  
How does a computerer process this file?  
Anotherer line of text that is very exciting.
```

Complex Pattern Matching – Regular Expressions

- Patterns are matched on lines
- Special characters for matching more complex patterns:
 - . – match any character
 - * – zero or more of previous character
 - [] – will match any character in the brackets
 - ^ – beginning of line
 - \$ – end of line
 - \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

- Match - abc123xyz
- Match - "123"
- Match - var g = 123;

Special characters

- . - match any character
- * - zero or more of previous character
- [] - will match any character in the brackets
- ^ - beginning of line
- \$ - end of line
- \ - removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

Pattern – 123

- Match – abc123xyz
- Match – "123"
- Match – var g = 123;

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

- Match - cat.
- Match - 896.
- Match - ?=+.
- Skip - abc1

Special characters

- . - match any character
- * - zero or more of previous character
- [] - will match any character in the brackets
- ^ - beginning of line
- \$ - end of line
- \ - removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

Pattern - \.

- Match - cat.
- Match - 896.
- Match - ?=+.
- Skip - abc1

Special characters

- . - match any character
- * - zero or more of previous character
- [] - will match any character in the brackets
- ^ - beginning of line
- \$ - end of line
- \ - removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

- Match – Pig
- Match – Peg
- Match – Pug
- Skip – pog

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

Pattern – P.g

- Match – Pig
- Match – Peg
- Match – Pug
- Skip – pog

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

- Match – Piig
- Match – Piiig
- Match – Piiiig
- Skip – Pig

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

Pattern – Pii i *

- Match – Piig
- Match – Piiig
- Match – Piiig
- Skip – Pig

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

- Match – can
- Match – man
- Match – fan
- Skip – dan
- Skip – ran
- Skip – pan

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

Pattern – [fcm]an

- Match – can
- Match – man
- Match – fan
- Skip – dan
- Skip – ran
- Skip – pan

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

- Match – Run Complete
- Skip – Last Run Failed to Complete
- Skip – Next Run in Progress

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

Complex Pattern Matching (grep) - Examples

Pattern – ^Run

- Match – Run Complete
- Skip – Last Run Failed to Complete
- Skip – Next Run in Progress

Special characters

- . – match any character
- * – zero or more of previous character
- [] – will match any character in the brackets
- ^ – beginning of line
- \$ – end of line
- \ – removes the "specialness" of a character

tr

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Used to translate, delete, and squeeze multiple occurrences of characters

```
$ cat example.1.3.txt|tr  
How does a computer process this file?  
Another line of text that is very exciting.
```

tr – default behavior

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Used to translate, delete, and squeeze multiple occurrences of characters

```
$ cat example.1.3.txt|tr "rt" "dg"  
How does a compuged process ghis file?  
Anoghed line of gexg ghag is vedy exciging.
```

tr - delete

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Used to translate, delete, and squeeze multiple occurrences of characters

```
$ cat example.1.3.txt|tr -d " "  
Howdoesacomputerprocesssthisfile?  
Anotherlineoftextthatisveryexciting.
```

tr - squeeze

- Example File example.1.3.txt:

```
How does a computer process this file?  
Another line of text that is very exciting.
```

- Used to translate, delete, and squeeze multiple occurrences of characters

```
$ cat example.1.3.txt| tr -s "s"  
How does a computer proces this file?  
Another line of text that is very exciting.
```

sort

- Provides several options to sort lines of text
- Options
 - -r – reverse sort
 - -n – numerical sort
 - -k – sort by selected column
- More practice in the worksheet

uniq

- Reports or filters out consecutively repeated lines in a file. Typically used after sorting.
- Example 1

How does a computer process this file?

- Another line of text that is very exciting.
- Another line of text that is very exciting.

repeated

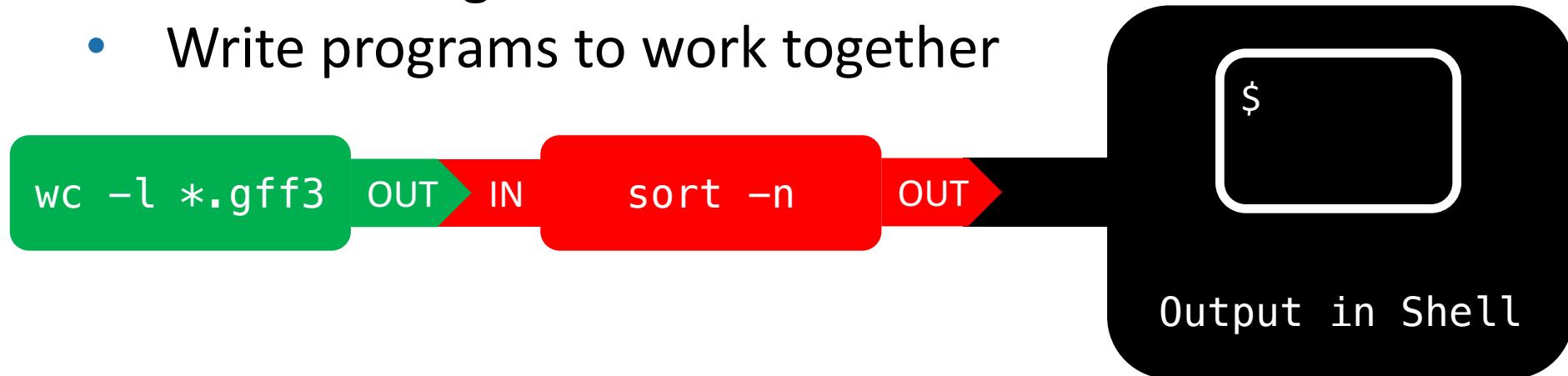
- Example 2

- Another line of text that is very exciting.
- How does a computer process this file?
- Another line of text that is very exciting.

not repeated

Combining Commands

- Unix Philosophy:
 - Do one thing and do it well
 - Write programs to work together

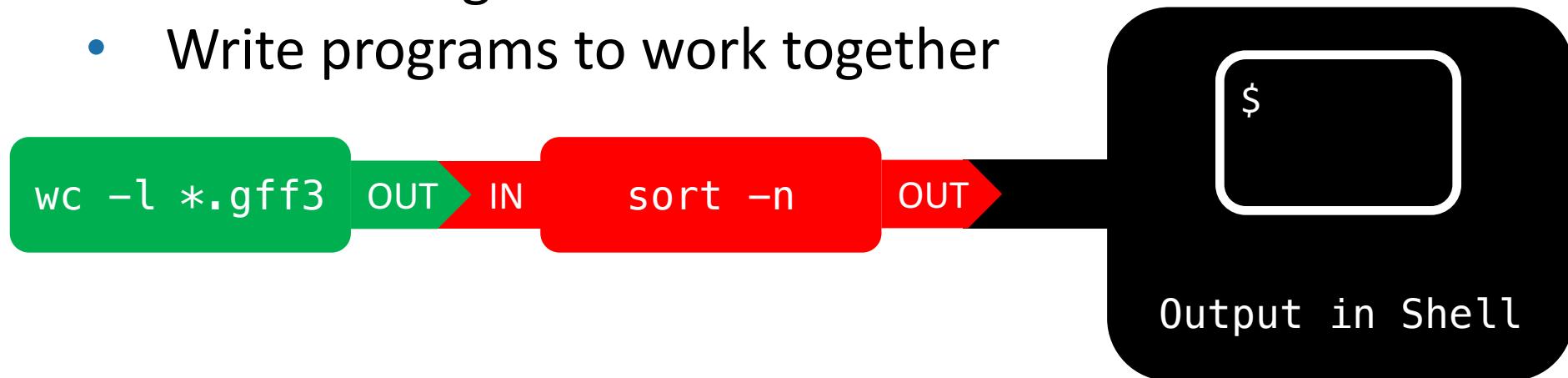


- `$ wc -l *.gff3 | sort -n`

The pipe character “|” allows you to use the output of one command as the input of another command

Combining Commands

- Unix Philosophy:
 - Do one thing and do it well
 - Write programs to work together



- `$ wc -l *.gff3 | sort -n`



- ? – Match any single character
- * – Match any number of characters
- [] – Specifies a range to match
- [!] – Specifies a range to not match