# Grep.efi user guide

# 目录

## Revision History

| Version | Comments | Author | Date |
|---|---|---|---|
| 2.0 | Initial Version, (support UEFI v2.6 and UEFI shell v2.2). | krishna | 2019.02.19 |
| 2.1 | 細節優化。 | krishna | 2019.04.17 |

## Convention

Front-iterator：前向迭代器或前指针，通常指向 stream 的第一個單元。

Reverse-iterator：反向迭代器或后指针，通常指向 stream 的最后一個單元。

Scope：由前向迭代器 和 反向迭代器 界定的 stream 范围。

Stream：输入的文件流。

STDIN ：标准输入。

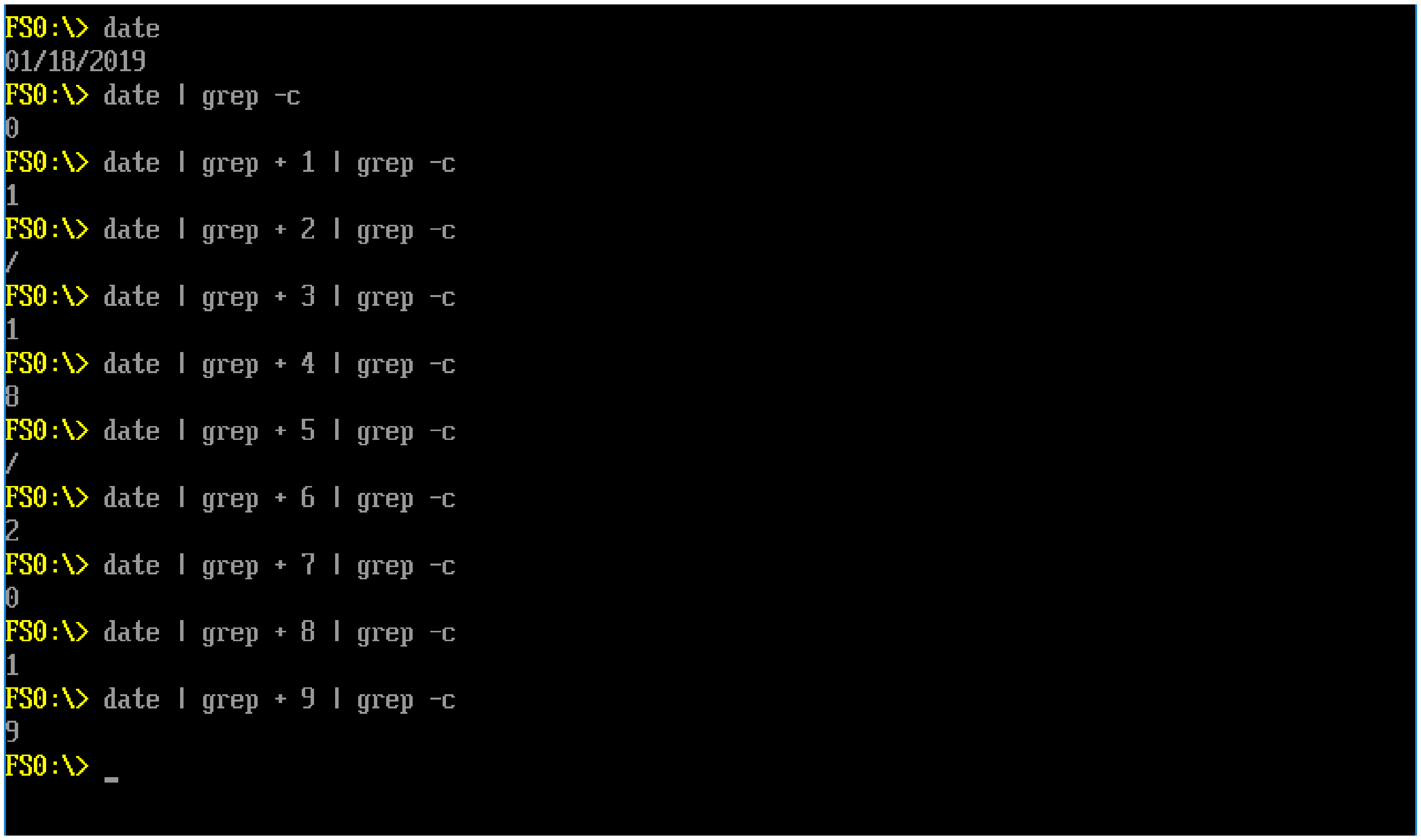
UCS2： Unicode specification, (2 char width) 。

ASCII：American Standard Code for Information Interchange。

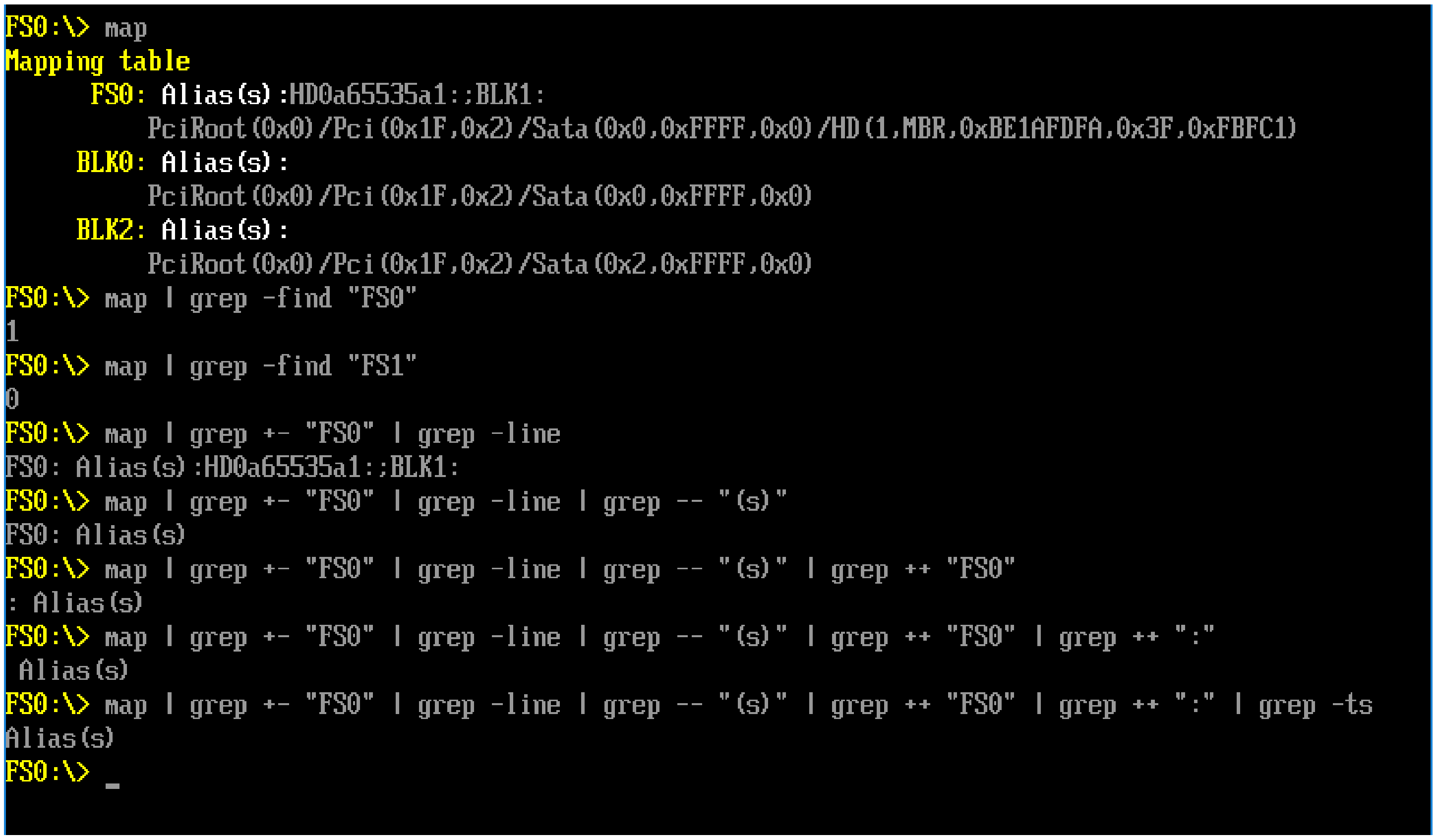
## 一， 管道流操作(直接重定向)

這個工具設計為抓取 console(screen)和各種 config/log 檔，以提供關鍵信息。

一些外置工具，可能不支持這種做法，但 **ReadHistory.efi 可以提供许多支持，請參考 ReadHistory.efi 及相關文檔**。



```
FS0:\> date
01/18/2019
FS0:\> date | grep -c
0
FS0:\> date | grep + 1 | grep -c
1
FS0:\> date | grep + 2 | grep -c
/
FS0:\> date | grep + 3 | grep -c
1
FS0:\> date | grep + 4 | grep -c
8
FS0:\> date | grep + 5 | grep -c
/
FS0:\> date | grep + 6 | grep -c
2
FS0:\> date | grep + 7 | grep -c
0
FS0:\> date | grep + 8 | grep -c
1
FS0:\> date | grep + 9 | grep -c
9
FS0:\> _
```



```
FS0:\> map
Mapping table
      FS0: Alias(s):HD0a65535a1:;BLK1:
          PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0)/HD(1,MBR,0xBE1AFDFA,0x3F,0xFBFC1)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x2,0xFFFF,0x0)
FS0:\> map | grep -find "FS0"
1
FS0:\> map | grep -find "FS1"
0
FS0:\> map | grep +- "FS0" | grep -line
FS0: Alias(s):HD0a65535a1:;BLK1:
FS0:\> map | grep +- "FS0" | grep -line | grep -- "(s)"
FS0: Alias(s)
FS0:\> map | grep +- "FS0" | grep -line | grep -- "(s)" | grep ++ "FS0"
: Alias(s)
FS0:\> map | grep +- "FS0" | grep -line | grep -- "(s)" | grep ++ "FS0" | grep ++ ":"
 Alias(s)
FS0:\> map | grep +- "FS0" | grep -line | grep -- "(s)" | grep ++ "FS0" | grep ++ ":" | grep -ts
Alias(s)
FS0:\> _
```

## 二， 管道流操作(結合 type 命令重定向)

### 1，抓取一个字、词、行

grep –char 读取前指针-指向的-第一個可见的字符

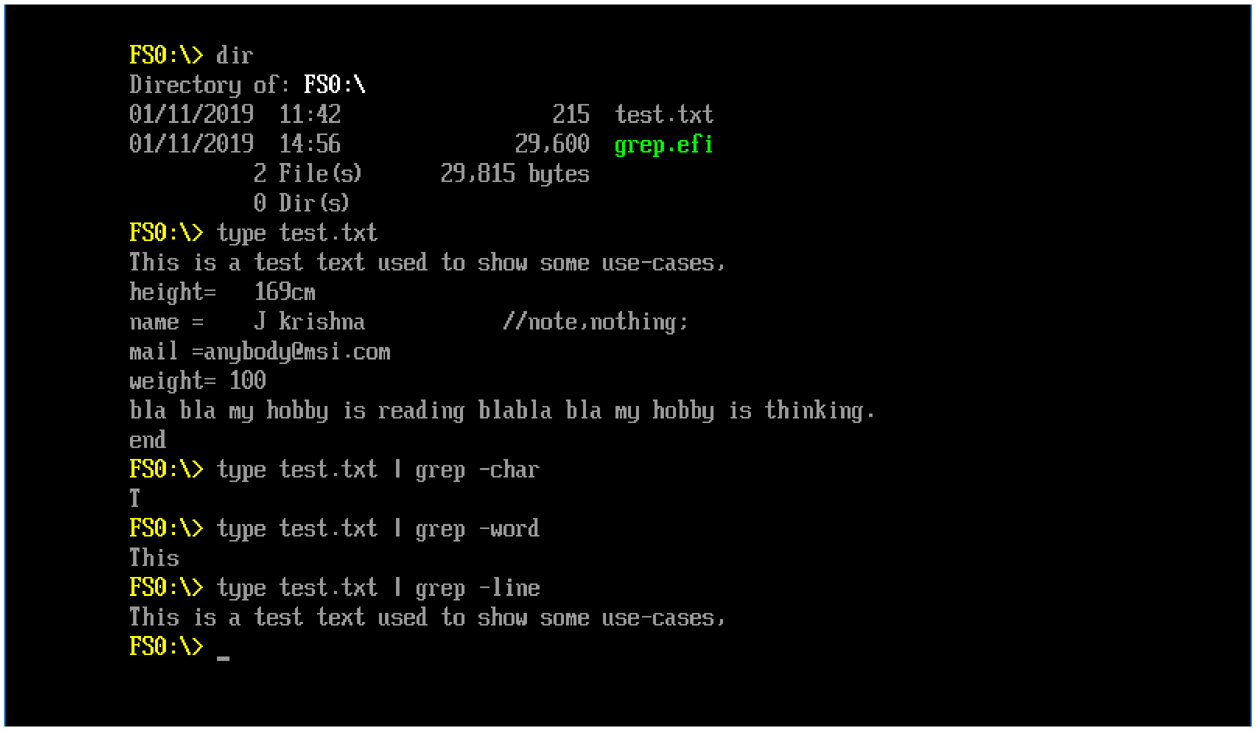
grep –word 读取前指针-指向的-第一個可见的單词(通常即是英文單詞)。

如果前指针指向的不是可见的字符，前指针会自动后移，直到以一个可见的字符为开始。

如果一个可见的字符都没有，输出为空。

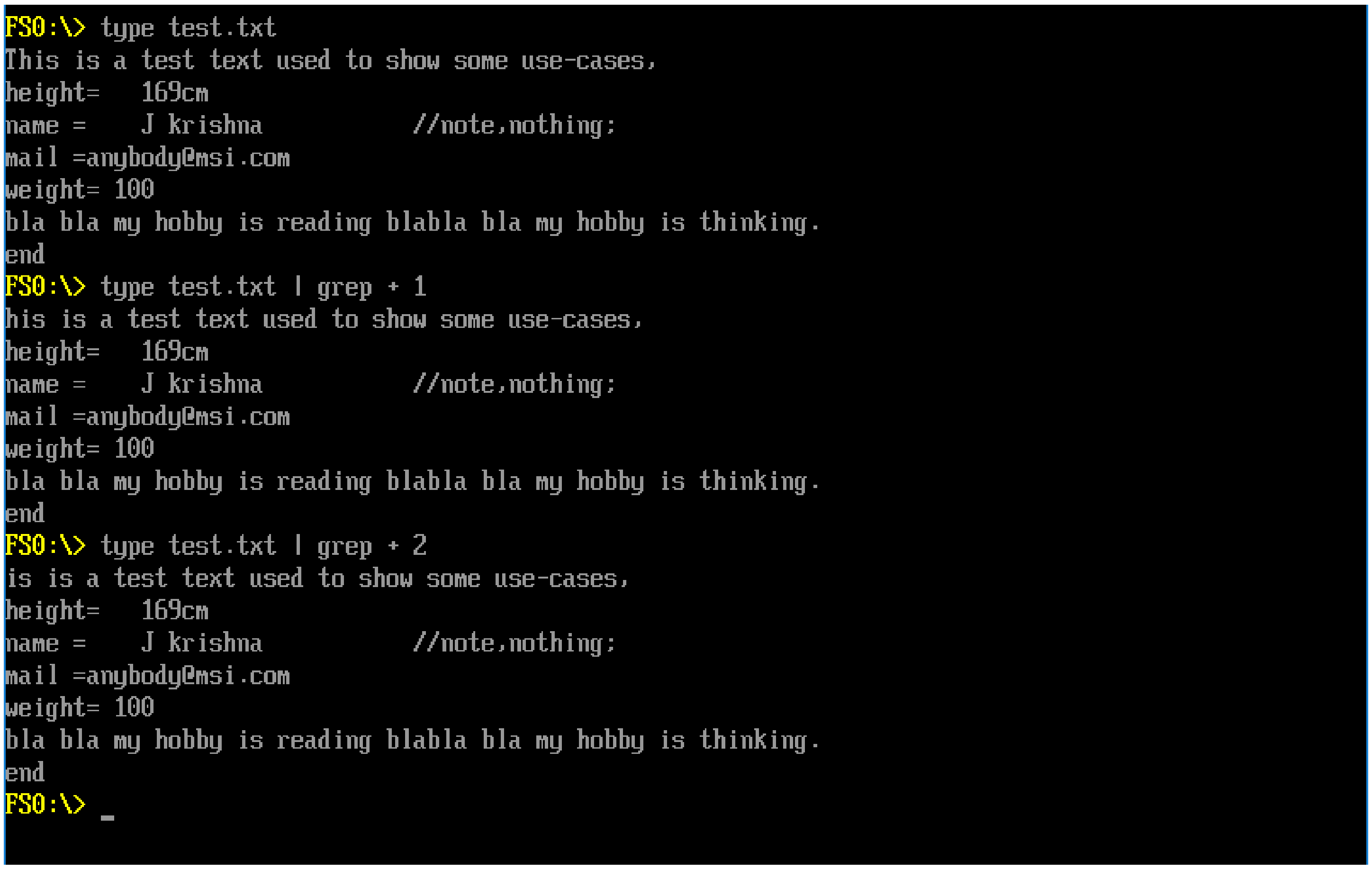
Grep –line 读取前指针-指向的-第一行。

通常，空格/Tab/Enter 都是不可見字符，可見的字符具體定義範圍是 ASCII/UCS2 的(0x21~0x7E) 。


```
FS0:\> dir
Directory of: FS0:\
01/11/2019  11:42                  215  test.txt
01/11/2019  14:56               29,600  grep.efi
          2 File(s)      29,815 bytes
          0 Dir(s)
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=    169cm
name =     J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep -char
T
FS0:\> type test.txt | grep -word
This
FS0:\> type test.txt | grep -line
This is a test text used to show some use-cases,
FS0:\> _
```

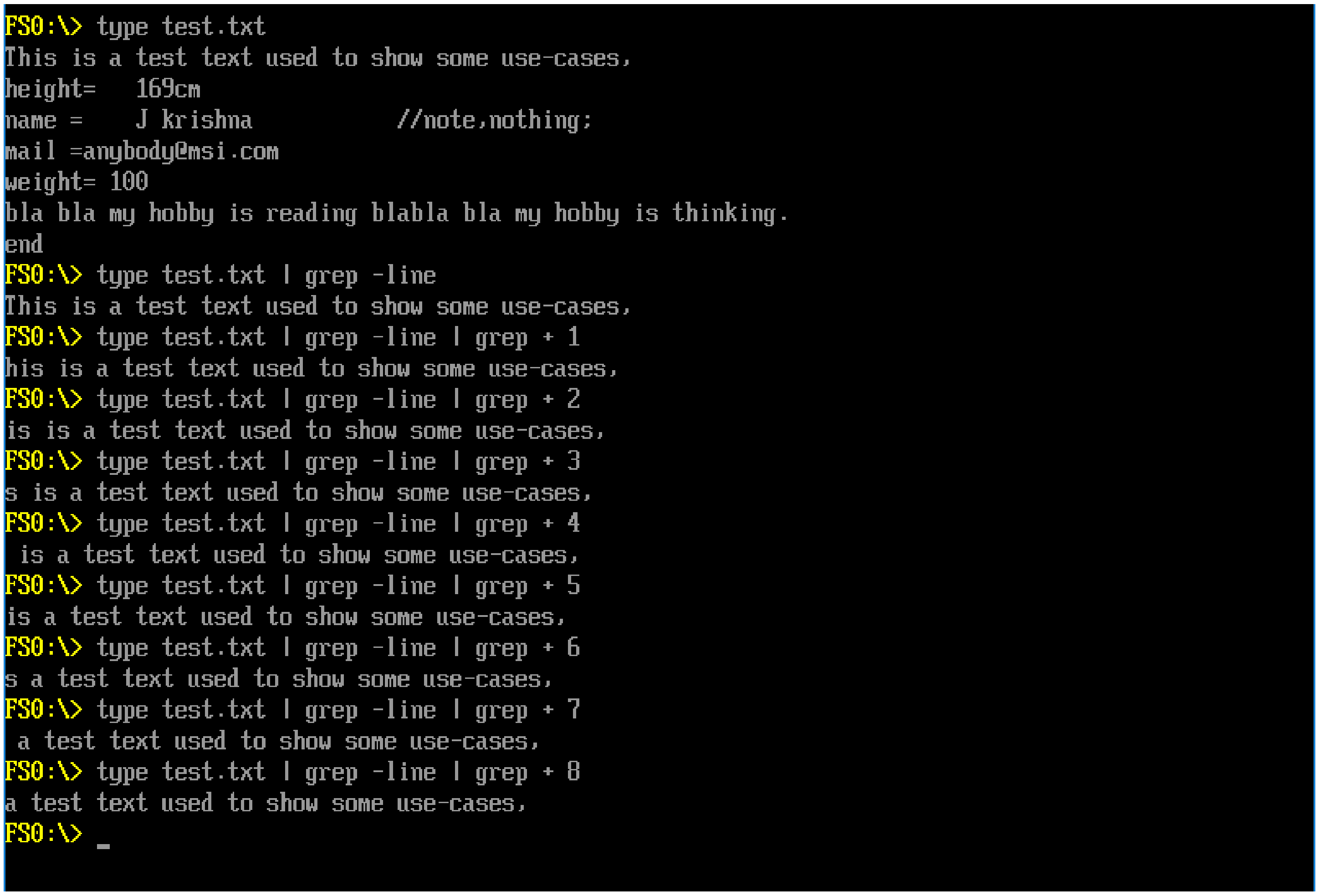
## 2，移动前指针 01

前针指只会往后移动。如果移动的字符数大于整个范围(scope)，则输出为空。



```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep + 1
his is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep + 2
is is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> _
```

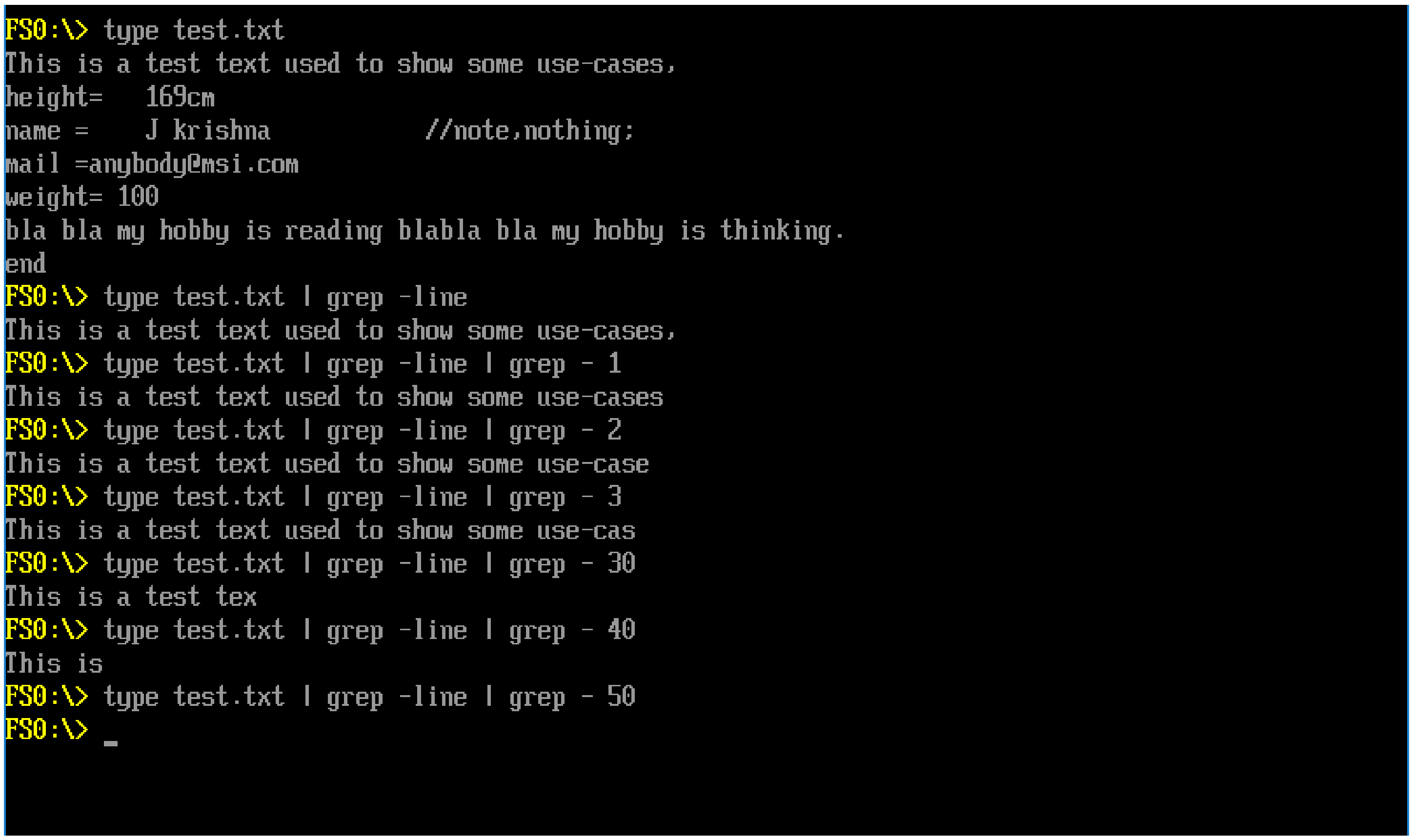
## 3，移动前指针 02



```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna          //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep -line
This is a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 1
his is a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 2
is is a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 3
s is a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 4
 is a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 5
is a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 6
s a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 7
 a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep + 8
a test text used to show some use-cases,
FS0:\> _
```

## 4，移动后指针

后指针只会往前移动。如果移动的字符数大于整个范围(scope)，则输出为空。

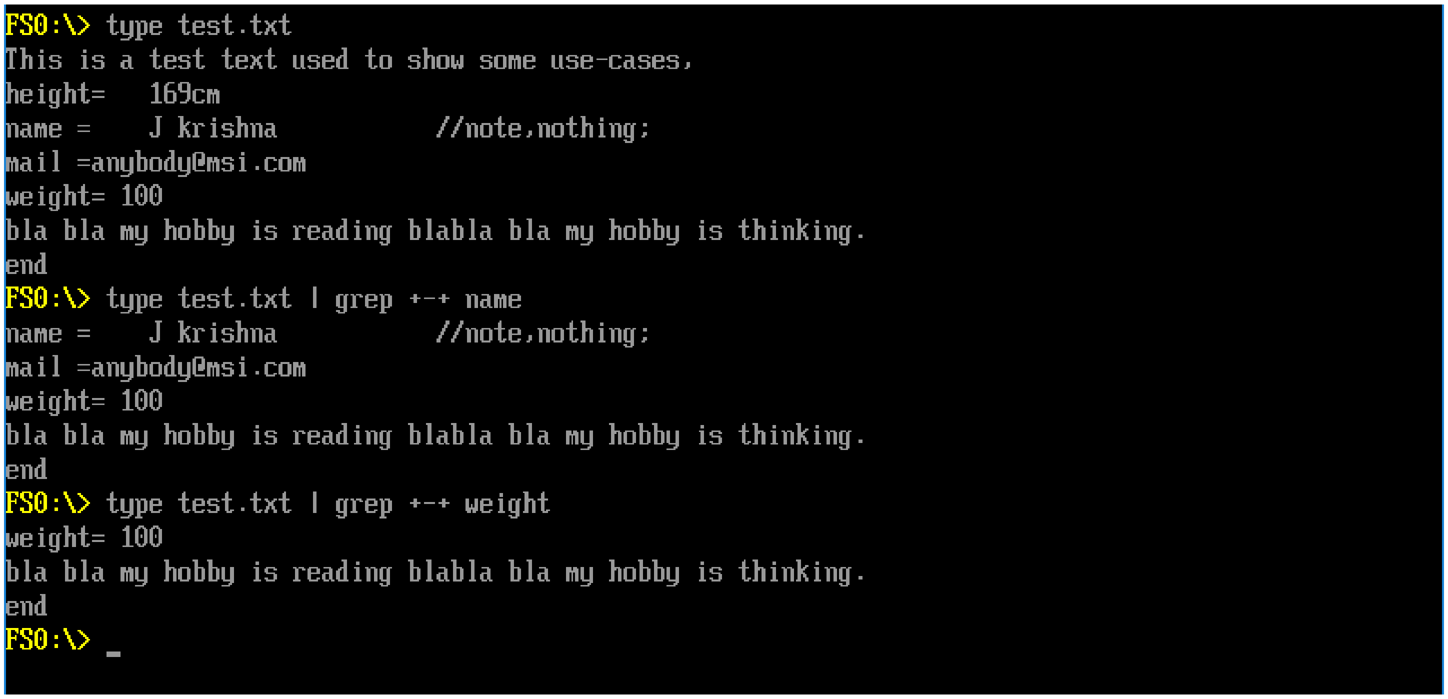


```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep -line
This is a test text used to show some use-cases,
FS0:\> type test.txt | grep -line | grep - 1
This is a test text used to show some use-cases
FS0:\> type test.txt | grep -line | grep - 2
This is a test text used to show some use-case
FS0:\> type test.txt | grep -line | grep - 3
This is a test text used to show some use-cas
FS0:\> type test.txt | grep -line | grep - 30
This is a test tex
FS0:\> type test.txt | grep -line | grep - 40
This is
FS0:\> type test.txt | grep -line | grep - 50
FS0:\> _
```

## 5，前指针转到指定字符串

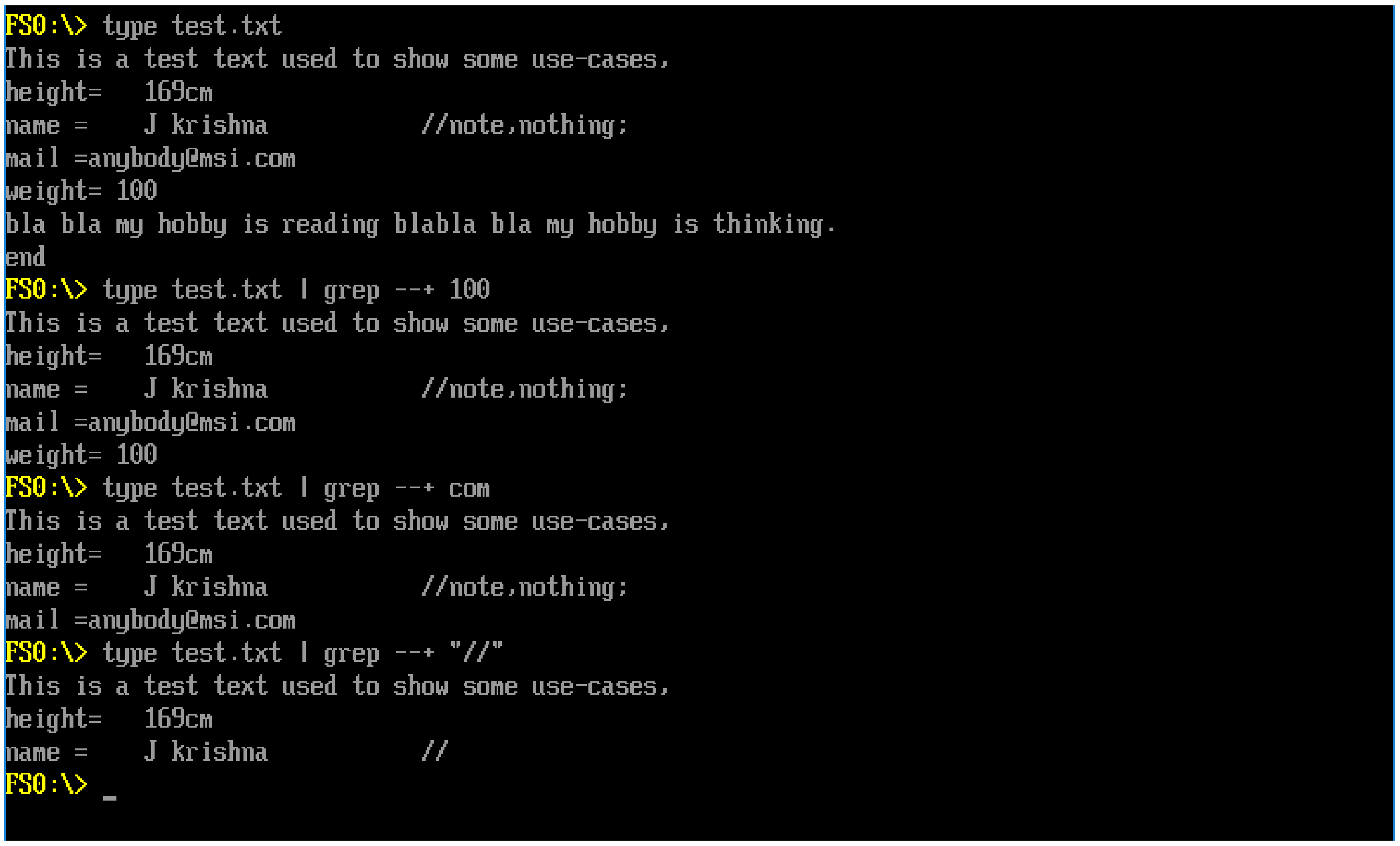
Force mode 下，转到指定字符串失败 (例如字符串不存在)， 会导致输出为空。

Non-force mode 下，转到指定字符串失败 (例如字符串不存在)， 会导致输出 (维持在操作失败之前的样子)， 適用于去掉一些可有可無的字符，例如注釋。



```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +-+ name
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +-+ weight
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> _
```

## 6，后指针转到指定字符串

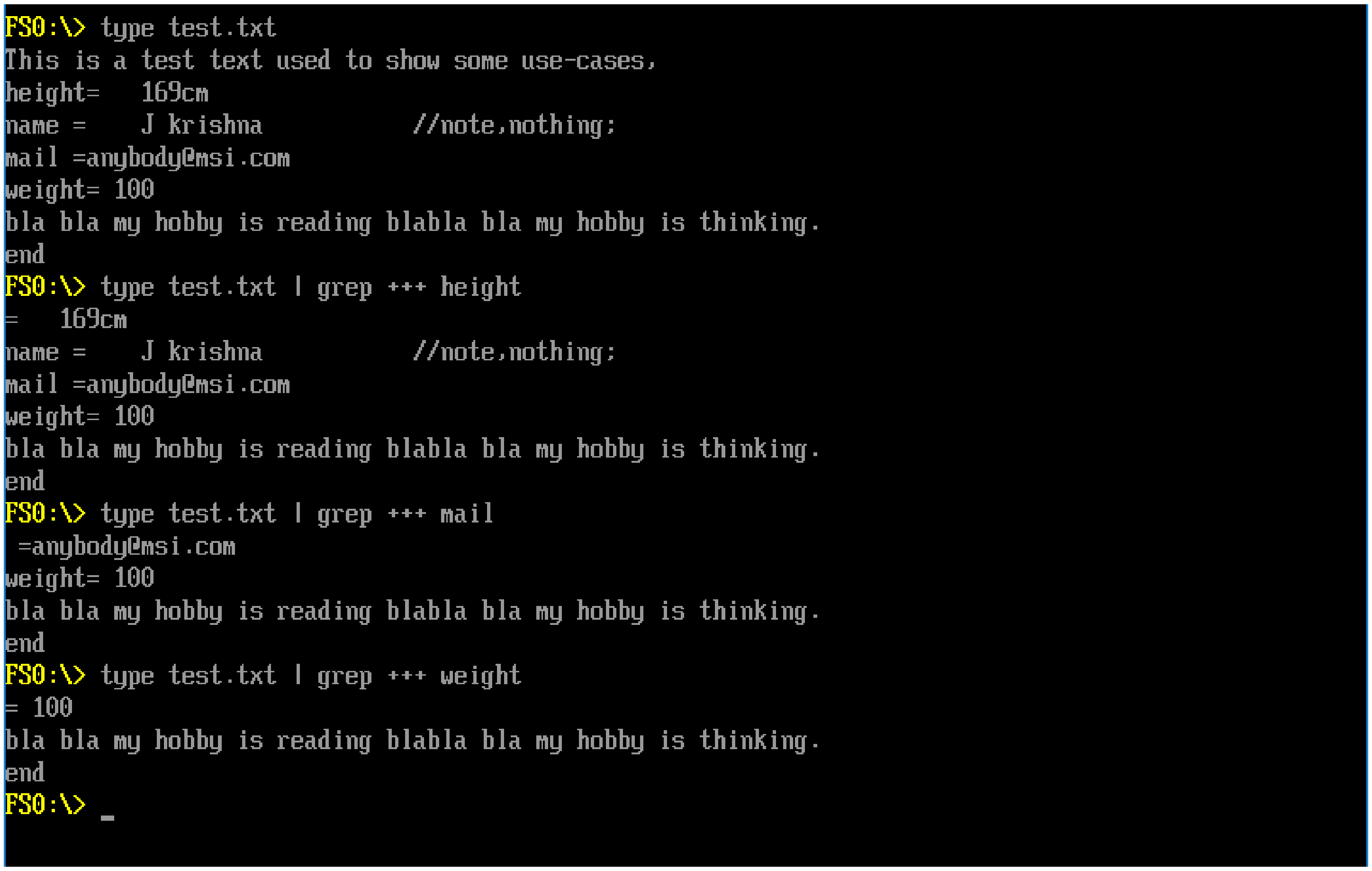


```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep --+ 100
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
FS0:\> type test.txt | grep --+ com
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
FS0:\> type test.txt | grep --+ "//"
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //
FS0:\> _
```

## 7，前指针跳过指定字符串

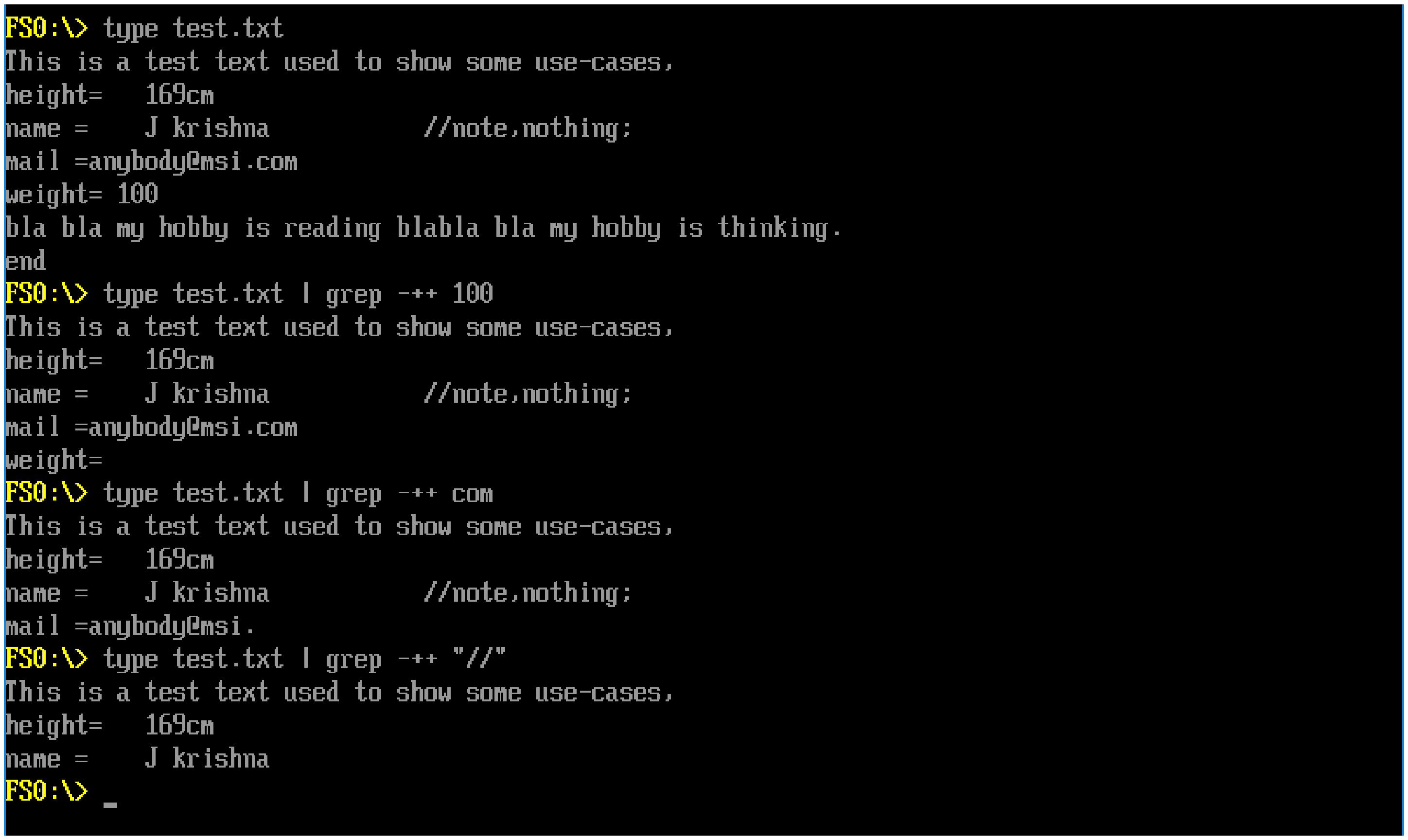
Force mode 下，跳过指定字符串失败 (例如字符串不存在)， 会导致输出为空。

Non-force mode 下，跳过指定字符串失败 (例如字符串不存在)， 会导致输出 (维持在操作失败之前的样子) 。



```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ height
=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ mail
 =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ weight
= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> _
```

## 8，后指针跳过指定字符串

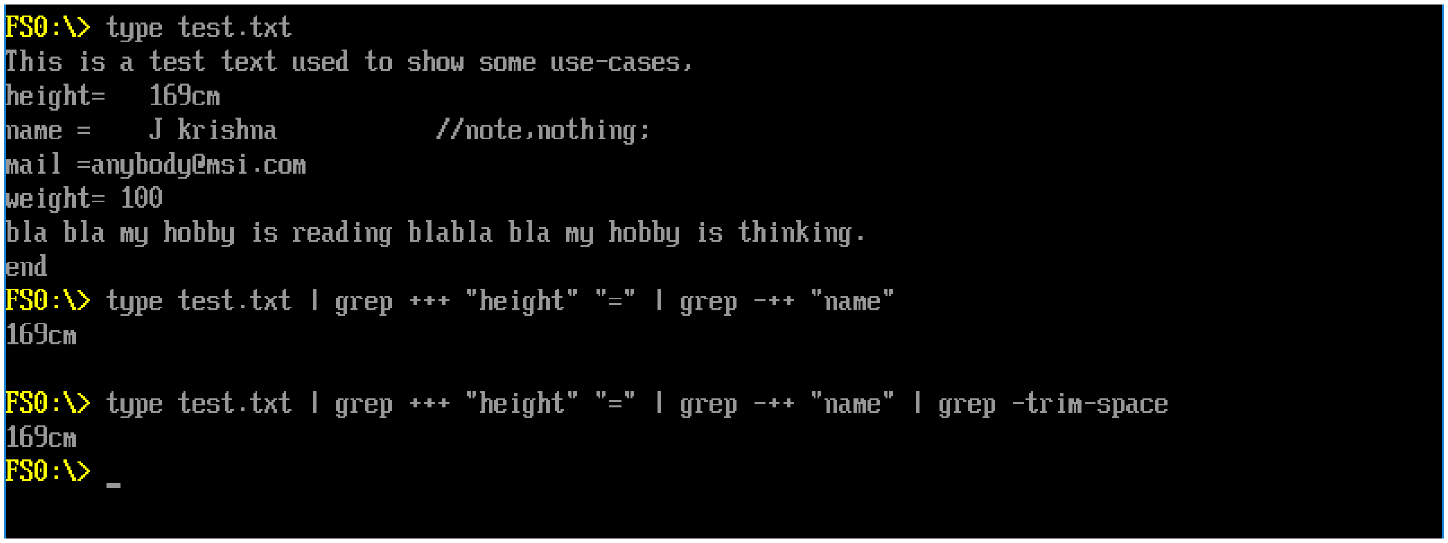


```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep -++ 100
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight=
FS0:\> type test.txt | grep -++ com
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.
FS0:\> type test.txt | grep -++ "//"
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna
FS0:\> _
```

## 9，前后指针连跳，结合使用

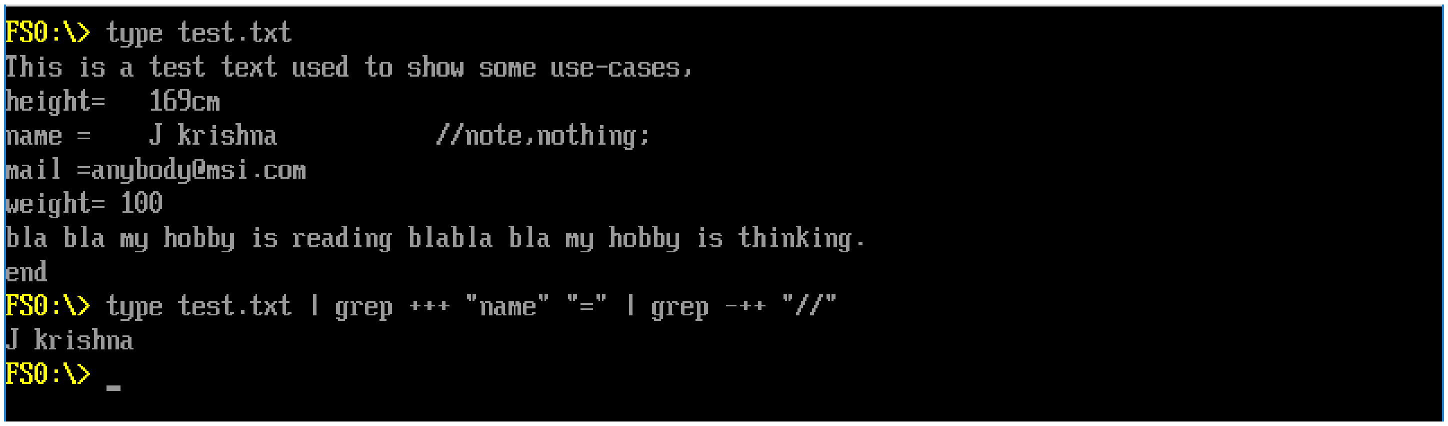
這個工具提供的所有操作都可以結合管道，進行連續的操作。

Grep –trim-sapce 用于去掉首尾可能存在的不可见字符。



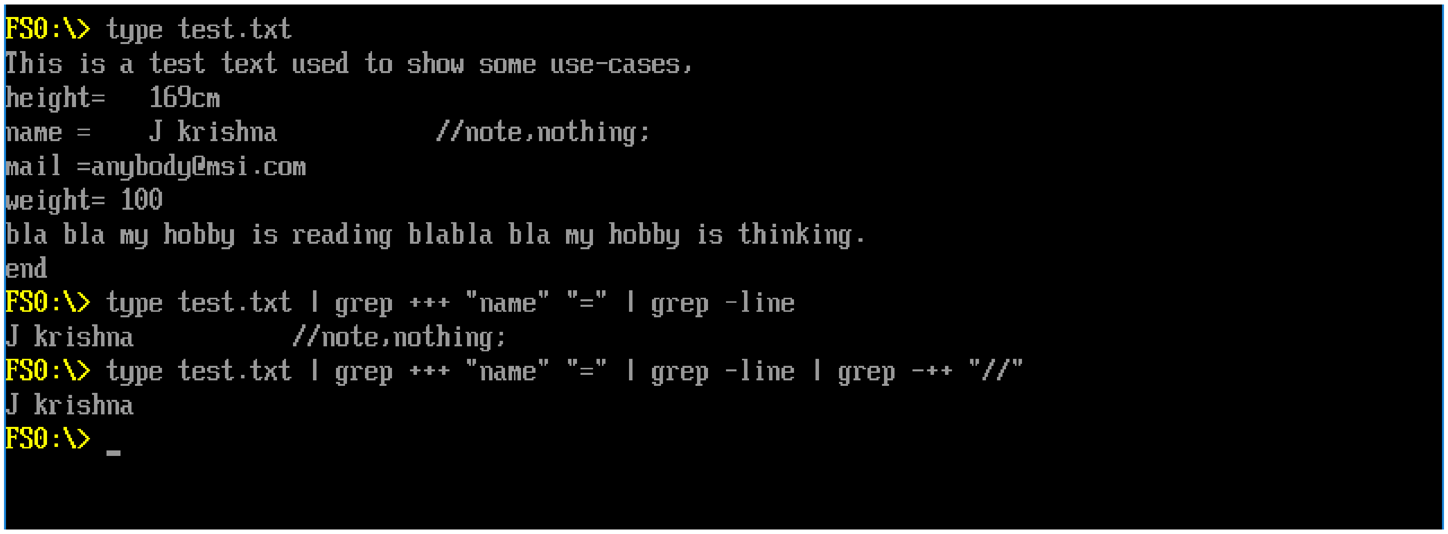
```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ "height" "=" | grep -++ "name"
169cm

FS0:\> type test.txt | grep +++ "height" "=" | grep -++ "name" | grep -trim-space
169cm
FS0:\> _
```

## 10，更多的例子 01



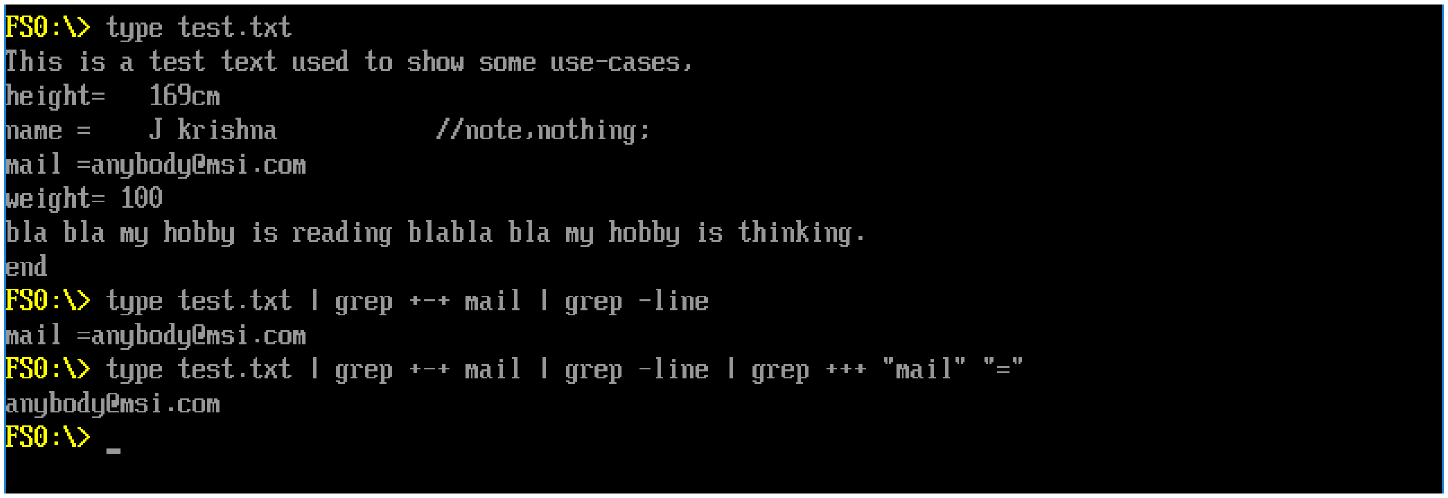
```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ "name" "=" | grep -++ "//"
J krishna
FS0:\> _
```

方法并不是唯一的。



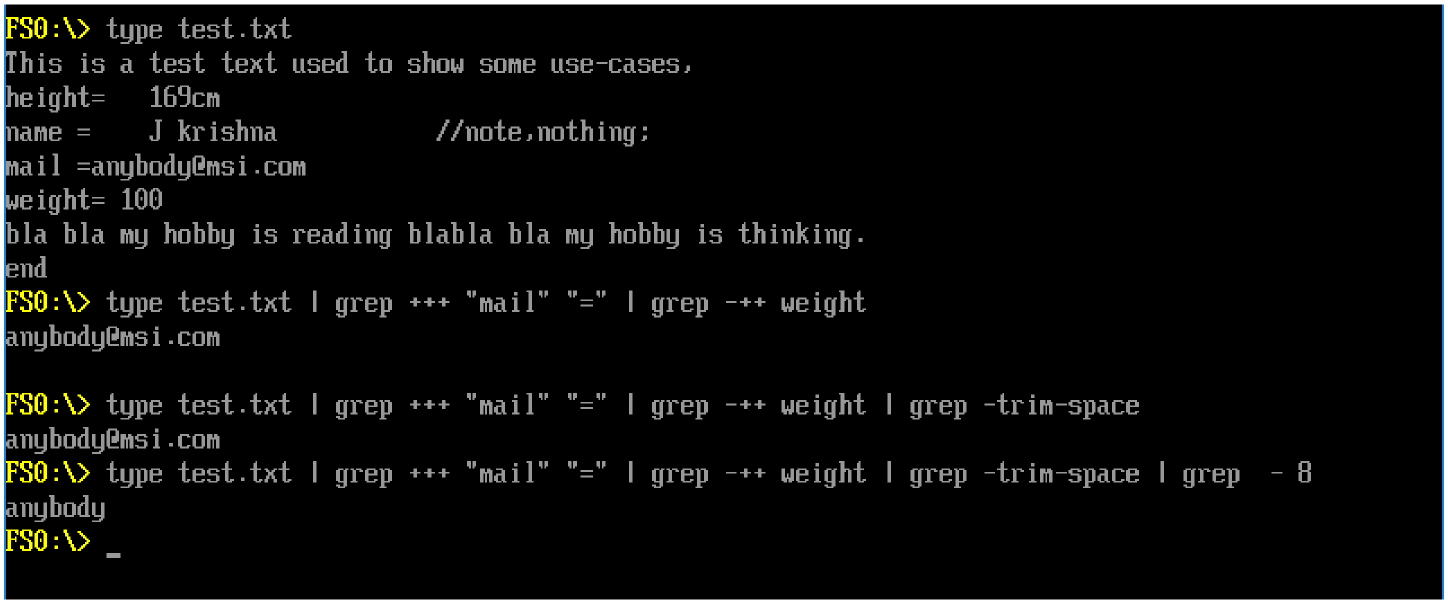
```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ "name" "=" | grep -line
J krishna           //note,nothing;
FS0:\> type test.txt | grep +++ "name" "=" | grep -line | grep -++ "//"
J krishna
FS0:\> _
```

## 11，更多的例子 02



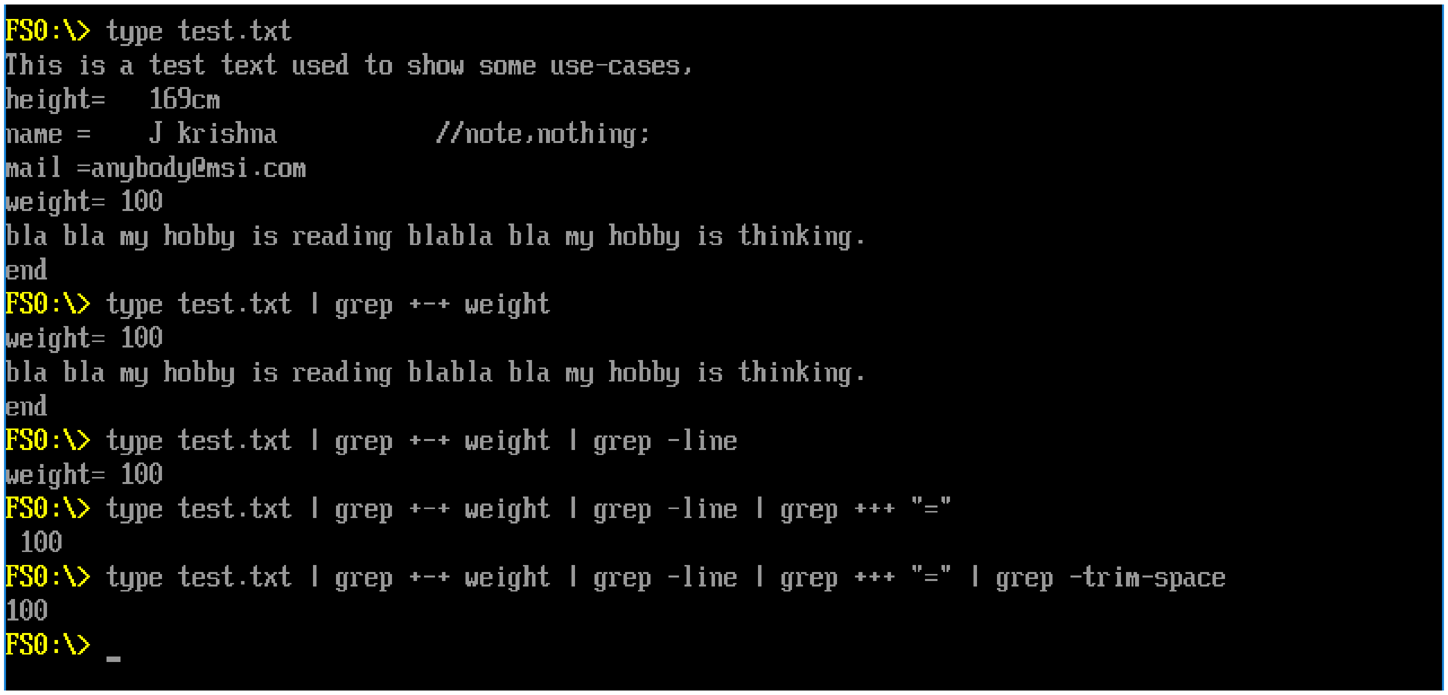
```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +-+ mail | grep -line
mail =anybody@msi.com
FS0:\> type test.txt | grep +-+ mail | grep -line | grep +++ "mail" "="
anybody@msi.com
FS0:\> _
```

方法并不是唯一的。

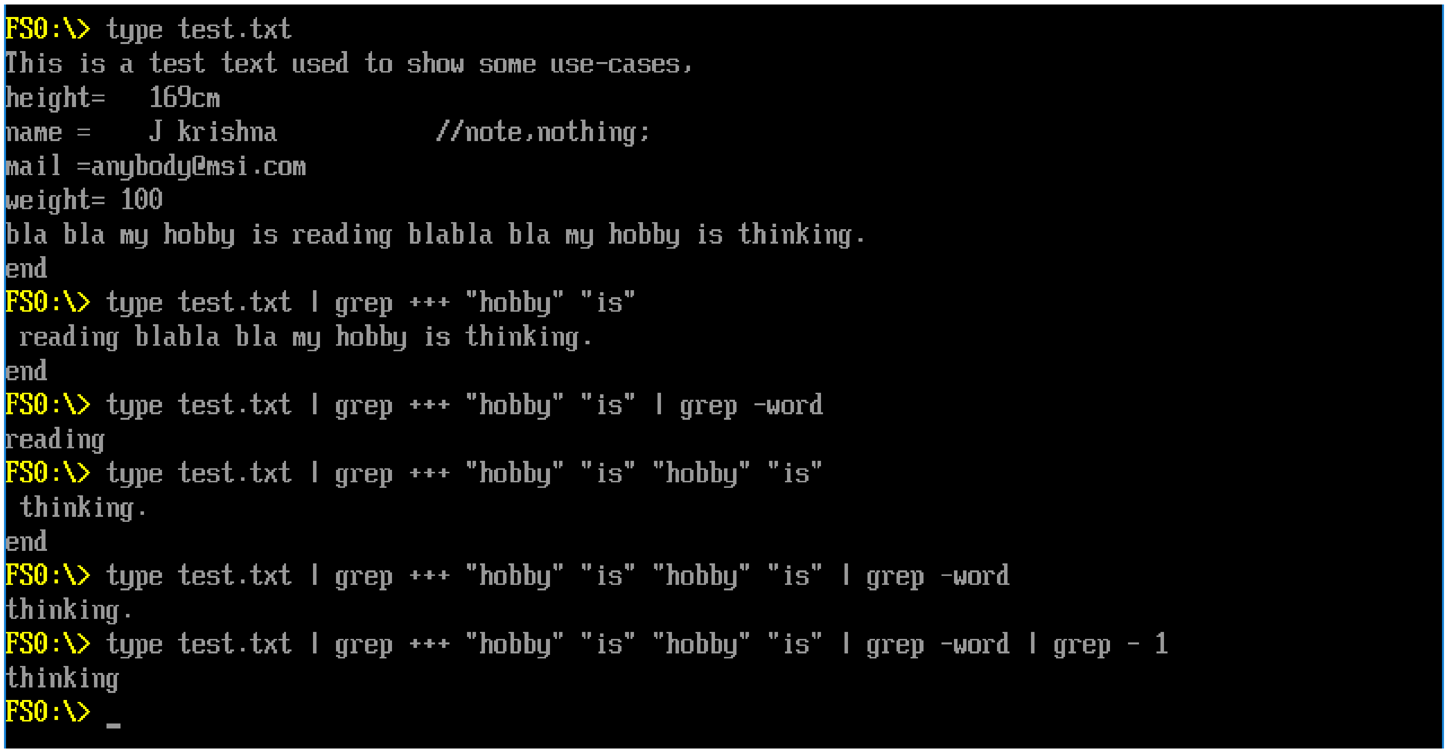


```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ "mail" "=" | grep -++ weight
anybody@msi.com

FS0:\> type test.txt | grep +++ "mail" "=" | grep -++ weight | grep -trim-space
anybody@msi.com
FS0:\> type test.txt | grep +++ "mail" "=" | grep -++ weight | grep -trim-space | grep  - 8
anybody
FS0:\> _
```

## 12，更多的例子 03

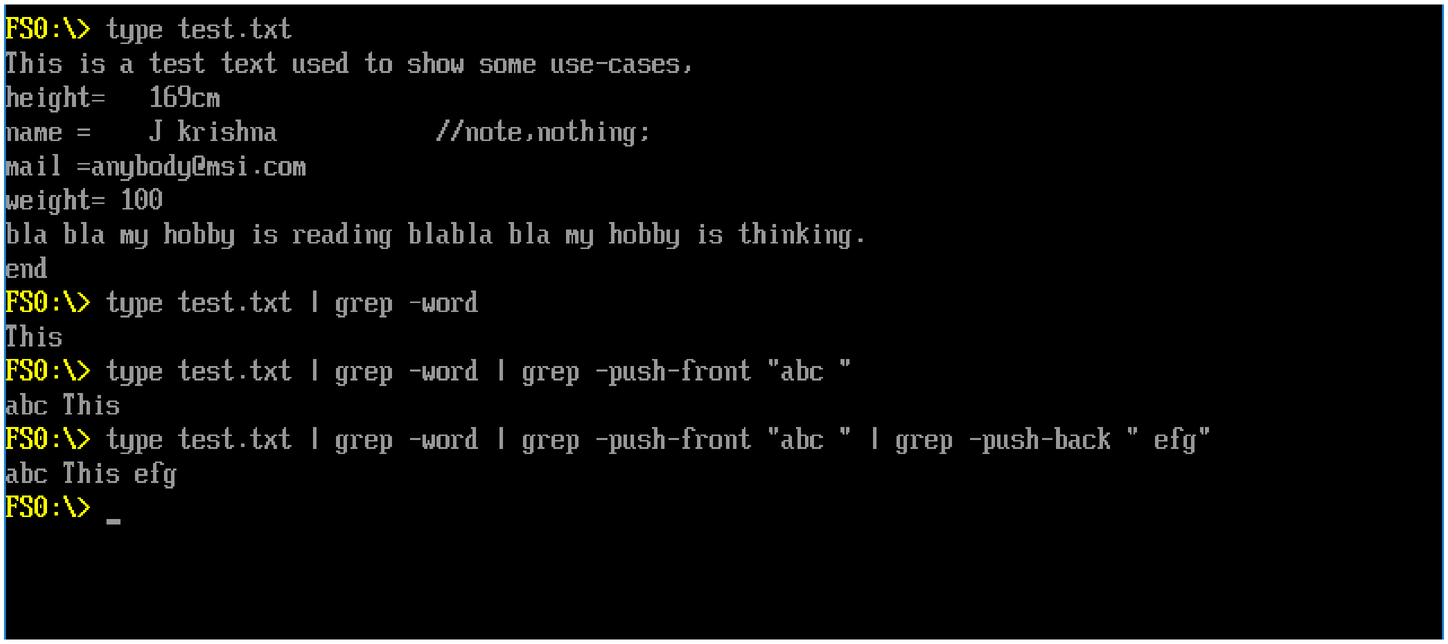


```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +-+ weight
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +-+ weight | grep -line
weight= 100
FS0:\> type test.txt | grep +-+ weight | grep -line | grep +++ "="
 100
FS0:\> type test.txt | grep +-+ weight | grep -line | grep +++ "="  | grep -trim-space
100
FS0:\> _
```



```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ "hobby" "is"
 reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep +++ "hobby" "is" | grep -word
reading
FS0:\> type test.txt | grep +++ "hobby" "is" "hobby" "is"
 thinking.
end
FS0:\> type test.txt | grep +++ "hobby" "is" "hobby" "is" | grep -word
thinking.
FS0:\> type test.txt | grep +++ "hobby" "is" "hobby" "is" | grep -word | grep - 1
thinking
FS0:\> _
```

## 13，追加字符



```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna            //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep -word
This
FS0:\> type test.txt | grep -word | grep -push-front "abc "
abc This
FS0:\> type test.txt | grep -word | grep -push-front "abc " | grep -push-back " efg"
abc This efg
FS0:\> _
```

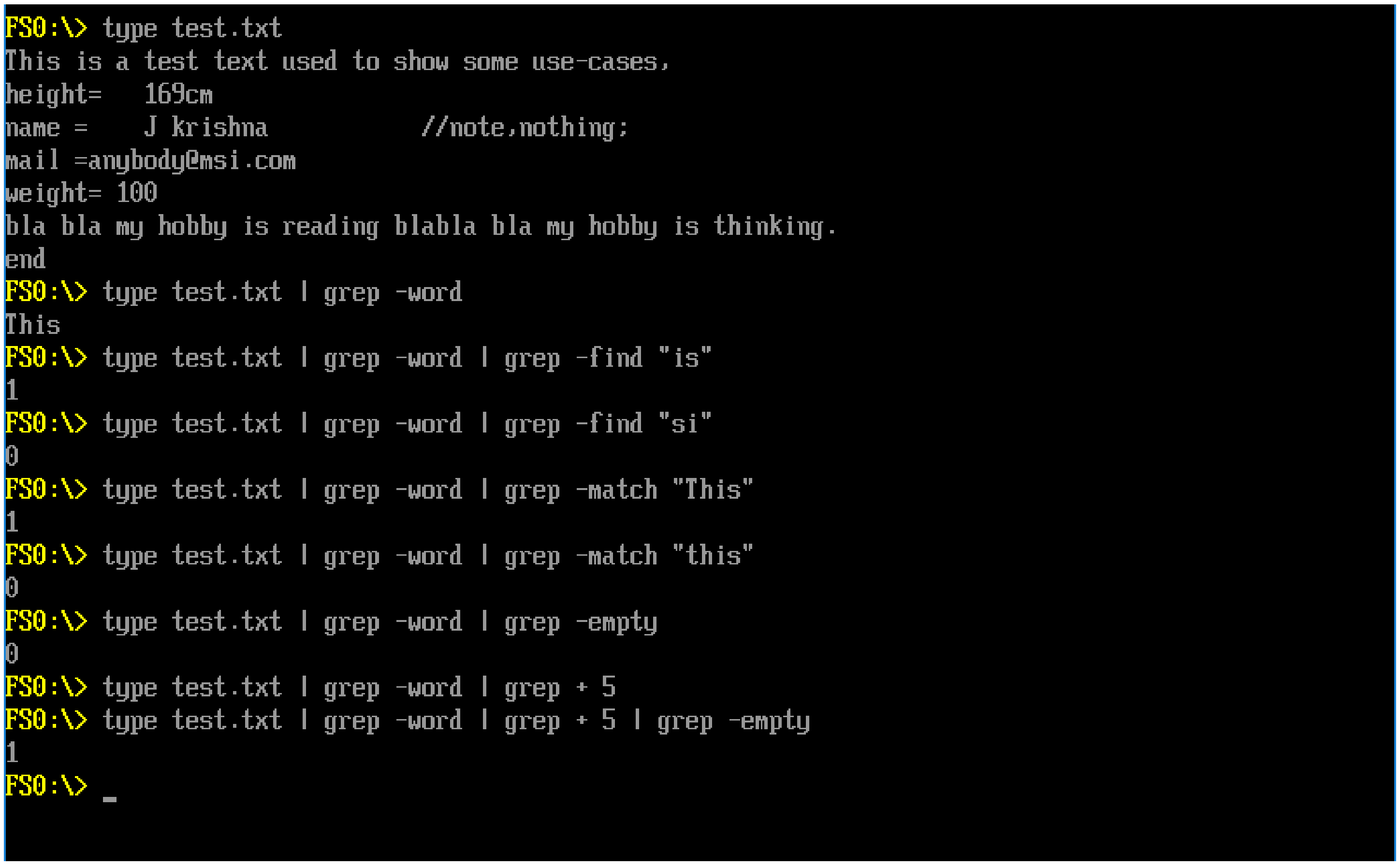
## 14，测试结果

Grep –find，測試 buffer 是否含有指定的字符串。

Grep –equal (或者 Grep –match)， 測試 buffer 是否等于指定的字符串。

Grep –empty，測試 buffer 是否是空的。如果 scope 全部由不可见的字符组成，也算 empty。

测试结果打印输出 0 或 1，**return code 也是相应的值 0 或 1**。



```
FS0:\> type test.txt
This is a test text used to show some use-cases,
height=   169cm
name =    J krishna           //note,nothing;
mail =anybody@msi.com
weight= 100
bla bla my hobby is reading blabla bla my hobby is thinking.
end
FS0:\> type test.txt | grep -word
This
FS0:\> type test.txt | grep -word | grep -find "is"
1
FS0:\> type test.txt | grep -word | grep -find "si"
0
FS0:\> type test.txt | grep -word | grep -match "This"
1
FS0:\> type test.txt | grep -word | grep -match "this"
0
FS0:\> type test.txt | grep -word | grep -empty
0
FS0:\> type test.txt | grep -word | grep + 5
FS0:\> type test.txt | grep -word | grep + 5 | grep -empty
1
FS0:\> _
```

## 三， 注意事項

### 1， 有一些特殊的字符，具有特殊的含意

請參考：

UEFI shell specification 2.2 -> 3.4.1 Special Characters

UEFI shell specification 2.2 -> 3.4.2 Escape Characters

舉個反面例子：grep –find # //這個命令會是無效參數。

正確的操作是：grep –find ^# //這個命令意思是，尋找'#'這個字符。

### 2， 在 UEFI-script 中， 需要在 echo –on 模式下使用

因為在@echo –off 模式下， 有時候管道信息可能會被 UEFI-Shell 自動過濾掉。

### 3， 設計上的限制

工具內部能夠容納的最大 stream 不大于 4MB 個字符。

每個輸入的 string，長度不大于 128 個字符。

內部僅支持 ASCII 對應的寬字符版本(UCS2)。

支持寬字符管道"|"，不支持 ASCII 管道"|a"。

### 4， 不推薦輸入大于 5000 以上的字符

輸入 stream 越長，處理越緩慢。

### 5， Terminal 與 UCS2

1，假設有以下數據:

```
#define WCHAR_CR      0x000d

#define WCHAR_LF      0x000a

CONST CHAR16 buffer[] = {L'A',L'B',L'C', WCHAR_CR,L'O',L'K', WCHAR_CR ,WCHAR_LF };
```

在 windows 文本編輯器上會顯示：

```
ABC

OK
```

在 UEFI-Shell 下，會顯示：

```
OKC
```

原因是 UEFI-Shell 下的字符輸出，遵守一定的終端規範，某些不可見字符具有特殊的含意，這與在 windows 文本編輯器中的含意是不一樣的。

在終端上，WCHAR_CR 會被解釋成“光標移動到屏幕最左端”，當輸出這個 buffer 的時候，

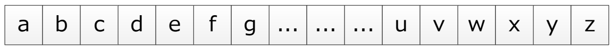
輸出”ABC”，遇到 WCHAR_CR 光標會再移到最左端，即字符 A 上，再輸出”OK”，原來的信息就被覆蓋了。

這造成 grep 在某些情況下，顯示輸出的信息似乎不太正常，是由于提供給 grep 的 buffer 有這樣的類似問題。

# 四， Concept design

## 1, buffer and iterators

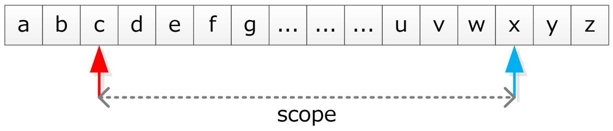
1.1, input is a stream.



1.2, scope is between a front iterator and a reverse iterator, default scope is the full-stream.

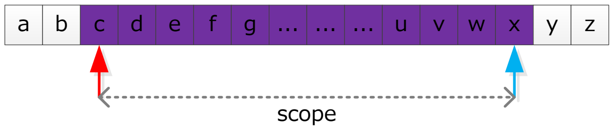
front iterator can only move to right;

reverse iterator can only move to left.



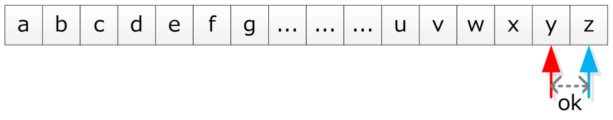
1.3, **buffer** (used for output) is defined by the scope.

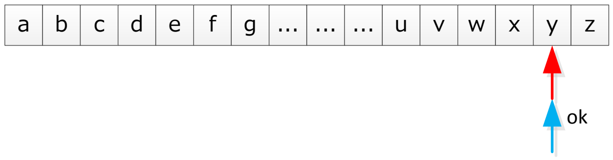
In other words, buffer or scope is defined by the front iterator and the reverse iterator.

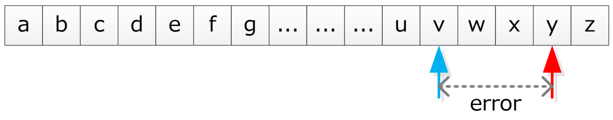


1.4, the front iterator and the reverse iterator can do movement in the stream, but they cannot exceed each other.

For example:



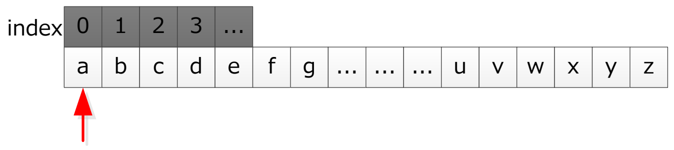




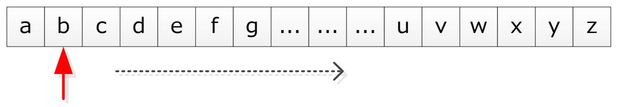
(備注: 前后指針可能相遇，但不能越過彼此)

## 2, Front iterator operations

2.1, there is a **front iterator**, point to a stream's begin default; there is an index, relative to this front pointer.



2.2, the front iterator can only move to right.

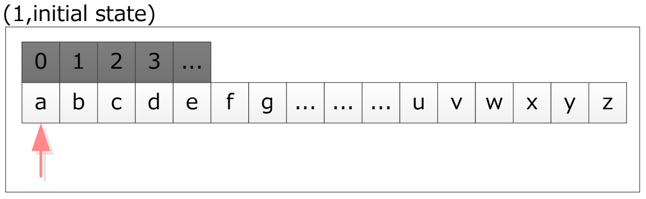


2.3, about a right move command.

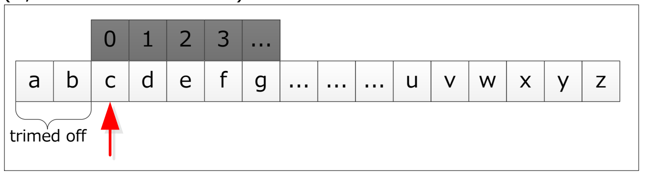
// pseudo command:

// -front_iterator -move-number 2

(1,initial state)



(2,move finished state)

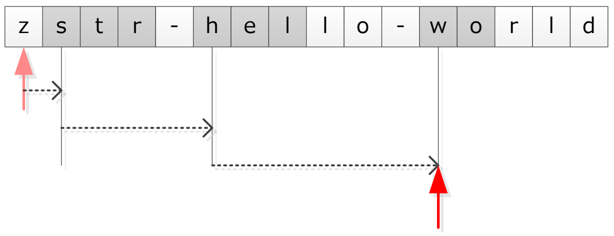


It looks like the front part of the stream is "ab", which will be trimmed off; just the rest part which pointed by front-iterator "cdefg…" is valid;

2.4, you can move front iterator by string(s)

// pseudo command:

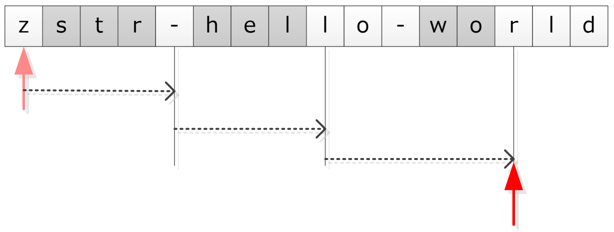
-front_iterator -move “str” ”hel” “wo”



2.5, you can move-skip front iterator by string(s).

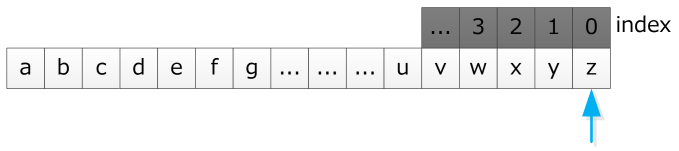
// pseudo command:

// -front_iterator -move-skip “str” ”hel” “wo”

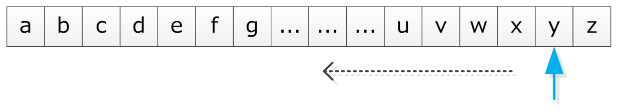


## 3, Reverse iterator operations

3.1, there is a **reverse iterator**, point to a stream's end; there is an index, relative to this reverse pointer.



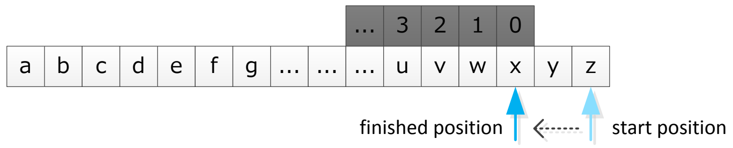
3.2, the reverse iterator can only move to left.



3.3, about a left move command.

// pseudo command:

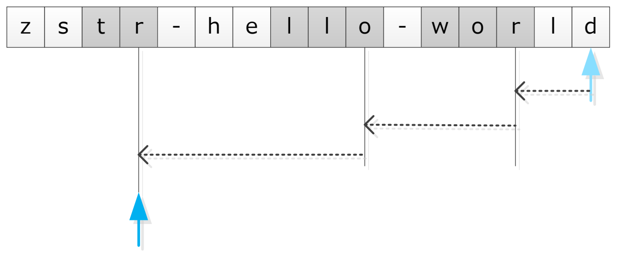
// -reverse_iterator -move-number 2



3.4, you can move reverse iterator by string(s)

// pseudo command:

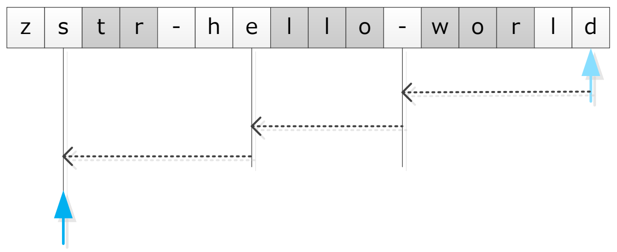
// -reverse_iterator -move "wor" "llo" "tr"



3.5, you can move-skip reverse iterator by string(s).

// pseudo command:

// -reverse_iterator -move-skip "wor" "llo" "tr"

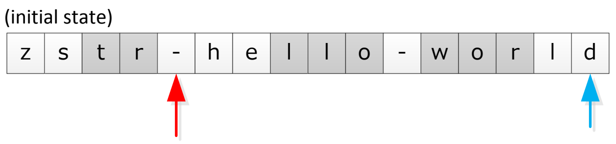


## 4, force & non-force

// one pseudo command:

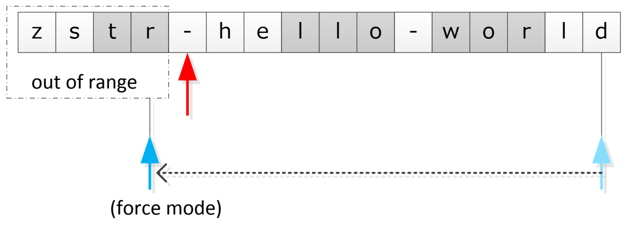
// -reverse_iterator -move-skip “wer”

If the stream doesn’t contain the string “wer”,



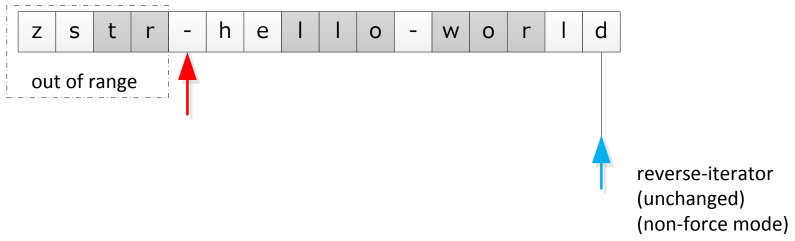
4.1, in force mode

the reverse-iterator will move-left to find the string “wer”, the string is not exist in this sample, so the reverse-iterator will exceed front-iterator, the output buffer will be NULL, see below picture.



4.2, in non-force mode

The reverse-iterator will try to find left string “wer”, it will not move if cannot find the string “wer”, so the reverse-iterator will remain unchanged in the picture.



4.3, use-case

For example follow string has a key=value, and a description:

```
    name = Li Lei     //description;
```

The description may or may not exist, in order to grep the value "Li Lei", we can using follow pseudo command:

```
        … | grep -line | grep –front-move-skip "=" | grep -reverse-move-skip-non-force "//" | grep –trim-front-back-space
```

//end.

# 五，Command line reference

## 1, command line.

Usage:

```
  grep.efi [option]
```

Options:

```
  [+|-] [n]                //move a iterator by a number.
                           1,[+|-]:front or reverse (iterator).
                           2,[n]:a number to move.
  [+|-][+|-]{+|-} [s]+     //move a iterator by string(s).
                           1,[+|-]:front or reverse;  2,[+|-]:skip or non-skip.
                           3,{+|-}:force or non-force;4,[s]+:string(s) reference.

  -char                    //get first visible char at front-iterator.
  -word                    //get first visible word at front-iterator.
  -line                    //get first visible line at front-iterator.
  -trim-space              //trim off scope's front and back invisible chars.

  -push-front [s]          //add a string at front of scope.
  -push-back  [s]          //add a string at back of scope.

  -empty                   //test scope is empty or not.
  -equal [s]               //test scope if it equals a string or not.
  -find  [s]               //test scope if it contains a string or not.
```

## 2, 关键点

**grep.efi [+|-] [n]**，用来移动前指针或后指针。

**[+|-]** 选择**前指针**或**后指针**，这是一个必选项。

**[n]**代表移动的步数，0<=n<=size of(scope)， 如果 n>size of (scope),输出为空，这是一个必选项。

前指针只会往后移；后指针只会往前移；前后指針可以相遇，不要越過彼此，越過彼此將使輸出為空。

**grep.efi [+|-][+|-]{+|-} [s]+**，用来移动前指针或后指针，根据提供的字符串。

**[+|-]** 选择前指针或后指针，这是一个必选项。

**[+|-]** 选择跳过或不跳过提供的字符串，这是一个必选项。

**{+|-}** 选择强制或非强制模式，这是一个**可选项**。默认为强制模式。

强制模式下，没有找到提供的字符串，将使输出为空；非强制模式下，没有找到提供的字符串，输出会保持操作失败之前的样子。

例如有某个命令：

… | grep +++ aaa bbb ccc | …

如果找到 aaa， 没有找到 bbb，操作中止，这个操作将会输出为空。

如果此命令换成：

… | grep ++- aaa bbb ccc | …

如果找到 aaa， 没有找到 bbb，操作中止，这个操作只会跳过 aaa 字符串，输出跳过 aaa 的部分。

**[s]+** 一个或多个字符串，多个字符串以空格作分隔。如果字符串本身包含空格，请以双引号包裹。

## 3, 别名

有一些命令，带有别名。具体如下所示：

| 參數 | 别名 1 | 别名 2 | 注釋 |
|---|---|---|---|
| -help | -h | --help,help,?,-? | 幫助 |
| | | | |
| + | | -move-front | 移動前指針 n 步 |
| - | | -move-reverse | 移動后指針 n 步 |
| | | | |
| +-+ | +- | -front-non-skip-force | 移動前指針 by 字符串 |
| +-- | | -front-non-skip-non-force | |
| +++ | ++ | -front-skip-force | |
| ++- | | -front-skip-non-force | |
| | | | |
| --+ | -- | -reverse-non-skip-force | 移動后指針 by 字符串 |
| --- | | -reverse-non-skip-non-force | |
| -++ | -+ | -reverse-skip-force | |
| -+- | | -reverse-skip-non-force | |
| | | | |
| -trim-front-space | -tfs | | 剪掉前端不可見字符 |
| -trim-back-space | -tbs | | 剪掉后端不可見字符 |
| -trim-space | -ts | | 剪掉前&后端不可見字符 |
| | | | |
| -push-front | -pf | | 字符追加 |
| -push-back | -pb | | |
| | | | |
| -char | -c | | 取一個字符(執行-trim-space) |
| -word | -w | | 取一個詞(執行-trim-space) |
| -line | -l | | 取一行(不執行-trim-space) |
| | | | |
| -size | -si | | 輸出 scope’s 字符數 |
| | | | |
| -empty | -em | | 測試結果 |
| -equal | -eq | -match | |
| -find | -fi | | |

關于操作“**grep.efi [+/-] [n]**”，可以縮寫為“**grep.efi [+/-] [n]**”，（即**[+/-]**和**[n]**之間沒有空格）。

有一些保留的操作項，說明如下：

“**grep.efi –size**”，計算 grep->buffer 字符數量。會計入不可見字符，結果有時候“看上去”不準確。

例如“hello”,實際上的 size 是 5。

例如“ hello”,實際上的 size 是 6,因為前面有一個空格。

例如“hello ”,實際上的 size 是 6,因為后面有一個空格。

例如“ hello ”,實際上的 size 是 7,因為前后都有一個空格。

如果要計算“ hello world ”不包含最前面的空格，不包含最后面的空格，之后的字符數，可以這樣使用:

```
… | grep –trim-front-space | grep –trim-back-space | grep –size
```

或者這樣使用:

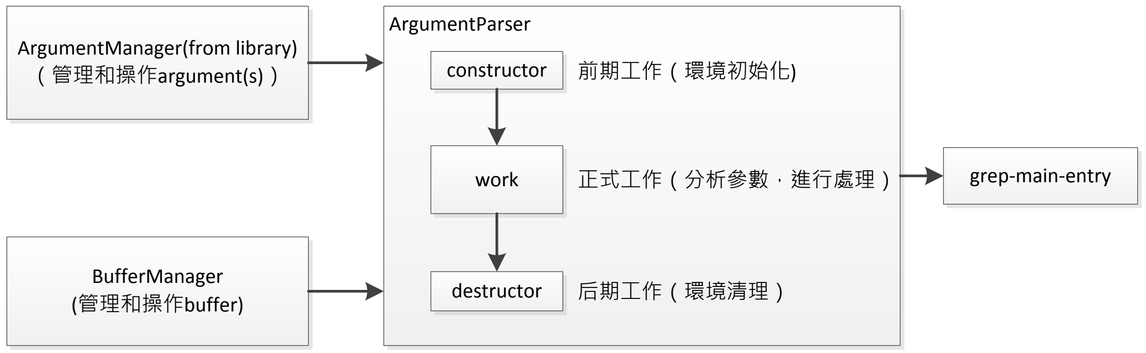
```
… | grep –trim-space | grep –size
```

結果都會是 11。

```
… | grep –trim-space | grep –size | grep –eq 11       //test pass.
```

“**grep.efi -trim-front-space**”和“**grep.efi -trim-back-space**”，這兩個功能之和為“**grep.efi -trim-space**”。

## 六，Framework


ArgumentManager(from library)
（管理和操作argument(s)）
BufferManager
(管理和操作buffer)
ArgumentParser
constructor
前期工作（環境初始化)
work
正式工作（分析參數，進行處理）
destructor
后期工作（環境清理）
grep-main-entry