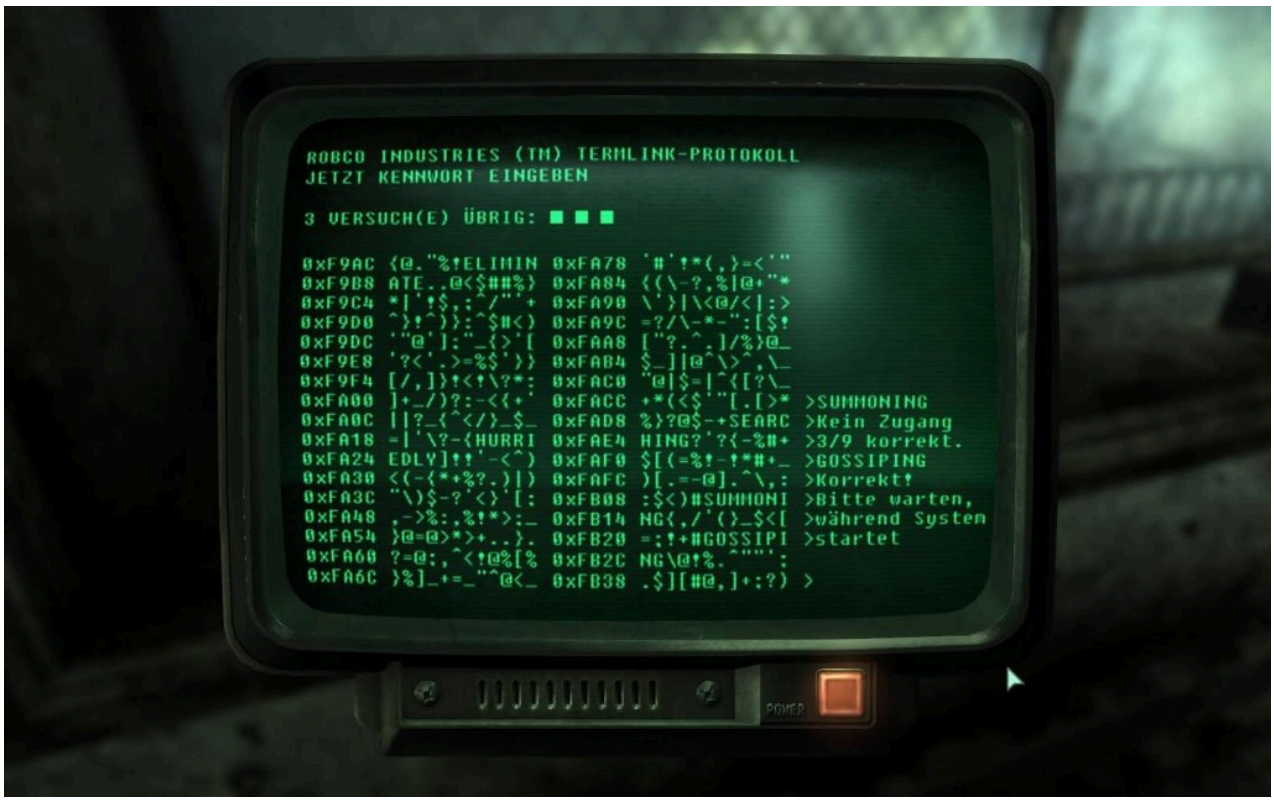
Linux 系統 xargs 指令範例與教學

2014/04/29



這裡提供 xargs 這個 Linux 指令的使用教學，並搜集一些常用的範例程式以供參考。

在 UNIX/Linux 系統中， xargs 這個指令跟其他的指令結合之後，將會變得非常有用，這裡我們整理了一些常見的 xargs 使用範例與教學，透過這些簡單的範例可以很快的了解 xargs 的各種使用方式。

雖然這裡的範例都很簡單，但是當你了解 xargs 的用法之後，你會發現

`xargs` 其實可以拿來處理各式各樣的問題，而且非常好用，尤其是對於伺服器的系統管理者而言，在管理大量的檔案與目錄結構時，有這樣的工具會方便很多。

xargs 基本用法

`xargs` 這個指令會標準輸入（standard input）讀取資料，並以空白字元或換行作為分隔，將輸入的資料切割成多個字串，並將這些字串當成指定指令（預設為 `/bin/echo`）執行時的參數。如果直接執行

```
xargs
```

接著在終端機中輸入

```
arg1 arg2
arg3
```

結束時按下 `Ctrl + d`，這時候 `xargs` 就會把輸入的兩行資料切割成 3 個字串（分別為 `arg1`、`arg2` 與 `arg3`），接著把這三個字串當作指定指令的參數，而這裡我們沒有指定任何指令，所以預設會使用 `/bin/echo`，直接把三個字串輸出至終端機中。

```
arg1 arg2 arg3
```

這裡雖然輸入的資料中有包含一個換行字元（`\n`），不過因為 `xargs` 預設不會保留換行字元，所以整個輸出就變成一整行。這個效果相當於

```
/bin/echo arg1 arg2 arg3
```

`xargs` 在讀取標準輸入的資料時，會自動忽略空白行，多餘的空白或是 tab 字元也會自動被忽略。

如果要指定 xargs 所要執行的指令，可以直接將指令的名稱放在所有 xargs 的參數之後，例如

```
xargs cat
arg1 arg2
arg3
```

就相當於

```
cat arg1 arg2 arg3
```

分隔字元

如果要指定 xargs 在讀取標準輸入時所使用的分隔字元，可以使用 -d 這個參數。例如：

```
xargs -d\n
arg1 arg2
arg3
```

輸出為

```
arg1 arg2
arg3
```

這裡當我們使用 -d 這個參數指定分隔字元時， xargs 就會將換行字元保留下來，所以輸出會分為兩行。

參數個數上限

預設的狀況下， xargs 會把從標準輸入的資料所分割出來的字串，一次全部都放進指定指令參數中，例如：

```
echo a b c d e f | xargs
```

就相當於

```
echo a b c d e f
```

所以輸出為

```
a b c d e f
```

如果我們不想讓所有的參數都放進一個指令中指令中執行，可以使用 `-n` 參數來指定每一次執行指令所使用的參數個數上限值，例如

```
echo a b c d e f | xargs -n 3
```

就相當於

```
echo a b c
echo d e f
```

所以輸出就變為

```
a b c
d e f
```

如果指定上限為 2 ：

```
echo a b c d e f | xargs -n 2
```

輸出就會變成

```
a b
c d
e f
```

執行前的確認

如果你對於 xargs 的使用方式不是很熟悉，或是需要以 root 權限執行一些不容出錯的指令，可以加上 -p 參數，讓指令在實際執行指令之前可以先進行確認的動作。例如：

```
echo a b c d e f | xargs -p -n 3
```

則在每一行指令執行前，都會先確認，如果要執行就輸入 y，若不執行則輸入 n，假設每一個指令我們都輸入 y 讓它執行，則整個輸出就會變成這樣：

```
/bin/echo a b c ?...y
/bin/echo d e f ?...a b c
y
d e f
```

這裡因為 xargs 的確認訊息與 echo 指令的輸出混在一起，所以看起來有點奇怪。

如果所有的指令進行確認時，我們都輸入 n，這樣就不會執行任何的 echo 指令，輸出就會像這樣：

```
/bin/echo a b c ?...n
/bin/echo d e f ?...n
/bin/echo ?...n
```

忽略空字串參數

大家應該有注意到上面的例子當我們使用 -p 參數時，如果所有的指令都輸入 n 跳過不執行時，最後還會出現一個沒有任何參數的 echo 指令，如果想要避免以這種空字串作為參數來執行指令，可以加上 -r 參數：

```
echo a b c d e f | xargs -p -n 3 -r
```

```
/bin/echo a b c ?...n
/bin/echo d e f ?...n
```

如果標準輸入為空字串時，`xargs` 預設還是會執行一次 `echo` 指令：

```
xargs -p
```

```
/bin/echo ?...n
```

這種狀況同樣可以加上 `-r` 參數：

```
xargs -p -r
```

顯示執行的指令

使用 `-t` 參數可以讓 `xargs` 在執行指令之前先顯示要執行的指令，這樣可以讓使用者知道 `xargs` 執行了哪些指令。

```
xargs -t
abcd
```

按下 `Ctrl + d` 之後，`xargs` 就會先輸出要執行的指令內容，接著才是實際指令執行的輸出，最後得到輸出為

```
/bin/echo abcd
abcd
```

結合 xargs 與 find 指令

`xargs` 本身的功能並不多，但是他跟其他的 Linux 指令一起搭配使用時，功能就會顯得很強大。

與 `find` 指令合在一起使用是 `xargs` 的一項非常重要的功能，它可以讓你找尋特定的檔案，並且進行特定的處理動作。

假設我們有一些資料夾，其中包含各式各樣的檔案，以 `tree` 指令查看目錄結構會呈現

```
.
├── folder1
│   ├── file1.c
│   ├── file1.h
│   ├── file2.c
│   └── file2.h
└── folder2
    ├── file3.c
    └── file3.h

2 directories, 6 files
```

假設我們想要找出目前路徑之下的所有 `.c` 檔案，並且將其刪除，就可以使用

```
find . -name "*.c" | xargs rm -f
```

這裡我們將 `xargs` 要執行的指令指定為 `rm -f`，讓 `find` 所找到的 `.c` 檔案都當成 `rm -f` 的參數，所以其指令的效果就相當於

```
rm -f ./folder2/file3.c ./folder1/file2.c ./folder1/file1.c
```

如果你對於指令的操作還不是很熟悉，也可以配合上面介紹的 `-p` 參數，在執行前確認一下

```
find . -name "*.c" | xargs -p rm -f
```

刪除 `.c` 檔案之後，以 `tree` 指令查看整個目錄樹就會變成

```
.
├── folder1
│   ├── file1.h
│   └── file2.h
└── folder2
    └── file3.h

2 directories, 3 files
```

這裡因為是示範用的例子，所以並沒有建立太多的檔案與目錄結構，在實際的應用上，當整個目錄結構很複雜、檔案又很多的時候，使用這樣的方式就會非常有效率。

包含空白的檔案名稱

假設我們的目錄中有一些檔名包含空白字元：

```
touch "G T Wang.c"
```

在這種狀況下如果用上面 `find` 與 `xargs` 的方式會無法將其刪除，原因在於當我們執行

```
find . -name "*.c" | xargs rm -f
```

的時候， `xargs` 所產生的指令為

```
rm -f ./G T Wang.c
```

因為檔名包含空白，所以這會會造成 `rm` 指令無法正確刪除該檔案。

這個時候我們可以將 `find` 指令加上 `-print0` 參數，另外將 `xargs` 指令加上 `-0` 參數，改成這樣

```
find . -name "*.c" -print0 | xargs -0 rm -rf
```

如此一來，即可正確處理包含空白字元的檔案名稱。

命令列長度的限制

使用 `--show-limits` 參數可以查看系統對於命令列長度的限制，這些限制會跟 `xargs` 的運作情況有關，如果要處理很大量的資料時，這些限制要注意一下。

```
xargs --show-limits
```

輸出為

```
Your environment variables take up 2022 bytes
POSIX upper limit on argument length (this system): 2093082
POSIX smallest allowable upper limit on argument length (all systems): 4096
Maximum length of command we could actually use: 2091060
Size of command buffer we are actually using: 131072

Execution of xargs will continue now, and it will try to read its input and run
Warning: /bin/echo will be run at least once.  If you do not want that to happen
```

結合 xargs 與 grep 指令

`xargs` 與 `grep` 兩個指令的合併也是一個很常見的使用方式，它可以讓你找尋特定檔案之後，進而搜尋檔案的內容。

假設我們要在所有的 `.c` 檔案中搜尋 `stdlib.h` 這個字串，就可以使用

```
find . -name '*.c' | xargs grep 'stdlib.h'
```

我直接拿 VTK 的原始碼來測試，輸出會像這樣

```
./CMake/TestOggTheoraSubsampling.c:#include <stdlib.h>
./Wrapping/Tools/lex.yy.c:#include <stdlib.h>
./Wrapping/Tools/vtkWrapPythonInit.c:#include <stdlib.h>
./Wrapping/Tools/vtkWrapTcl.c:#include <stdlib.h>
./Wrapping/Tools/vtkWrapTclInit.c:#include <stdlib.h>
./Wrapping/Tools/vtkParsePreprocess.c:#include <stdlib.h>
./Wrapping/Tools/vtkWrapText.c:#include <stdlib.h>
./Wrapping/Tools/vtkParseData.c:#include <stdlib.h>
./Wrapping/Tools/vtkParseHierarchy.c:#include <stdlib.h>
./Wrapping/Tools/vtkParseMain.c:#include <stdlib.h>
[略]
```

這個對於程式設計師在看一堆專案原始碼的時候特別有用。

參考資料：The Geek Stuff

LINUX

指令

G. T. Wang
個人使用 Linux 經驗長達十餘年，樂於分享各種自由軟體技術與實作文章。

2 留言

coco

謝謝你，好詳細的教學，我終於懂了xargs

2019/12/11

afat6666

謝謝分享, 很清楚好懂!

2020/05/03

Comments are Closed

搜尋

分類

ARDUINO (5) BEAGLEBONE BLACK (1)
DIY (54) LINUX (317) MACOS (33)
OCTAVE (15) PERL (12) R (47)
WINDOWS (98) WORDPRESS (16)
個人 (15) 免費 (35) 兒童 (30)
實用工具 (85) 手機 (13) 技巧 (45)
有趣 (99) 樹莓派 (57) 物聯網 (55)
玄學 (11) 生活 (210) 程式設計 (137)
統計學 (8) 網頁空間 (36)
網頁開發 (128) 虛擬化 (7) 農業 (42)
遊戲 (9) 開箱 (133) 雲端 (4)

宗教

如何戒邪淫、遠離婚外情

戒淫寶典：《壽康寶鑑》白話有聲書

公益

家扶基金會

Yahoo 奇摩公益

智邦公益網

igiving 公益網

社團法人新竹縣愛心物資集發協會