

計算機概論A班 實習課

W5
助教:王常在

課程內容

- awk 指令講解與實作
- sed 指令講解與實作
- 問答時間
- HW
 - Linux 文字處理工具作業*3

awk 文字分析工具

- 常用在對文字和資料進行分析處理
- 檔案逐行的讀入

awk Script

```
awk 'BEGIN{ print "start" } pattern{ commands } END{ print "end" }' filename
```

工作原理:

- 第一步執行BEGIN 語句
- 第二步從檔案或標準輸入讀取一行, 然後再執行pattern語句, 逐行掃描檔案到檔案全部被讀取
- 第三步執行END語句

awk Script

BEGIN { statement }

主要目的是用來更改一些預設的設定, 如: FS (輸入行的欄位分割符號), RS (輸入行的換行分割符號)...等等, 以及讀取命令行的參數

pattern { action }

pattern { action }

pattern { action }

被執行的次數是依據被讀取文字輸入檔有多少行來決定 一般都是在這區塊做資料的判斷統計, 可以在 pattern 中加入一些判斷式, 來決定要不要執行 action.

END { statement }

輸出統計完的結果
(存到檔案, 或是傳到標準輸出)

awk record & field

對Linux來說，每一行讀進來
都是新的一行，所以都是\$0。

	\$1	Field 1 (First_Name)	\$2	Field 2 (Last_Name)	\$3	Field 3 (Pay_Rate)	\$4	Field 4 (Hours)
\$0 Record 2		Susan		White		6.00		23
\$0 Record 4		Mark		Eagle		6.25		40
		Tuan		Nguyen		7.89		44
		Dan		Black		7.23		40
		Amanda		Trapp		6.95		40
		Brian		Devaux		7.95		0
		Chris		Walljasper		6.89		32
		Mary		Lamb		8.22		40
		Jackie		Kammaoto		7.59		40
\$0 Record 10		Nicky		Barber		6.35		40

A file with 10 records, each with four fields

awk 語法

```
awk [options] 'scripts' var=value filename
```

常用引數

- -F 指定分隔符(可是字串或正規表達法), 預設是空白(space)
- -f 從指令碼檔案(awk script)中讀取awk命令
- -v var=value賦值變數, 將外部變數遞給awk

awk print record & field

```
awk -F "\tab" '{print $0}' fruit.txt
```

```
awk '{print $1, $2, $4}' fruit.txt
```

\$0: entire line

\$1: column 1

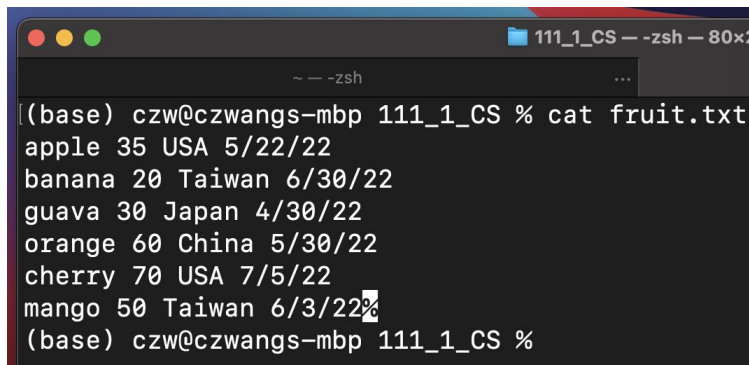
\$2: column 2

awk 參數

- \$0 # 當前record(列、橫行)
- \$1~\$n # 當前record的第 N 個欄位
- FS # 輸入field直欄分隔符(-F相同作用)預設空格
- RS # 輸入record(列、橫行)分割符, 預設換行符
- NF # 欄位個數
- NR # record 數, 就是行號, 預設從 1 開始
- OFS # 輸出欄位分隔符, 預設空格
- ORS # 輸出record分割符, 預設換行符

討論時間

題目：將每行前面加入行號並印出，如下圖二

A terminal window titled '111_1_CS -- zsh -- 80x20' showing the command 'cat fruit.txt' and its output. The output lists six fruit items with their quantities, countries, and dates.

```
(base) czw@czwangs-mbp 111_1_CS % cat fruit.txt
apple 35 USA 5/22/22
banana 20 Taiwan 6/30/22
guava 30 Japan 4/30/22
orange 60 China 5/30/22
cherry 70 USA 7/5/22
mango 50 Taiwan 6/3/22%
(base) czw@czwangs-mbp 111_1_CS %
```

圖一：原本檔案內容

A terminal window showing the same six fruit items as in Figure 1, but each line is now preceded by a line number from 1 to 6.

```
1 apple 35 USA 5/22/22
2 banana 20 Taiwan 6/30/22
3 guava 30 Japan 4/30/22
4 orange 60 China 5/30/22
5 cherry 70 USA 7/5/22
6 mango 50 Taiwan 6/3/22
(base) czw@czwangs-mbp 111_1_CS %
```

圖二

awk 算術運算子

awk '\$2 + \$3 >= 160 {print \$0}' filename

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

```
111_1_CS -- -zsh -- 80x24
~ -- -zsh
(base) czw@czwangs-mbp 111_1_CS % cat people.txt
John 23 177
May 5 120
Tom 1 80
Jenny 15 150
David 27 185
Jennifer 18 158
Henry 38 186
Gary 41 172
Lulu 22 167%
(base) czw@czwangs-mbp 111_1_CS %
```

```
111_1_CS -- -zsh -- 80x24
~ -- -zsh
~/Desktop/111_1_CS -- -zsh
(base) czw@czwangs-mbp 111_1_CS % awk '$2 + $3 >= 188 {print $0}' people.txt
John 23 177
David 27 185
Henry 38 186
Gary 41 172
Lulu 22 167
(base) czw@czwangs-mbp 111_1_CS %
```

awk 關係運算子

```
awk '{if ($3 == 120) print $0}' filename
```

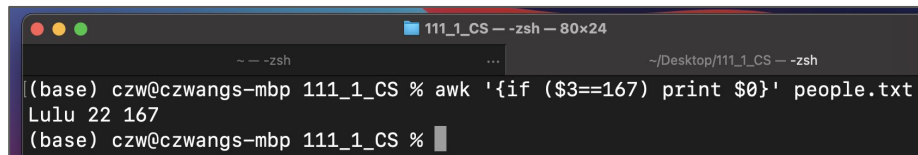
Operator	Meaning	Example
<	Less than	x<y
<=	Less than or equal	x<=y
==	Equal to	x==y
!=	Not equal to	x!=y
>	Greater than	x>y
>=	Greater than or equal to	x>=y

#關係運算子

```
awk '{if ($3 == 120) print $0}' people.txt
```

```
awk '{if ($3 != 120) print $0}' people.txt
```

```
awk '{if ($3 > 120) print $0}' people.txt
```

A terminal window titled '111_1_CS - zsh - 80x24' showing the execution of an awk command. The prompt is '(base) czw@czwangs-mbp 111_1_CS %'. The command entered is 'awk '{if (\$3==167) print \$0}' people.txt'. The output is 'LuLu 22 167'. The prompt then changes to '(base) czw@czwangs-mbp 111_1_CS %' with a cursor.

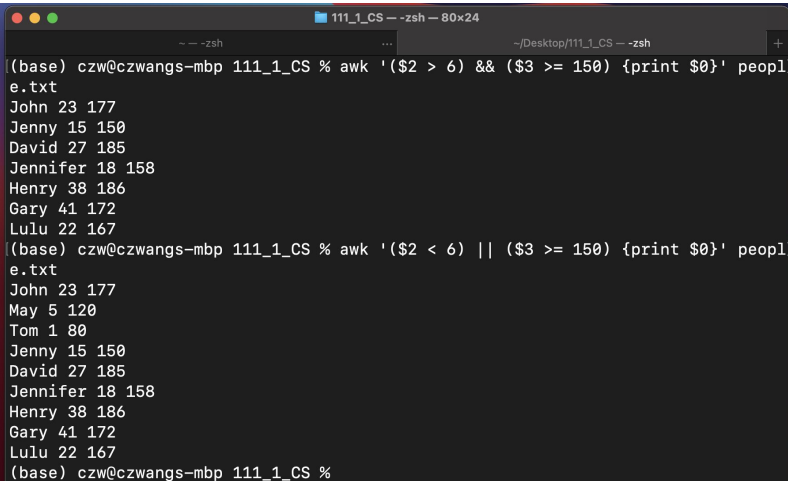
```
(base) czw@czwangs-mbp 111_1_CS % awk '{if ($3==167) print $0}' people.txt
LuLu 22 167
(base) czw@czwangs-mbp 111_1_CS %
```

awk 邏輯運算子

awk '(\$2 > 6) && (\$3 >= 150) {print \$0}' filename

Operator	Meaning	Example
&&	Logical AND	x&& y
	Logical OR	x y

```
awk '($2 > 6) && ($3 >= 150){print$0}' people.txt
awk '($2 < 6) || ($3 >= 150){print$0}' people.txt
```



A terminal window titled '111_1_CS -- zsh -- 80x24' showing the execution of two awk commands. The first command filters for lines where the second field is greater than 6 and the third field is greater than or equal to 150, outputting: John 23 177, Jenny 15 150, David 27 185, Jennifer 18 158, Henry 38 186, Gary 41 172, and Lulu 22 167. The second command filters for lines where the second field is less than 6 or the third field is greater than or equal to 150, outputting: John 23 177, May 5 120, Tom 1 80, Jenny 15 150, David 27 185, Jennifer 18 158, Henry 38 186, Gary 41 172, and Lulu 22 167.

```
(base) czw@czwangs-mbp 111_1_CS % awk '($2 > 6) && ($3 >= 150) {print $0}' people.txt
e.txt
John 23 177
Jenny 15 150
David 27 185
Jennifer 18 158
Henry 38 186
Gary 41 172
Lulu 22 167
(base) czw@czwangs-mbp 111_1_CS % awk '($2 < 6) || ($3 >= 150) {print $0}' people.txt
e.txt
John 23 177
May 5 120
Tom 1 80
Jenny 15 150
David 27 185
Jennifer 18 158
Henry 38 186
Gary 41 172
Lulu 22 167
(base) czw@czwangs-mbp 111_1_CS %
```

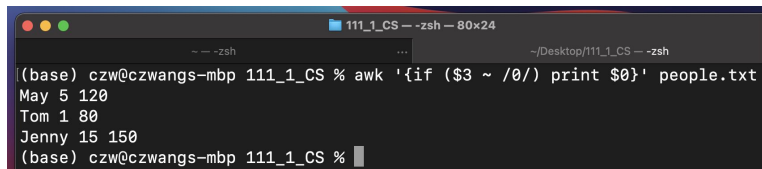
awk 正則運算子

awk '{if (\$3 ~ /0/) print \$0}' filename

Operator	Meaning	Example
~	Matched by reg exp	x~/y/
!~	Not matched by req exp	x!~/y/

#正則運算子

```
awk '{if ($3 ~ /0/) print $0}' people.txt
```



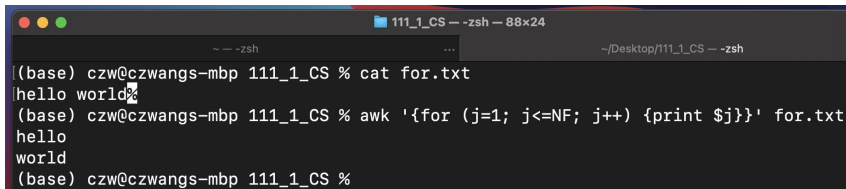
```
111_1_CS -- zsh -- 80x24
(base) czw@czwangs-mbp 111_1_CS % awk '{if ($3 ~ /0/) print $0}' people.txt
May 5 120
Tom 1 80
Jenny 15 150
(base) czw@czwangs-mbp 111_1_CS %
```

awk 賦值運算子

awk '{for (j=1; j <= NF; j++) { print \$j }}' filename

Operator	Meaning
=	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

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



```
111_1_CS -- zsh -- 88x24
-- zsh
(base) czw@czwangs-mbp 111_1_CS % cat for.txt
hello world
(base) czw@czwangs-mbp 111_1_CS % awk '{for (j=1; j<=NF; j++) {print $j}}' for.txt
hello
world
(base) czw@czwangs-mbp 111_1_CS %
```

sed 文字分析工具

- 「stream editor」的縮寫，顧名思義是進行串流 (stream) 的編輯
- 字串取代、複製、刪除的功能
- 自動化的修改文字檔

Sed指令

```
sed [option] '[n1,n2] [command] / [pattern] / [replacement] / [flag]' file.txt
```

- ◆ **option**: 以 - 符號開頭的功能, 如 -n、-r、-i, 可省略
- ◆ **n1,n2**: 代表開始行數和結尾行數, 不輸入代表每行都會執行command
- ◆ **command**: 進行的動作, 如s, a, c, d, i
- ◆ **pattern**: 給command使用的參數
- ◆ **replacement**: 當使用s指令時會使用
- ◆ **flag**: 當使用s指令時會使用

Sed語法 – 常用選項

```
sed [-nefi] '[n1,n2] [command] / [pattern] / [replacement] / [flag]' file.txt
```

option:

- ◆ -n: 沉默模式, 只有經過sed處理的那行才會被印出
- ◆ -e : 直接在命令模式編輯, 預設
- ◆ -f: 直接將sed動作寫在一個檔案內, -f file會直接執行file裡面的動作
- ◆ -i: 修改檔案

Sed語法 – 常用指令

```
sed [-nefi] '[n1,n2] [command] / [pattern] / [replacement] / [flag]' file.txt
```

command:

- ◆ a: 新增, 在**下一行**插入字串, 未指定行數的話, 則是在每一行之後插入字串
- ◆ c: 取代, 取代指定行數的內容
- ◆ d: 刪除, 後面通常不接任何東西
- ◆ i: 插入, 在指定行數的**前一行**插入字串
- ◆ p: 印出, 只印出受影響的行, 常搭配-n使用
- ◆ **s**: 搜尋、取代, 最常用的指令, 可搭配正規表達式使用, 將搜尋的內容進行取代

Sed語法 – 常用旗幟

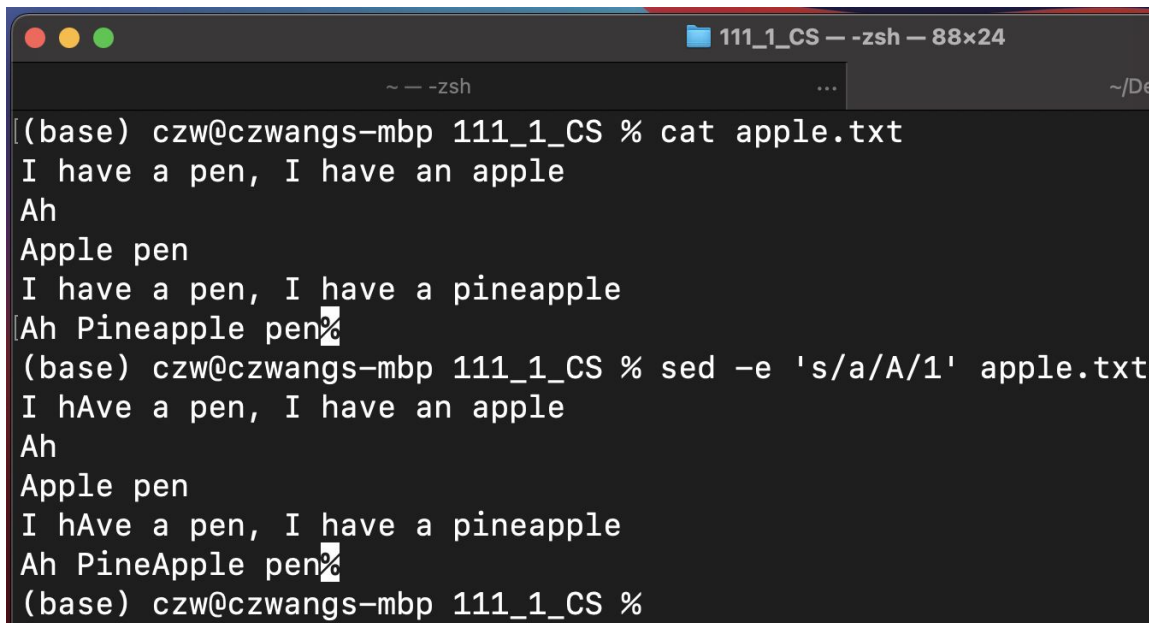
```
sed [-nefi] '[n1,n2] [command] / [pattern] / [replacement] / flag' file.txt
```

flag:

- ◆ g: 全部取代
- ◆ [0-9]: 每行第幾次出現
- ◆ I: 忽略大小寫
- ◆ w: 把符合的結果寫入檔案

Sed應用 – s搜尋並取代

```
sed -e 's/a/A/1' apple.txt
```

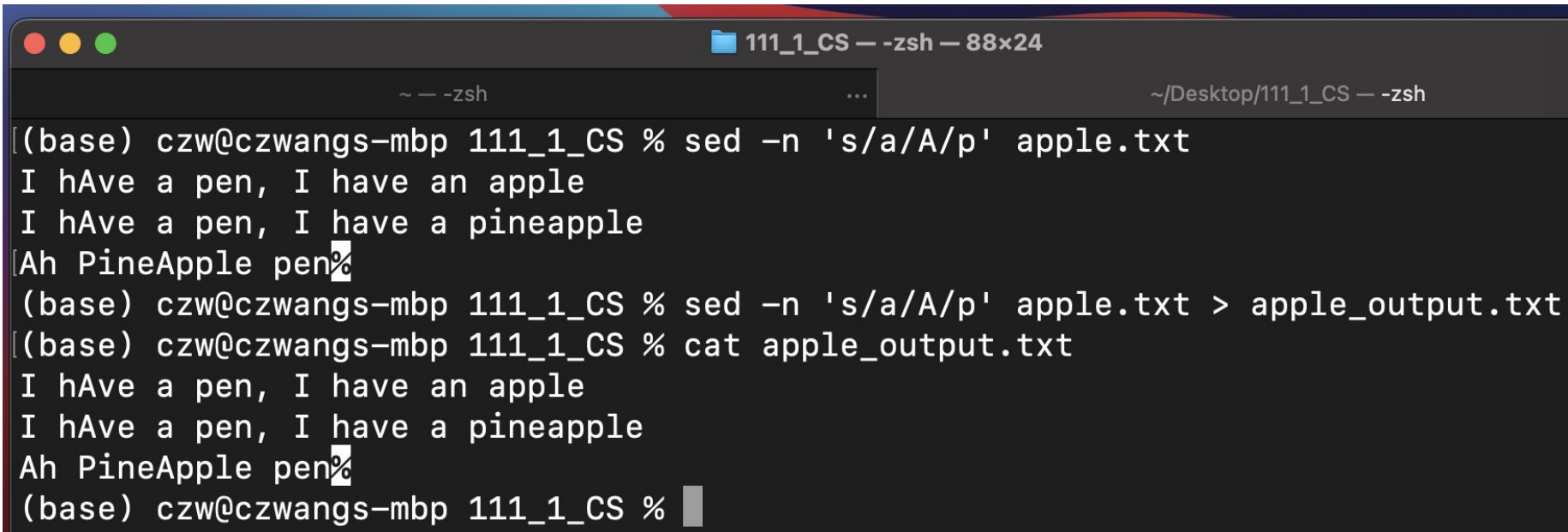


```
(base) czw@czwangs-mbp 111_1_CS % cat apple.txt
I have a pen, I have an apple
Ah
Apple pen
I have a pen, I have a pineapple
Ah Pineapple pen%
(base) czw@czwangs-mbp 111_1_CS % sed -e 's/a/A/1' apple.txt
I hAve a pen, I have an apple
Ah
Apple pen
I hAve a pen, I have a pineapple
Ah PineApple pen%
(base) czw@czwangs-mbp 111_1_CS %
```

Sed應用 – -n沉默模式 + p只印出受影響的行

```
sed -n 's/a/A/p' apple.txt
```

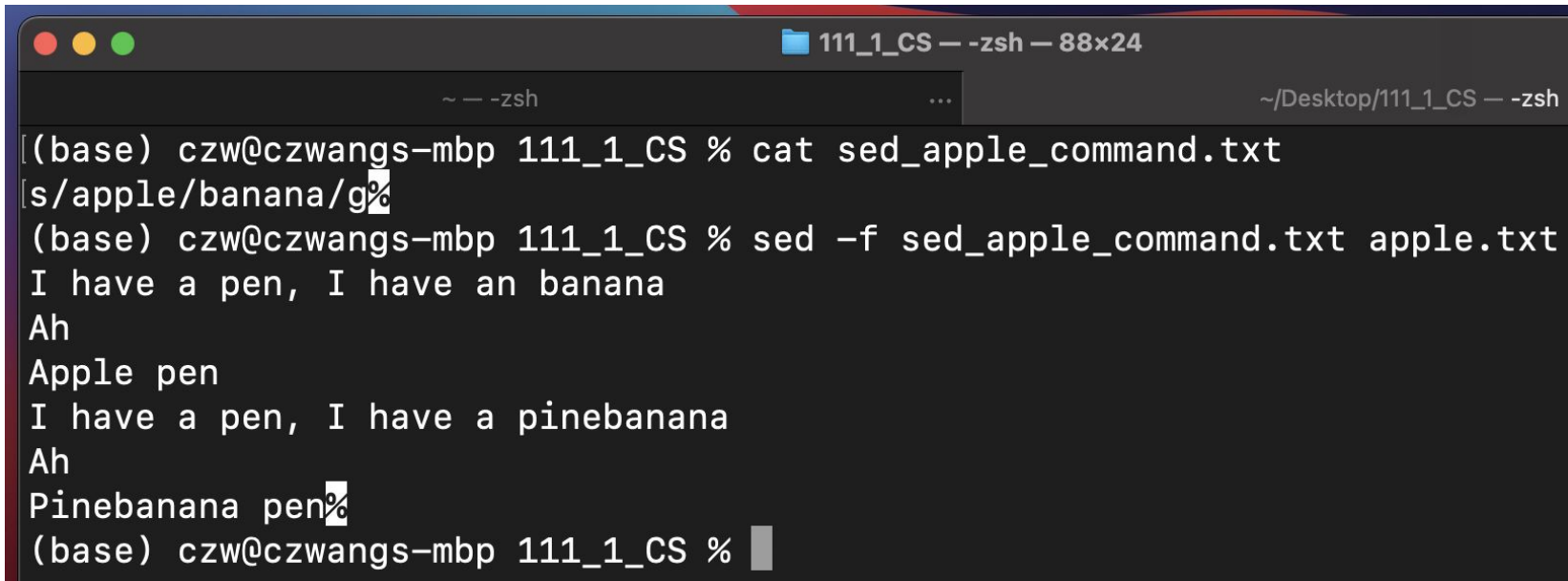
```
sed -n 's/a/A/p' apple.txt > apple_output.txt
```



```
(base) czw@czwangs-mbp 111_1_CS % sed -n 's/a/A/p' apple.txt
I hAve a pen, I have an apple
I hAve a pen, I have a pineapple
Ah PineApple pen%
(base) czw@czwangs-mbp 111_1_CS % sed -n 's/a/A/p' apple.txt > apple_output.txt
(base) czw@czwangs-mbp 111_1_CS % cat apple_output.txt
I hAve a pen, I have an apple
I hAve a pen, I have a pineapple
Ah PineApple pen%
(base) czw@czwangs-mbp 111_1_CS %
```

Sed應用 – -f 讀取檔案手稿

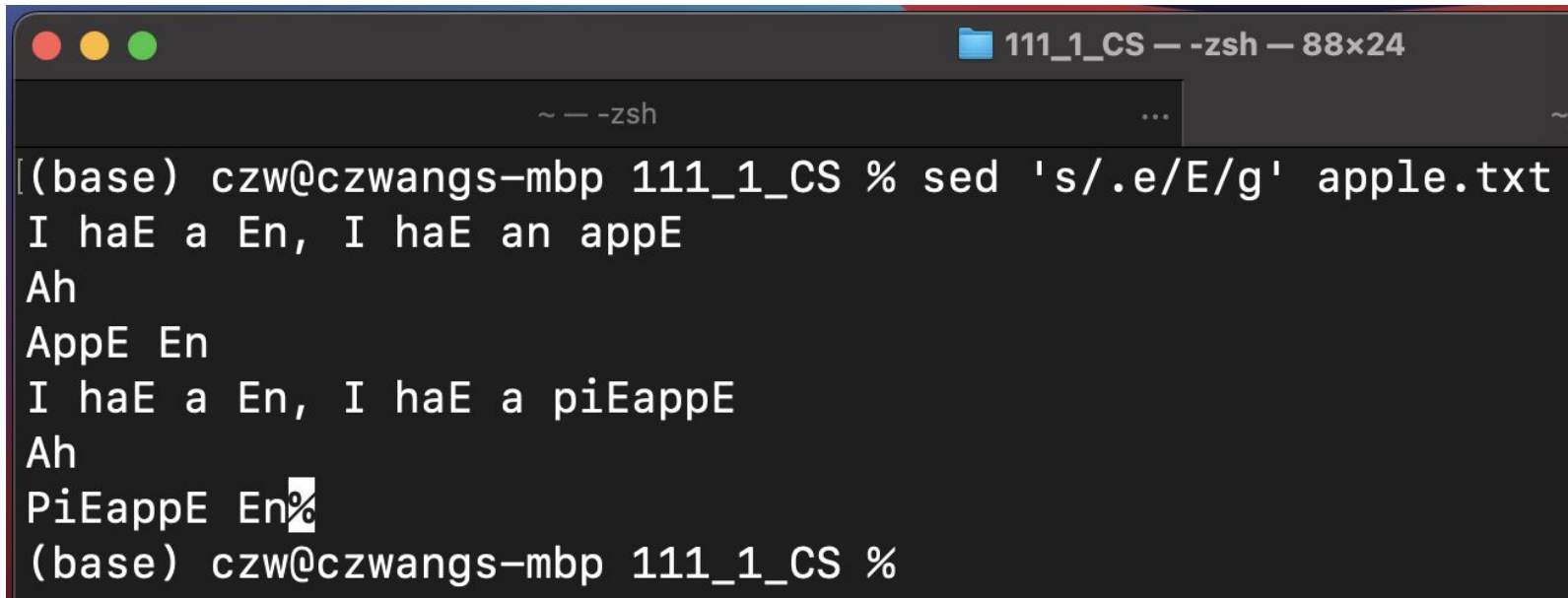
```
sed -f apple_command.txt apple.txt
```



```
(base) czw@czwangs-mbp 111_1_CS % cat sed_apple_command.txt
s/apple/banana/g%
(base) czw@czwangs-mbp 111_1_CS % sed -f sed_apple_command.txt apple.txt
I have a pen, I have an banana
Ah
Apple pen
I have a pen, I have a pinebanana
Ah
Pinebanana pen%
(base) czw@czwangs-mbp 111_1_CS %
```

Sed應用 – s搜尋並取代+正規表達法

```
sed s/.e/E/g' apple.txt
```



A terminal window titled "111_1_CS — -zsh — 88x24" showing the execution of the `sed` command. The prompt is `(base) czw@czwangs-mbp 111_1_CS %`. The command entered is `sed 's/.e/E/g' apple.txt`. The output is as follows:

```
(base) czw@czwangs-mbp 111_1_CS % sed 's/.e/E/g' apple.txt
I haE a En, I haE an appE
Ah
AppE En
I haE a En, I haE a piEappE
Ah
PiEappE En%
(base) czw@czwangs-mbp 111_1_CS %
```


Grep, Awk, Sed比較

→ grep:文字搜尋工具

- ◆ 可用正規表達法, 找出匹配的內容

→ awk:文字分析工具

- ◆ 逐行的讀入, 可搭配正規表達法
- ◆ 主要用在對文字和資料進行分析處理, 以空格為預設分隔符號
- ◆ 同時也是程式語言
- ◆ 用於處理有欄位規則的行內內容, 並支援格式化輸出

→ sed :是一種線上編輯器

- ◆ 它一次處理一行內容, 可搭配正規表達法
- ◆ 主要用來自動編輯一個或多個檔案, 簡化對檔案的反覆操作
- ◆ 用於行間的內容操作, 如新增、刪除、修改、查詢、替換

作業四

Linux 指令與文字處理工具習題 * 3

- 繳交 word 檔至 tronclass 作業區
- 檔名:學號_姓名_計概實習課HW4
- 期限:11/28 23:59前

