如何變更 xterm 的主題

Ric Lister, ric@giccs.georgetown.edu
譯者: 曾元佑 *yytseng@ms16.hinet.net*
v2.0, 1999 年十月 27 日

本文解釋如何使用溢位序列以動態變更 xterm 的視窗及圖示. 範例中, 包含數種命令解譯器語法. 而附錄亦列出給其他類型終端機使用的溢位序列.

目錄

1 可以在哪找到這份文件

這份文件現已是 *Linux HOWTO Index* <http://sunsite.unc.edu/LDP/HOWTO/> 的一部份, 亦可以在這找到 <http://sunsite.unc.edu/LDP/HOWTO/mini/Xterm-Title.html>.

最新版本的各種文件格式都可以在這找到 <http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/>.

本文取代了原本由 Winfried Trümper 所寫的 howto.

2 靜態主題

任一種終端機 xterm, color-xterm 或 rxvt 都可以藉由 -T 及 -n 的參數設定來設定靜態主題:

```
xterm -T "My XTerm's Title" -n "My XTerm's Icon Title"
```

3 動態主題

許多人發現這在顯示一些動態資訊時相當有用, 比如 使用者所登入的主機名稱, 現行的工作目錄, 等等.

3.1 xterm 溢位序列

一個已在執行的 xterm 的視窗及圖示的主題都可以透過溢位序列來變更. 下列列出跟這個設定有關的序列:

- ESC]0;stringBEL – 設定圖示及視窗的主題爲 **string**
- ESC]1;stringBEL – 設定圖示名爲 **string**
- ESC]2;stringBEL – 設定視窗主題爲 **string**

在這裡 ESC 是指 **escape** 字元 (\033), 而BEL 是指 **bell** 字元 (\007).

在 xterm 裡頭就可以使得視窗及圖示的主題變更.

注意: 這些序列可以應用到大部份 xterm 衍生的程式, 比如 nxterm, color-xterm 及 rxvt. 其他的 終端機類型大都使用不一樣的序列; 參考附錄所列舉的範例. 如要參考 xterm 所有的溢位序列參考這個檔案 *ctlseq2.txt* <http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/ctlseq2.txt>, 這會隨著 xterm 的套件發行, 或 *xterm.seq* <http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/xterm.seq>, 隨著 *rxvt* <http://www.rxvt.org/> 套件發行.

3.2 印出這些序列

有些資訊在命令解譯器的生命週期中是自始至終都不變的, 比如 主機名稱 及 使用者名稱, 那麼在命令解譯器的初始啓動檔 (rc file) 用 echo 指令印出這些字串就夠了:

```
echo -n "\033]0;${USER}@${HOST}\007"
```

應該會產生像 username@hostname 這樣的主題, 假設命令解譯器的變數 $USER 及 $HOST 都已設定正確的話. echo 所需的選項依命令解譯器使用的類型而有所不同 (參考下面的說明).

有些資訊在命令解譯器的生命週期中是一直在變的, 比如 現行工作目錄, 這些溢位資訊就必須在每次提示字元變化時隨著改變. 這下子字串就會在每次你輸入命令時更新, 而且你還可以追蹤保留 現行目錄, 使用者名, 主機名, 等資訊. 部份命令解譯器提供這類用途的特殊功能, 有些則沒有. 而我們就必須直接插入主題溢位資訊到題示字串中. 這會在下一節中說明

4 各種命令解譯器的範例

以下我們提功一些範例給常見的命令解譯器使用. 我們先從 zsh 開始, 他可以提供許多靈巧的機制使得我們很容易地完成我們所要的工作. 然後我們再進展到難度較高的範例.

在所有的範例中我們都測試了 $TERM 環境變數. 以確定我們只需把這個溢位資訊送到 xterm. 我們對 $TERM=xterm* 做測試; 萬用字元是因爲些許的差異性 (比如 rxvt) 會設定 $TERM=xterm-color.

我們要對 C 命令解譯器 (C Shell) 族系作特別的註解, 比如 tcsh 及 csh. 在 C 命令解譯器, 使用到未定義的變數就算是致命性的錯誤. 因此, 在測試變數 $TERM 之前, 先確定其是否存在否則會使在非交談模式下工作的命令解譯器停擺. 要達到這個目的你必須把部份東西用下面這種樣式包起來:

```
if ($?TERM) then
    ...
endif
```

(我們是不主張用 C 命令解譯器 的理由之一. 參考 Csh Programming Considered Harmful <http://language.perl.com/versus/csh.whynot> 有很多有用的討論文章).

以下的範例可以被用來插入到合適的命令解譯器的初始啓動檔; 即 那個在交談式命令解譯器啓動時會去讀的那個檔. 在大部份的情況它被稱爲 .命令解譯器名rc (如 .zshrc, .tcshrc, 等).

4.1 zsh

zsh 提供部份功能與延伸功能, 而我們將用到:

precmd ()	一個在提示命令前必執行的功能含式
chpwd ()	一個在目錄有所變化時會執行的功能含式
\e	溢位字元 (ESC)
\a	bell 的溢位字元 (BEL)
%n	被解釋爲 $USERNAME
%m	被解釋爲主機名稱在第一個 '.' 之前的部份
%~	被解釋爲目錄, 以 '~' 取代 $HOME

更多的延伸功能: 參考 zshmisc 使用手冊.

因此, 以下將設定 xterm 的主題爲 "*username@hostname: directory*":

```
case $TERM in
    xterm*)
        precmd () {print -Pn "\e]0;%n@%m: %~\a"}
        ;;
esac
```

這也可以用 chpwd() 取代 precmd() 來達成. print 內建的工作是跟 echo 一樣, 但可以讓我們去存取 % 命令提示溢位資訊.

4.2 tcsh

tcsh 的部份功能含式與延伸功能與 zsh 相同:

precmd ()	一個在提示命令前必執行的功能含式
cwdcmd ()	一個在目錄有所變化時會執行的功能含式
%n	被解釋爲 $USERNAME
%m	被解釋爲主機名稱
%~	被解釋爲目錄, 以 '~' 取代 $HOME
%#	對正常的使用者解釋爲 '>', 而超級使用者則爲 '#'
%{...%}	引入一個字串爲連續的溢位序列

不幸的, 並沒有與 zsh 的 print 相同功能的指令, 能讓我們在主題字串中使用提示溢位資訊, 因此我們唯一能使用的命令解譯器變數 (在 ~/.tcshrc):

```
switch ($TERM)
    case "xterm*":
        alias precmd 'echo -n "\033]0;${HOST}:$cwd\007"'
        breaksw
endsw
```

然而, 這會用目錄完整的路徑取代使用 ~. 改把這些字串插入題示字元:

```
switch ($TERM)
    case "xterm*":
        set prompt="%{\033]0;%n@%m:%~\007%}tcsh%# "
        breaksw
    default:
        set prompt="tcsh%# "
        breaksw
endsw
```

這會設定 "tcsh% " 的提示字元, 而 xterm 的主題及圖示爲 "*username@hostname: directory*". 要注意 "%{... (且不能被放在 提示的最後一項: 參考 tcsh 使用手冊說明會更詳細).

4.3 bash

bash 支援變數 $PROMPT_COMMAND 內含一個指令在提示字元之前執行. 這個範例將設定主題爲 username@hostname: directory:

```
PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME}: ${PWD}\007"'
```

在這的 \033 是代表字元 ESC, 而 \007 則是 BEL.

記住引號在這相當重要: 會被解釋的變數是放在 "...", 而不是 '...'. 因此 $PROMPT_COMMAND 是被設定爲一個不被解釋的值, 但變數在 "..." 裏如果有用到 $PROMPT_COMMAND 則會被解釋.

然而, $PWD 產生完整的路徑. 如果我們要使用 ~ 這種速記, 則我們要把溢位字串嵌入到提示字元, 這會讓我們可以利用命令解譯器所提供的命令列延伸功能:

\u	被解釋爲 $USERNAME
\h	被解釋爲主機名稱在第一個 '.' 之前的部份
\w	被解釋爲目錄, 以 '~' 取代 $HOME
\$	對正常的使用者解釋爲 '>', 而超級使用者則爲 '#'
\[...\]	嵌入非列印字元溢位序列

因此, 以下將產生 bash$ 的提示字元, 並設定 xterm 的主題爲 username@hostname: directory:

```
case $TERM in
    xterm*)
        PS1="\[\033]0;\u@\h: \w\007\]bash\\$ "
        ;;
    *)
        PS1="bash\\$ "
        ;;
esac
```

記住 \[...\] 的使用, 在計算題示長度時, 將告訴 bash 忽略掉非列印的控制字元. 否則行編輯指令將會在移動游標時搞亂掉.

4.4 ksh

ksh 幾乎不提供這種方式的功能函式與延伸功能, 因此我們必須插入溢位字串到提示字元中, 使他能動態更新 這個範例將產生主題爲 username@hostname: directory 而and a prompt of ksh$.

```
case $TERM in
    xterm*)
        HOST=`hostname`
        PS1='^[]0;${USER}@${HOST}: ${PWD}^Gksh$ '
        ;;
    *)
        PS1='ksh$ '
        ;;
esac
```

而，$PWD 會產生完整的目錄路徑．我們可以 用 ${...##...} 的方式移去 $HOME/ 的字首． 我們也可以用 ${...%

```
HOST=`hostname`
HOST=${HOST%%.*}
PS1='^[]0;${USER}@${HOST}: ${PWD##${HOME}/}^Gksh$ '
```

記住 ^[及 ^G 在命令列字串是單一字元 的 ESC 及 BEL (在 emacs 的環境下可以用 C-q ESC 及 C-q C-g 輸入).

4.5 csh

要在 csh 完成同樣的事眞的有點困難，而我們用下面的方式來解決問題：

```
switch ($TERM)
    case "xterm*":
        set host=`hostname`
        alias cd 'cd \!*; echo -n "^[]0;${user}@${host}: ${cwd}^Gcsh% "'
        breaksw
    default:
        set prompt='csh% '
        breaksw
endsw
```

麻煩的地方是我們要把 cd 這個指令化身成可送出溢位字串的功能． 記住 ^[及 ^G 在命令列字串是單一字元的 ESC 及 BEL (在 emacs 的環境下可以用 C-q ESC 及 C-q C-g 輸入).

記住： 在部份的系統 hostname -s 可能會取得較短的 hostname 而不是完整的全名． 部份使用者在有符號連結的目錄應該會發現 `pwd` (括起來以確定執行的是 pwd 指令) 可以得到比 $cwd 更精確的路徑名．

5 顯示正在執行的工作名稱

通常使用者會啓動一個一直在前景執行的工作如 top，或 一個編輯器，一個 email 用户端，等等，並希望這個工作的名稱被顯示在 xterm 的主題上． 這是個很棘手的問題且只能在 zsh 環境下可以輕鬆完成．

5.1 zsh

zsh 爲這種需求提供了一個很理想的內建功能：

preexec()	一個在命令執行前一定會叫用的功能函式名
$*,$1,...	傳送到 preexec() 的參數

因此，我們可以用下面的方式把工作名稱放到主題上：

```
case $TERM in
    xterm*)
      preexec () {
        print -Pn "\e]0;$*\a"
      }
    ;;
esac
```

記住: preexec() 功能函式在 zsh 3.1.2 版出現, 如果你用的版本較舊那就需要更新一下.

5.2 其他命令解譯器s

對其它缺少 preexec() 函式的命令解譯器, 就不是那麼簡單了. 如果任何人有範例可以完成同樣的工作請把它 email 給作者.

6 附錄: 其它型式終端機的溢位序列

許多新式的終端機都衍生自 xterm 或 rxvt 並且支援我們所使用的溢位序列. 某些專利的終端機隨著各種不同的 unix 販賣則會使用他們字訂的溢位序列.

6.1 IBM aixterm

aixterm 可以辨識出 xterm 的溢位序列.

6.2 SGI wsh, xwsh 及 winterm

這類終端機會設定 $TERM=iris-ansi 並採用下列的溢位序列:

- ESCP1.y*string*ESC\ 設定視窗主題爲 *string*
- ESCP3.y*string*ESC\ 設定圖示主題爲 *string*

而 xwsh 完整的溢位序列可以參考 xwsh(1G) 的使用說明.

Irix 的終端機支援 xterm 各別設定視窗與圖示主題的功能, 但不能兩者都設定.

6.3 Sun cmdtool 及 shelltool

cmdtool 及 shelltool 會設定 $TERM=sun-cmd 並採用下列的溢位序列:

- ESC]l*string*ESC\ 設定視窗主題爲 *string*
- ESC]L*string*ESC\ 設定圖示主題爲 *string*

這些眞的是很可怕的程式: 盡量別使用.

6.4 CDE dtterm

dtterm 會設定 $TERM=dtterm, 似乎是可以識別出標準的 xterm 溢位序列及 Sun cmdtool 的溢位序列 (在 Solaris 2.5.1, Digital Unix 4.0, HP-UX 10.20 測試過).

6.5 HPterm

hpterm 會設定 $TERM=hpterm 並採用下列的溢位序列:

- ESC&f0k*length*D*string* 設定視窗主題爲長 *length* 的 *string*
- ESC&f-1k*length*D*string* 設定圖示主題爲長 *length* 的 *string*

一個簡單的 C 語言的程式用來計算長度並回應字串, 如下:

```
#include <string.h>
int main(int argc, char *argv[])
{
    printf("\033&f0k%dD%s", strlen(argv[1]), argv[1]);
    printf("\033&f-1k%dD%s", strlen(argv[1]), argv[1]);
    return(0);
}
```

我們也可以寫一個小的命令腳本程序, 用 ${#string} (zsh, bash, ksh) 或 ${%string} (tcsh) 的延伸功能來找出字串長度. 以下可用在 zsh:

```
case $TERM in
    hpterm)
        str="\e]0;%n@%m: %~\a"
        precmd () {print -Pn "\e&f0k${#str}D${str}"}
        precmd () {print -Pn "\e&f-1k${#str}D${str}"}
        ;;
esac
```

7 附錄: 其它語言的範例

寫一個小程式利用 xterm 的溢位功能把傳遞的參數當成主題. 以下是一些範例.

7.1 C

```
#include <stdio.h>

int main (int argc, char *argv[]) {
  printf("%c]0;%s%c", '\033', argv[1], '\007');
  return(0);
}
```

7.2 Perl

```
#!/usr/bin/perl
print "\033]0;@ARGV\007";
```

8 銘謝Credits

感謝下列諸位提供建議，修正，及範例爲本文增色不少.

Paul D. Smith <psmith@BayNetworks.COM> 及 Christophe Martin <cmartin@ipnl.in2p3.fr> 兩位指出在 bash $PROMPT_COMMAND 中我引述的錯誤部份. 正確的方法變化 是 動態展開的.

Paul D. Smith <psmith@BayNetworks.COM> 建議在 bash 中使用 \[...\] 以整合不能顯示的字元.

Christophe Martin <cmartin@ipnl.in2p3.fr> 提供 ksh 的解決方法.

Keith Turner <keith@silvaco.com> 提供給 Sun 的 cmdtool 及 shelltool 使用的溢位序列.

Jean-Albert Ferrez <ferrez@dma.epfl.ch> 指出在 "PWD" 及 "$PWD" 相矛盾之處，及 "\" 與 "\\" 的用法.

Bob Ellison <papillo@hpellis.fc.hp.com> 及 Jim Searle <jims@broadcom.com> 測試在 HP-UX 下的 dtterm.

Teng-Fong Seak <seak@drfc.cad.cea.fr> 建議用 -s 選項以 hostname，使用 ‘pwd‘，以及 在 csh 下使用 echo .

Trilia <trilia@nmia.com> 建議加入其他語言的用法.

Brian Miller <bmiller@telstra.com.au> 加入 hpterm 的溢位序列與範例.

Lenny Mastrototaro <lenny@click3x.com> 解釋 Irix 終端機環境下，如何使用 xterm 溢位序列.

Paolo Supino <paolo@init.co.il> 建議在 bash 命令列題示下採用 \\$.