Lab 2

INTRODUCTION TO AWK

What can you do with awk?

awk operation:

- scans a file line by line
- splits each input line into fields
- compares input line/fields to pattern
- performs action(s) on matched lines

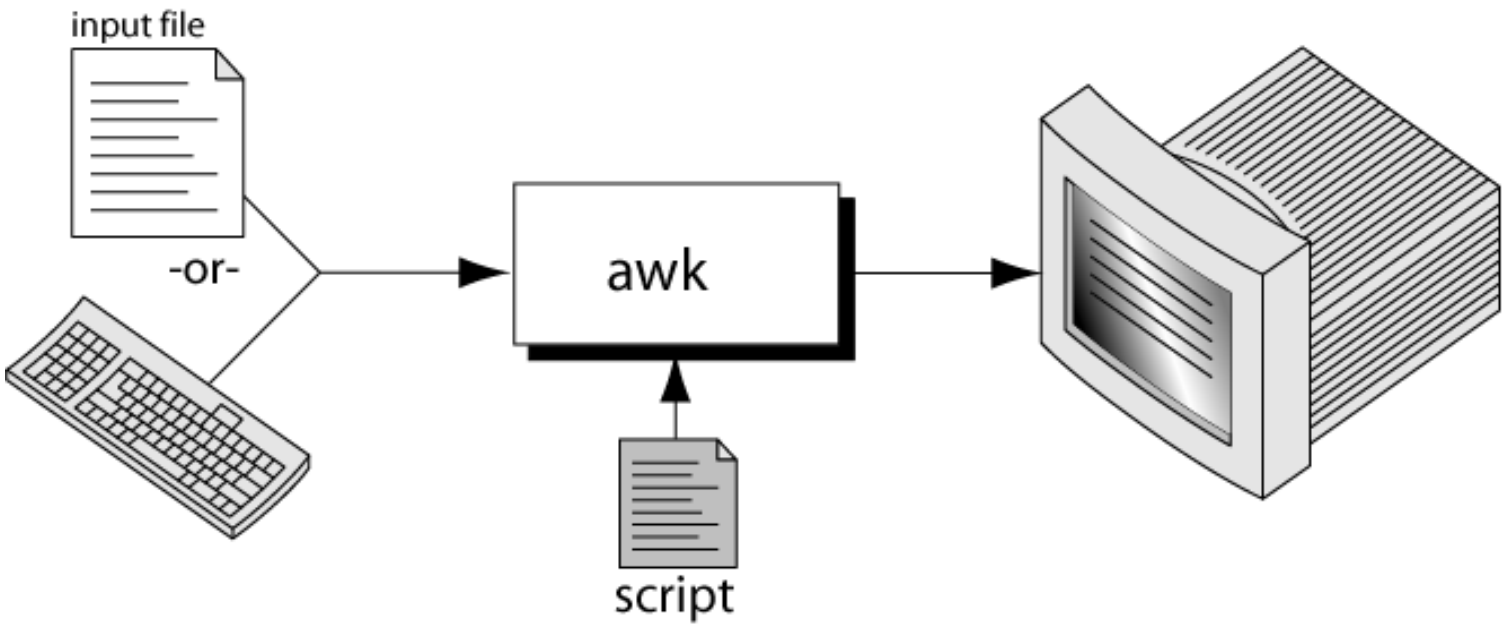
Useful for:

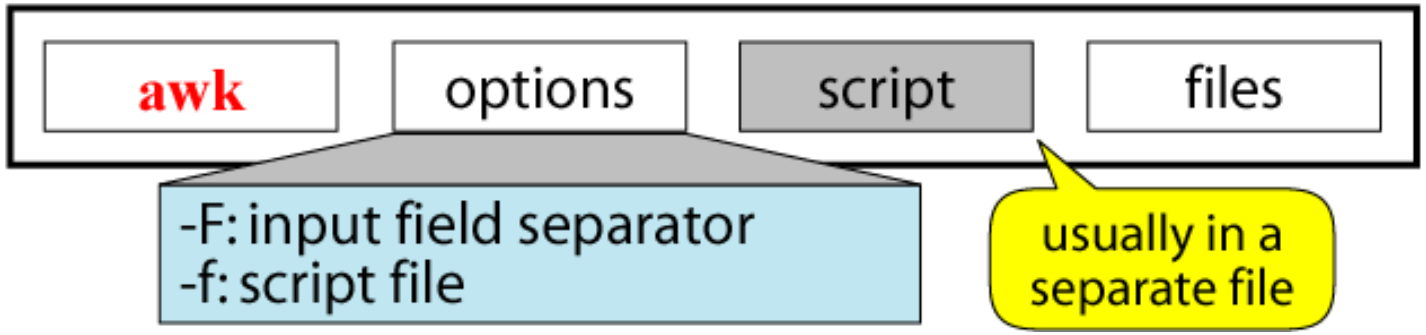
- transform data files
- produce formatted reports

Programming constructs:

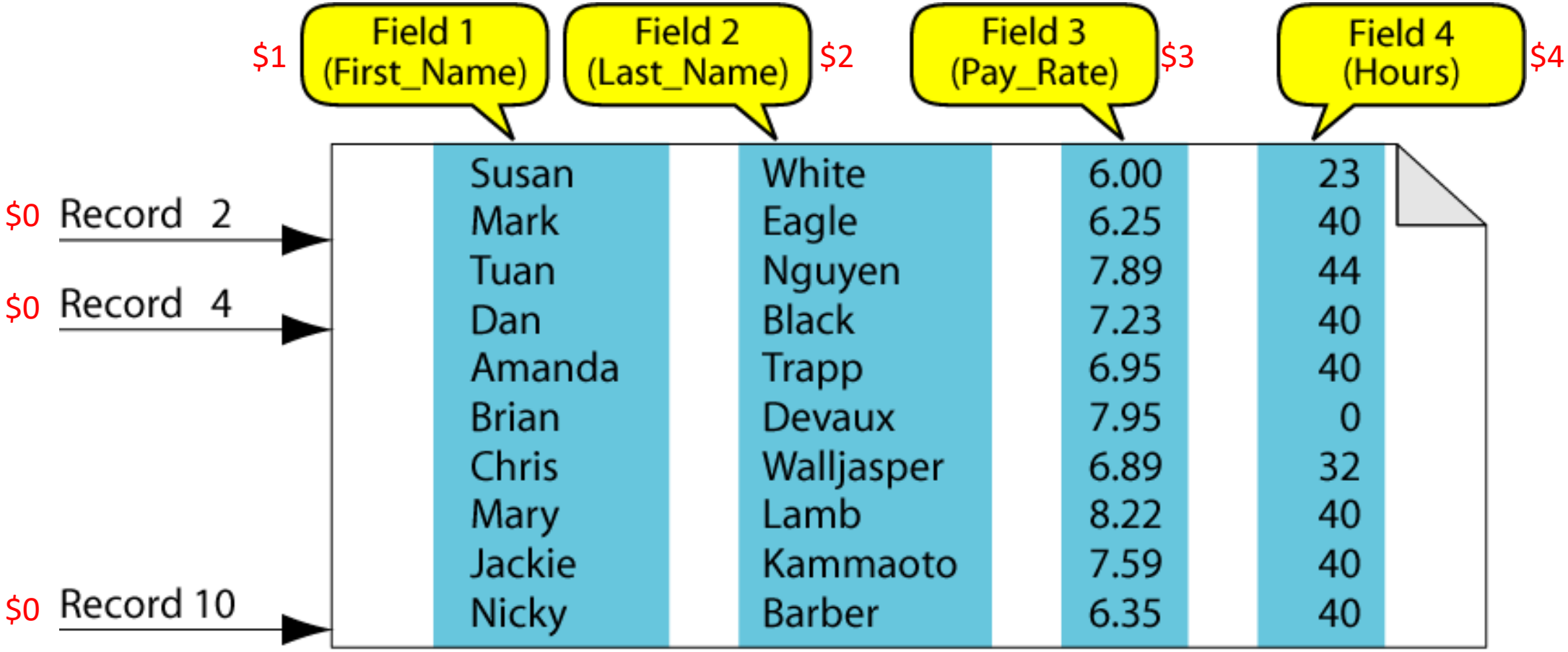
- format output lines
- arithmetic and string operations
- conditionals and loops

The Command: awk





Example Input File



A file with 10 records, each with four fields

Print columns

- $0 = entire line
- $1 = column 1
- $2 = column 2

```
awk '{print $1}' file
```

Print defined columns

- `awk '{print $1}' file`

Will separate by white space

- `awk -F "\t" '{print $1}' file`

Will separate by tabs

- `awk -F "," '{print $1}' file`

Will separate by comma

Some System Variables

FS	Field separator (default=whitespace)
RS	Record separator (default=\n)
NF	Number of fields in current record
NR	Number of the current record
OFS	Output field separator (default=space)
ORS	Output record separator (default=\n)
FILENAME	Current filename

Example: Records and Fields

```
% cat emps
Tom Jones        4424    5/12/66 543354
Mary Adams       5346    11/4/63 28765
Sally Chang      1654    7/22/54 650000
Billy Black      1683    9/23/44 336500

% awk '{print NR, $0}' emps
1 Tom Jones      4424    5/12/66 543354
2 Mary Adams     5346    11/4/63 28765
3 Sally Chang    1654    7/22/54 650000
4 Billy Black    1683    9/23/44 336500
```

Example: Space as Field Separator

```
% cat emps
Tom Jones        4424    5/12/66 543354
Mary Adams       5346    11/4/63 28765
Sally Chang      1654    7/22/54 650000
Billy Black      1683    9/23/44 336500

% awk '{print NR, $1, $2, $5}' emps
1 Tom Jones 543354
2 Mary Adams 28765
3 Sally Chang 650000
4 Billy Black 336500
```

Example: Colon as Field Separator

```
% cat em2
Tom Jones:4424:5/12/66:543354
Mary Adams:5346:11/4/63:28765
Sally Chang:1654:7/22/54:650000
Billy Black:1683:9/23/44:336500

% awk -F: '/Jones/{print $1, $2}' em2
Tom Jones 4424
```

print Example

```
% awk '{print $1, $2}' grades
john 85
andrea 89

% awk '{print $1 "," $2}' grades
john,85
andrea,89
```

print Example

```
% awk '{OFS="-";print $1 , $2}' grades
john-85
andrea-89

% awk '{OFS="-";print $1 "," $2}' grades
john,85
andrea,89
```

Redirecting print output

Print output goes to standard output

unless redirected via:

> “file” → rewrite data into file

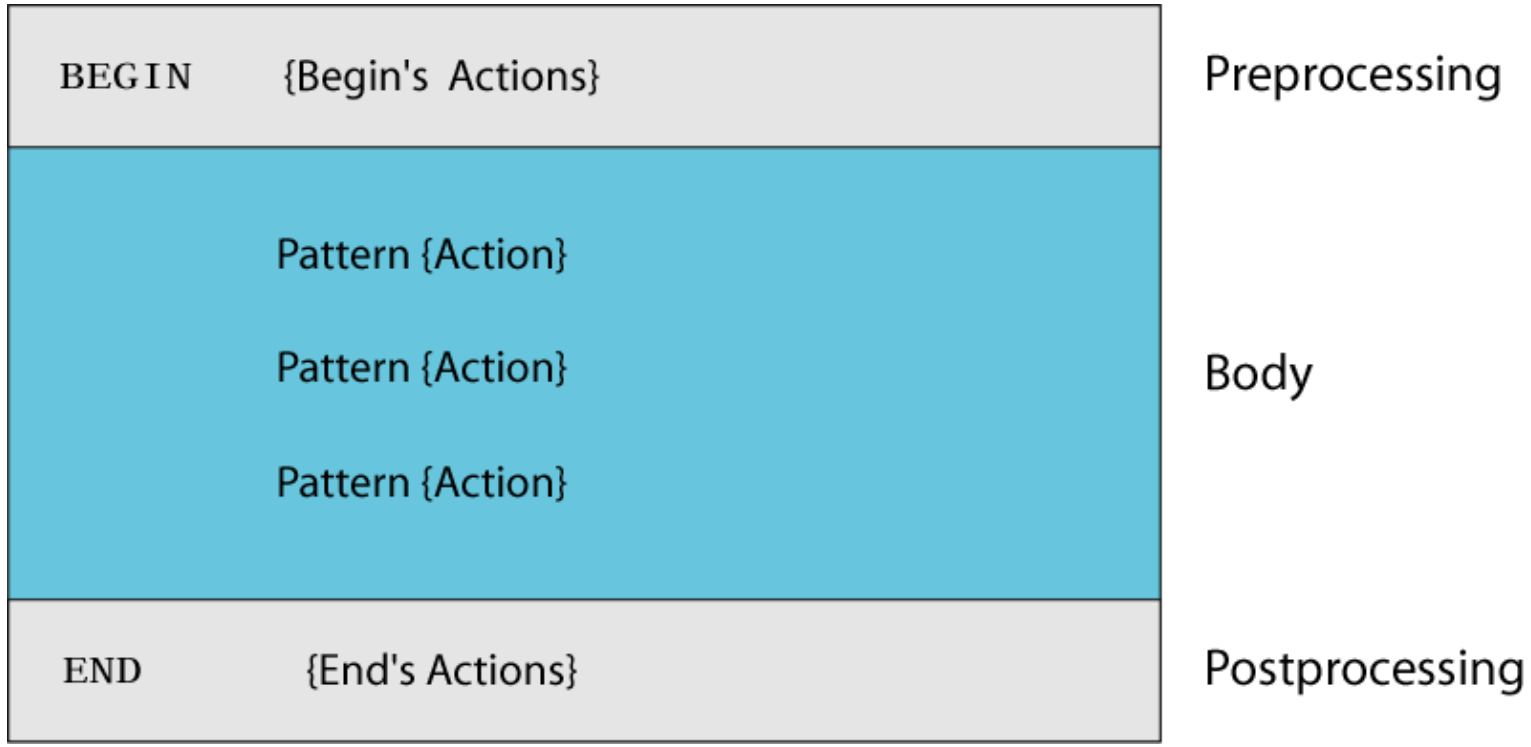
>> “file” → keep previous data and write data from the end

will open file or command only once

subsequent redirections append to already open stream

awk Scripts

awk scripts are divided into three major parts:



comment lines start with #

Expression Pattern types

match

- entire input record

  regular expression enclosed by ‘/’s
- explicit pattern-matching expressions

  ~ (match), !~ (not match)

expression operators

- arithmetic
- relational
- logical

Arithmetic Operators

Operator	Meaning	Example
+	Add	x + y
-	Subtract	x – y
*	Multiply	x * y
/	Divide	x / y
%	Modulus	x % y
^	Exponential	x ^ y

Example:

```
% awk '$3 * $4 > 500 {print $0}' file
```

Relational Operators

Operator	Meaning	Example
<	Less than	x < y
< =	Less than or equal	x < = y
==	Equal to	x == y
!=	Not equal to	x != y
>	Greater than	x > y
> =	Greater than or equal to	x > = y
~	Matched by reg exp	x ~ /y/
!~	Not matched by req exp	x !~ /y/

If conditions

- `awk '{if ($1 == 5) print $0}' file`

If column one is equal to 5, print line

- `awk '{if ($1 != 5) print $0}' file`

If column one is not equal to 5, print line

- `awk '{if ($1 > 5) print $0}' file`

If column one is greater than 5, print line

- `awk '{if ($1 ~ /5/) print $0}' file`

If column contains 5, print line

Example: match input record

```
% cat employees2
Tom Jones:4424:5/12/66:543354
Mary Adams:5346:11/4/63:28765
Sally Chang:1654:7/22/54:650000
Billy Black:1683:9/23/44:336500

% awk -F: '/00$/' employees2
Sally Chang:1654:7/22/54:650000
Billy Black:1683:9/23/44:336500
```

Example: explicit match

```
% cat datafile
northwest NW    Charles Main        3.0    .98    3   34
western   WE    Sharon Gray         5.3    .97    5   23
southwest SW    Lewis Dalsass       2.7    .8     2   18
southern  SO    Suan Chin           5.1    .95    4   15
southeast SE    Patricia Hemenway   4.0    .7     4   17
eastern   EA    TB Savage           4.4    .84    5   20
northeast NE    AM Main             5.1    .94    3   13
north     NO    Margot Weber        4.5    .89    5    9
central   CT    Ann Stephens        5.7    .94    5   13

% awk '$5 ~ /\.[7-9]+/' datafile
southwest SW    Lewis Dalsass       2.7    .8     2   18
central   CT    Ann Stephens        5.7    .94    5   13
```

Logical Operators

Operator	Meaning	Example
&&	Logical AND	a && b
||	Logical OR	a || b
!	NOT	! a

Examples:

```
% awk '($2 > 5) && ($2 <= 15) {print $0}' file
% awk '$3 == 100 || $4 > 50' file
```

If conditions

- `awk '{if ($1 == 5) print $1 ; else print $2}' file`

If column one is equal to 5, print column 1; else print column 2

- `awk '{if ($1 == 5 && $3 ==10) print $1 ; else print $2}' file`

If column one is equal to 5 AND column 3 is equal to 10, print column 1; else print column 2

awk assignment operators

=	assign result of right-hand-side expression to left-hand-side variable
++	Add 1 to variable
--	Subtract 1 from variable
+=	Assign result of addition
-=	Assign result of subtraction
*=	Assign result of multiplication
/=	Assign result of division
%=	Assign result of modulo
^=	Assign result of exponentiation

variables

- Variable allow you to store values (number, word, path, etc.)

- You define a variable as:

```
x=1
y=hello
Data=/home/jm36w/experiment1
z="hello world"
```

- You refer to a defined variable as:

```
$x
$y
$Data
```

Variable for program

計算檔案的總行數

```
1.   wc -l file.txt

2.   F=file.txt
     wc -l $F

3.   F=file
     wc -l $F".txt"
```

Awk example

- File: grades

```
john 85 92 78 94 88
andrea 89 90 75 90 86
jasper 84 88 80 92 84
```

- awk script: average

```
# average five grades
{ total = $2 + $3 + $4 + $5 + $6
  avg = total / 5
  print $1, avg }
```

- Run as:

```
awk -f average grades
```

awk: useful constructions & examples

eg3.txt =
The cow jumped over the moon

if statements

- `awk '{if ($1 == "he") { print $0; }}' eg3.txt`

  (empty)
- `awk '{if ($1 ~ "he") { print $0; } else { … }}' eg3.txt`

  The cow jumped over the moon

for loops

- `awk '{for (j=1; j <= NF; j++) { print $j }}' eg3.txt`

The
cow
jumped
over
the
moon

Awk compare files

- `awk 'FNR==NR {x[$1];next} ($2 in x)' File1.txt File2.txt`

Store column 1 of file 1 in memory

For each line in file 2, if column 2 is in memory print line

Awk fnr nr

```
likegeeks@likegeeks-VirtualBox ~/Desktop
File  Edit  View  Search  Terminal  Help
likegeeks@likegeeks-VirtualBox ~/Desktop $ awk '
> BEGIN {FS=","}
> {print $1,"FNR="FNR,"NR="NR}
> END{print "There were",NR,"records processed"}' myfile myfile
This is a test. FNR=1 NR=1
This is the second test. FNR=2 NR=2
This is the thrid test. FNR=3 NR=3
This is the fourth test. FNR=4 NR=4
This is a test. FNR=1 NR=5
This is the second test. FNR=2 NR=6
This is the thrid test. FNR=3 NR=7
This is the fourth test. FNR=4 NR=8
There were 8 records processed
likegeeks@likegeeks-VirtualBox ~/Desktop $
```

Awk mod

- `awk '{if (NR==1000) print $0' file`

Print the 1000th line

- `awk 'NR%10==1' file`

Print every 10th line starting with line 1

If mod 10(line number) equals 1 print line

- `awk '{if (NR%10==1) print $0}' file`

Awk replace

- `awk '{gsub(/foo/,"bar"); print}' File1.txt`

Replace all instances of foo with bar and print out everything to the terminal

Awk maximum/minimum

- ```
  awk 'BEGIN {max = 0} {if ($3>max) max=$3} END {print max}'
  ```

Print the maximum value of column 3

- ```
  awk 'BEGIN {min = 100} {if ($3 < min) min=$3} END {print min}'
  ```

Print the minimum value of column 3

File combination

a.txt格式如下：

20000401 100000999
20000401 100002999
20000401 100007999
20000401 100013999
20100503 100000999
20100503 400002999
20100503 100007999
20100503 400013999

b.txt格式如下：

100000999 123
100002999 456
100007999 137
100013999 253
400002999 394
400013999 672

形成的c.txt格式如下：

20000401 100000999 123
20000401 100002999 456
20000401 100007999 137
20000401 100013999 253
20100503 100000999 123
20100503 400002999 394
20100503 100007999 137
20100503 400013999 672

```
awk ‘NR==FNR{a[$1]=$2;next}{if($2 in a)print $0,a[$2]}’ b.txt a.txt > c.txt
```

File compare

awk ‘ARGIND==1 {…} ARGIND==2 {…} ARGIND==3 {…} … ‘ file1 file2 file3 …fileN

awk ‘FILENAME==ARGV[1] {…} FILENAME==ARGV[2] {…} FILENAME==ARGV[3] {…} … ‘ file1 file2 file3 …fileN

awk ‘FILENAME==”file1”{…} FILENAME==”file2” {…} FILENAME==”file3” {…} … ‘ file1 file2 file3 …fileN

a.txt	***b.txt***
1	1
qw	2
2	23
123	qw

ARGIND表示awk正在處理的文件（ARGIND==1處理第一個文件，將每條記錄賦值給陣列a，ARGIND==2處理第二個文件檔，通過判斷條件：($1 in a)
當處理第二個檔時，判斷$1是否在陣列a（讀取第一個文件時候生成的陣列）中,此時$1為第二個檔的第一個欄位與讀取第一個檔時時的陣列相同

awk 'ARGIND==1 {a[$0]} ARGIND>1&&!($0 in a) {print $0}' a.txt b.txt

a[1]
a[qw]
a[2]
a[123]

Print header

- ```
  awk 'BEGIN {print “Name \t
  Age”{ print $1 “\t” $2 }' File1.txt
  ```

Print a tab delimited header before the data
  ```

What is sed?

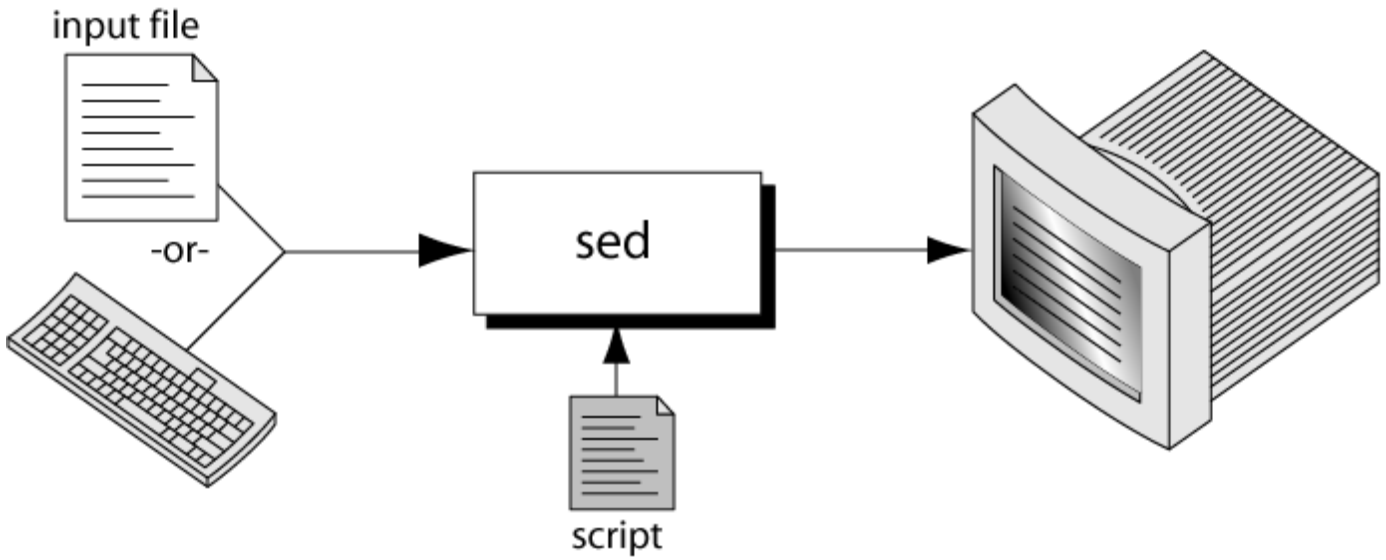
A non-interactive stream editor

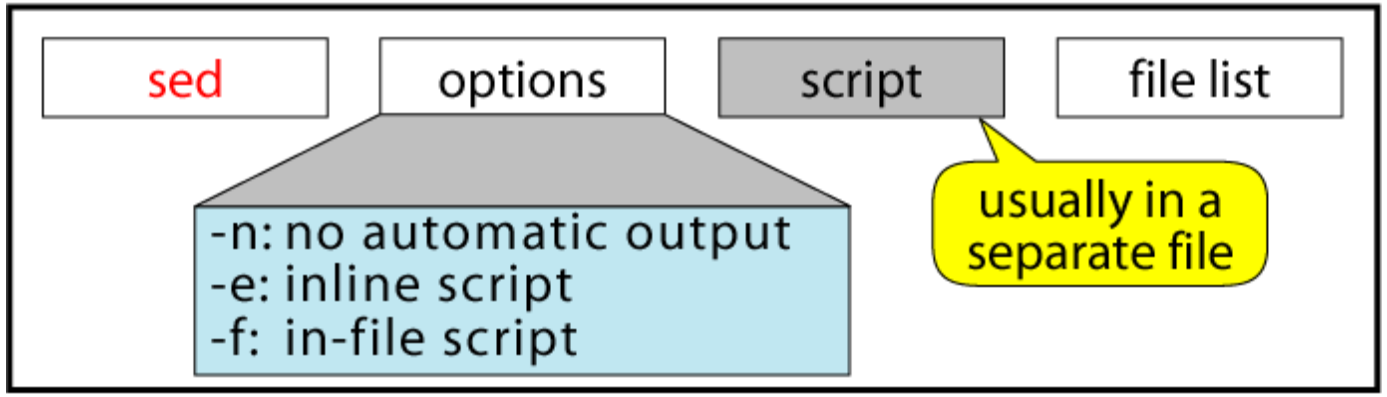
Interprets sed instructions and performs actions

Use sed to:

- Automatically perform edits on file(s)
- Simplify doing the same edits on multiple files
- Write conversion programs

The sed command





sed command syntax

```
$ sed -e 'address command' input_file
```

(a) Inline Script

```
$ sed -f script.sed input_file
```

(b) Script File

How Does sed Work?

sed reads line of input

- line of input is copied into a temporary buffer called pattern space
- editing commands are applied
  - subsequent commands are applied to line in the pattern space, not the original input line
  - once finished, line is sent to output
    (unless –n option was used)
- line is removed from pattern space

sed reads next line of input, until end of file

Note: input file is unchanged

sed

sed [-nefr] [動作]

- -n
  - 使用(silent)模式。在一般 sed 的用法中，所有來自 STDIN 的資料一般都會被列出到螢幕上。但如果加上 -n 參數後，則只有經過 sed 特殊處理的那一行(或者動作)才會被列出來。
- -e
  - 直接在指令列模式上進行 sed 的動作編輯；
- -f
  - 直接將 sed 的動作寫在一個檔案內， -f filename 則可以執行 filename 內的 sed 動作；
- -r
  - sed 的動作支援的是延伸型正規表示法的語法。(預設是基礎正規表示法語法)

sed（續）

sed [-nefr] [n1[,n2]]function

- a
  - 新增， a 的後面可以接字串，而這些字串會在新的一行出現(下一行)
- c
  - 取代， c 的後面可以接字串，這些字串可以取代 n1,n2 之間的行！
- d
  - 刪除，因為是刪除啊，所以 d 後面通常不接任何咚咚；
- i
  - 插入， i 的後面可以接字串，而這些字串會在新的一行出現(上一行)；
- p
  - 列印，亦即將某個選擇的資料印出。通常 p 會與參數 sed -n 一起運作～
- s
  - 搜尋，不但可以搜尋，還能夠進行取代的工作哩！通常這個 s 的動作可以搭配正規表示法！例如 1,20s/old/new/g 就是啦！

sed（續）

範例：

1. 將 /etc/passwd 的內容列出，需要列印行號，且將第 2~5 行刪除

   nl /etc/passwd | sed '2,5d‘

2. 呈上題，在第二行後(亦即是加在第三行)加上『drink tea?』

   nl /etc/passwd | sed '2a drink tea‘

3. 在第二行後面加入兩行字，例如『Drink tea or .....』『drink beer?』

   nl /etc/passwd | sed '2a Drink tea or .....\

   > drink beer ?‘

4. 我想將第2-5行的內容取代成為『No 2-5 number』呢？

   nl /etc/passwd | sed '2,5c No 2-5 number‘

5. 僅列出第 5-7 行

   nl /etc/passwd | sed -n '5,7p'

6. 我們可以使用 ifconfig 來列出 IP ，若僅要 eth0 的 IP 時？

   ifconfig eth0 | grep 'inet ' | sed 's/^.*addr://g' | sed 's/Bcast.*$//g'

Example: Replacement String &

```
$ cat datafile
Charles Main            3.0     .98     3       34
Sharon Gray             5.3     .97     5       23
Patricia Hemenway       4.0     .7      4       17
TB Savage               4.4     .84     5       20
AM Main Jr.             5.1     .94     3       13
Margot Weber            4.5     .89     5        9
Ann Stephens            5.7     .94     5       13

$ sed -e 's/[0-9][0-9]$/&.5/' datafile
Charles Main            3.0     .98     3       34.5
Sharon Gray             5.3     .97     5       23.5
Patricia Hemenway       4.0     .7      4       17.5
TB Savage               4.4     .84     5       20.5
AM Main Jr.             5.1     .94     3       13.5
Margot Weber            4.5     .89     5        9
Ann Stephens            5.7     .94     5       13.5
```

sed: overview

a stream editor

WHEN

- "search-and-replace"
- great for using regular expressions to change something in the text

HOW

- sed 's/regexp/replacement/g'
  - 's/… = substitute
  - …/g' = global replace
    (otherwise will only replace first occurrence on a line!)

sed: (simple) examples

eg.txt =
The cops saw the robber with the binoculars

```
sed 's/robber/thief/g' eg.txt
```

The cops saw the thief with the binoculars

```
sed 's/^/She said, "/g' eg.txt
```

She said, "The cops saw the robber with the binoculars

```
sed 's/^/She said, "/g' eg.txt | sed 's/$/"/g'
```

She said, "The cops saw the robber with the binoculars"

sed: syntax examples (from NLP)

eg2.txt =
(TOP (NP (DT The) (NNS cops)) (VP (VBD saw) (NP (DT the) (NN robber)) (PP (IN with) (NP (DT the) (NNS binoculars)))))

"remove the syntactic labels"
hint!: all of (and only) the syntactic labels start with '('

```
cat eg2.txt | sed 's/([^ ]* //g' | sed 's/)//g'
```

The cops saw the robber with the binoculars

"now add explicit start & stop sentence symbols
(<s> and </s>, respectively)"

```
cat eg2.txt | sed 's/([^ ]* //g' | sed 's/)//g' |
sed 's/^/<s> /g' | sed 's/$/ <\/s>/g'
```

<s> The cops saw the robber with the binoculars </s>

sed: (more complicated) example

eg2.txt =
(TOP (NP (DT The) (NNS cops)) (VP (VBD saw) (NP (DT the) (NN robber)) (PP (IN with) (NP (DT the) (NNS binoculars)))))

"show just the POS-and-word pairs: e.g., (POS word)"

```
cat eg2.txt | sed 's/([^ ]* [^(]/~&/g' |
sed 's/[^)~]*~/ /g' |
sed 's/^ *//g' |
sed 's/))*/)/g'
```

(DT The) (NNS cops) (VBD saw) (DT the) (NN robber) (IN with) (DT the) (NNS binoculars)

Resources

You can always look at the man page for help on any of these tools!

- i.e.: `man sed', or `man tail'

My favorite online resources:

- sed: www.grymoire.com/Unix/Sed.html
- awk: www.vectorsite.net/tsawk.html
- bash: www.tldp.org/LDP/abs/html/
  Google it. ☺