Come cambiare il titolo di una xterm

Ric Lister, `ric@giccs.georgetown.edu` v2.0, 27 ottobre 1999

Questo documento spiega come utilizzare le sequenze di escape per modificare dinamicamente i titoli della finestra e dell'icona di una xterm. Sono forniti esempi per diverse shell e l'appendice fornisce le sequenze di escape per altri tipi di terminale. Traduzione a cura di Lorenza Romano (titti@dei.unipd.it) settembre 2000.

Indice

1 Dove trovare questo documento **2**

2 Titoli statici **2**

3 Titoli dinamici **2**

3.1 Sequenze di escape xterm 2

3.2 Riprodurre le sequenze di escape 3

4 Esempi per shell diverse **3**

4.1 zsh 3

4.2 tcsh 4

4.3 bash 5

4.4 ksh 5

4.5 csh 6

5 Stampare il nome del job corrente **6**

5.1 zsh 6

5.2 Altre shell 7

6 Appendice: escape per altri tipi di terminale **7**

6.1 IBM `aixterm` 7

6.2 SGI `wsh`, `xwsh` e `winterm` 7

6.3 Sun `cmdtool` e `shelltool` 7

6.4 CDE `dtterm` 8

6.5 HPterm 8

7 Appendice: esempi in altri linguaggi **8**

7.1 C 8

7.2 Perl 9

8 Ringraziamenti **9**

1 Dove trovare questo documento

Questo documento fa ora parte del

Linux HOWTO Index `http://sunsite.unc.edu/LDP/HOWTO/`

e può essere trovato all'indirizzo

`http://sunsite.unc.edu/LDP/HOWTO/mini/Xterm-Title.html` .

L'ultima versione può sempre essere trovata in diversi formati all'indirizzo

`http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/` .

Questo documento prende il posto dell'howto originario scritto da Winfried Trümper.

2 Titoli statici

Può essere impostato un titolo statico per qualsiasi terminale `xterm`, `color-xterm` o `rxvt` utilizzando le opzioni di riga di comando (switch) `-T` e `-n`:

```
xterm -T Il proprio Titolo della XTerm -n Il proprio Titolo dell'Icona della XTerm
```

3 Titoli dinamici

Molte persone ritengono utile impostare il titolo di un terminale affinché rifletta informazioni dinamiche, ad esempio il nome dell'host a cui è collegato l'utente, l'attuale directory di lavoro, ecc.

3.1 Sequenze di escape xterm

I titoli della finestra e dell'icona di una xterm in esecuzione possono essere modificati utilizzando le sequenze di escape Xterm. Al riguardo sono utili le seguenti sequenze:

- `ESC]0;stringaBEL` – Imposta il nome dell'icona e il titolo della finestra a **stringa**
- `ESC]1;stringaBEL` – Imposta il nome dell'icona a **stringa**
- `ESC]2;stringaBEL` – Imposta il titolo della finestra a **stringa**

dove `ESC` è il carattere **escape** (\033) e `BEL` è il carattere **bell** (\007).

La riproduzione di una di queste sequenze entro la xterm provoca la modifica del titolo della finestra o dell'icona.

Nota: queste sequenze si applicano alla maggior parte dei derivati xterm, ad esempio `nxterm`, `color-xterm` e `rxvt`. Altri tipi di terminale spesso usano sequenze di escape diverse; si veda l'appendice per degli esempi. Per la lista completa delle sequenze di escape xterm si veda il file *ctlseq2.txt* `http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/ctlseq2.txt` , incluso nella distribuzione xterm, oppure

xterm.seq `http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/xterm.seq` fornito con la distribuzione

rxvt `http://www.rxvt.org/` .

3.2 Riprodurre le sequenze di escape

Per le informazioni che rimangono costanti per tutta la durata della vita della shell, ad esempio l'host e lo username, basterà semplicemente fare l'echo della stringa di escape nel file rc della shell:

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

dovrebbe produrre un titolo del tipo `username@hostname` assumendo che le variabili della shell `$USER` e `$HOST` siano impostate correttamente. Le opzioni necessarie al comando `echo` possono variare da shell a shell (si vedano gli esempi che seguono).

Per le informazioni che possono cambiare durante il corso della vita della shell, ad esempio l'attuale directory di lavoro, è necessario applicare queste sequenze ogniqualvolta il prompt cambia. In questo modo la stringa viene aggiornata con ogni comando che si immette e può tenere traccia di informazioni tipo l'attuale directory di lavoro, lo username, l'hostname ecc. A questo scopo, alcune shell mettono a disposizione delle funzioni speciali, altre no e si devono inserire le sequenze del titolo direttamente nella stringa del prompt. Ciò è chiarito nella prossima sezione.

4 Esempi per shell diverse

Di seguito forniamo una serie di esempi per alcune delle shell più comuni. Iniziamo con la `zsh` dato che fornisce parecchie facilitazioni che rendono più agevole il nostro lavoro. Proseguiremo poi attraverso esempi di difficoltà crescente.

In tutti gli esempi esaminiamo la variabile di ambiente `$TERM` per assicurarci di applicare le sequenze di escape solo alle xterm. Verifichiamo che `$TERM=xterm*`; la ragione della presenza del metacarattere è che alcune varianti (ad esempio rxvt) possono impostare `$TERM=xterm-color`.

È necessario fare una osservazione aggiuntiva sui derivati delle shell C, tipo `tcsh` e `csh`. Nelle shell C, le variabili indefinite provocano un errore fatale (fatal error). Perciò prima di esaminare la variabile `$TERM` è necessario verificare la sua esistenza così da non interrompere shell non interattive. Per ottenere ciò è necessario includere gli esempi sottostanti in qualcosa tipo:

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

(A nostro avviso questa è proprio una delle molte ragioni per non usare le shell C. Si veda Csh Programming Considered Harmful `http://language.perl.com/versus/csh.whynot` per una discussione proficua).

Gli esempi che seguono dovrebbero essere utilizzati inserendoli nell'apposito file di inizializzazione della shell; cioè un file di cui le shell interattive fanno il source all'avvio. Nella maggior parte dei casi il nome del file è qualcosa tipo `.`*`shell`*`rc` (per esempio `.zshrc`, `.tcshrc`, ecc).

4.1 zsh

La `zsh` fornisce alcune funzioni ed espansioni che utilizzeremo:

```
precmd ()   funzione che viene eseguita appena prima di ogni prompt
chpwd ()    funzione che viene eseguita ogniqualvolta la directory
            viene modificata
\e          sequenza di escape per escape (ESC)
\a          sequenza di escape per bell (BEL)
```

```
%n          viene espansa in $USERNAME
%m          viene espansa nell'hostname fino al primo '.'
%~          viene espansa nella directory, sostituendo $HOME con '~'
```

Sono disponibili molte altre espansioni: si veda la pagina di manuale `zshmisc`.

Quanto segue imposta perciò il titolo della xterm a: *`username@hostname: directory`*:

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

Ciò potrebbe anche essere ottenuto utilizzando `chpwd()` al posto di `precmd()`. La primitiva `print` si comporta come `echo`, ma ci dà la possibilità di usare le sequenze di escape del prompt `%`.

4.2 tcsh

La `tcsh` ha delle funzioni ed espansioni simili a quelle della `zsh`:

```
precmd ()   funzione che viene eseguita appena prima di ogni prompt
cwdcmd ()   funzione che viene eseguita ogniqualvolta la directory
            viene modificata
%n          viene espansa in username
%m          viene espansa in  hostname
%~          viene espansa nella directory, sostituendo $HOME con '~'
%#          viene espansa in '>' per gli utenti normali,'#' per gli
            utenti root
%{...%}     include una stringa sotto forma di sequenza di escape
            costante
```

Sfortunatamente non c'è un comando equivalente al comando `print` della `zsh` che permetta di usare le sequenze di escape del prompt nella stringa del titolo, perciò la cosa migliore che si possa fare è utilizzare le variabili della shell (in `~/.tcshrc`):

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

In ogni modo ciò fornisce l'intero percorso della directory al posto di far uso di `~`. In alternativa si può inserire la stringa nel prompt:

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

che imposta un prompt pari a `tcsh%` ed un titolo di xterm e icona pari a *username*@*hostname*:*directory*. Si noti che `%{...%}` deve contenere sequenze di escape (e non può essere l'ultima voce nel prompt: si veda la pagina di manuale `tcsh` per i dettagli).

4.3 bash

La `bash` mette a disposizione una variabile `$PROMPT_COMMAND` che contiene un comando da eseguirsi prima del prompt. Questo esempio imposta il titolo a *username*@*hostname*: *directory*:

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

dove `\033` è il codice carattere per `ESC` e `\007` quello per `BEL`.

Si noti che qui è importante l'uso delle virgolette: le variabili vengono sviluppate, espanse se sono tra `...` e non vengono espanse se sono tra `'...'`. Perciò `$PROMPT_COMMAND` è impostata ad un valore non espanso, ma le variabili poste all'interno delle `...` vengono espanse nel momento in cui `$PROMPT_COMMAND` viene utilizzata.

In ogni caso, `$PWD` produce l'intero percorso della directory. Se si vuole usare la forma abbreviata `~`, si deve includere la stringa di escape nel prompt, il che permette di trarre vantaggio dalle seguenti espansioni del prompt, fornite dalla shell:

```
\u        viene espansa in $USERNAME
\h        viene espansa in hostname fino al primo '.'
\w        viene espansa in directory, sostituendo $HOME with '~'
\$        viene espansa in '$' per gli utenti normali, '#' per root
\[...\]   include una sequenza di caratteri non stampabili
```

Perciò, quanto segue produce un prompt `bash$` e un titolo di xterm *username*@*hostname*: *directory*:

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

Si noti l'uso di `\[...\]`, che dice alla `bash` di ignorare i caratteri di controllo non stampabili nel calcolo della lunghezza del prompt. Diversamente i comandi per l'editor di linea si confonderebbero nel posizionare il cursore.

4.4 ksh

La `ksh` fornisce poco dal punto di vista delle funzioni ed espansioni, perciò si deve inserire la stringa di escape nel prompt affinché venga aggiornata dinamicamente. Questo esempio produce un titolo `username@hostname: directory` e un prompt `ksh$`.

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

```
        *)
            PS1='ksh$ '
            ;;
esac
```

Ad ogni modo, `$PWD` fornisce l'intero percorso della directory. Si può rimuovere il prefisso `$HOME/` dalla directory usando il costrutto `${...##...}`. Si può anche usare `${...%%...}` per troncare l'hostname:

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

Si noti che, nella stringa del prompt, `^[` e `^G` sono singoli caratteri per `ESC` e `BEL` (possono essere inseriti in emacs utilizzando `C-q ESC` e `C-q C-g`).

4.5 csh

Tutto ciò è davvero molto difficile in `csh` e si finisce per fare qualcosa del tipo:

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

dove si è dovuto definire un alias per il comando `cd` per svolgere la funzione di invio delle sequenze di escape. Si noti che, nella stringa del prompt, `^[` e `^G` sono singoli caratteri per `ESC` e `BEL` (possono essere inseriti in emacs utilizzando `C-q ESC` e `C-q C-g`).

Note: su alcuni sistemi si può utilizzare `hostname -s` per ottenere un hostname breve anziché interamente specificato. Alcuni utenti, con collegamenti simbolici a directory, possono scoprire che `` `pwd` `` (apici per eseguire il comando `pwd`) fornisce un percorso più accurato di `$cwd`.

5 Stampare il nome del job corrente

Spesso un utente avvia un job in primo piano (in foreground) di lunga durata tipo `top`, un editor, un client email, ecc e desidera che il nome del job sia mostrato nel titolo. Questo è un problema più spinoso e si risolve facilmente solo nella `zsh`.

5.1 zsh

La `zsh` fornisce una funzione primitiva ideale per questo scopo:

```
preexec()   funzione che viene eseguita esattamente prima che un
            comando venga eseguito
$*,$1,...   argomenti passati a preexec()
```

Perciò si può inserire nel titolo il nome del job nel seguente modo:

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

Nota: la funzione `preexec()` è apparsa attorno alla versione 3.1.2 della `zsh`, perciò una versione precedente dovrà essere aggiornata.

5.2 Altre shell

Ciò non è facile con altre shell che sono prive di una funzione equivalente alla `preexec()`. Se qualcuno dispone di esempi per favore li spedisca all'autore.

6 Appendice: escape per altri tipi di terminale

Molti terminali moderni sono discendenti di `xterm` o `rxvt` e supportano le sequenze di escape che abbiamo utilizzato fino a questo punto. Alcuni terminali proprietari forniti con tipi diversi di unix usano le proprie sequenze di escape.

6.1 IBM `aixterm`

`aixterm` riconosce le sequenze di `escape xterm`.

6.2 SGI `wsh`, `xwsh` e `winterm`

Questi terminali impostano `$TERM=iris-ansi` e usano i seguenti escape:

- `ESCP1.y`*`stringa`*`ESC\ Imposta il titolo della finestra a` *`stringa`*
- `ESCP3.y`*`stringa`*`ESC\ Imposta il titolo dell'icona a` *`stringa`*

Per l'intera lista degli escape di `xwsh` si veda la pagina di manuale `xwsh(1G)`.

I terminali Irix supportano gli escape `xterm` per impostare separatamente il titolo della finestra e il titolo dell'icona ma non l'escape per impostare entrambi.

6.3 Sun `cmdtool` e `shelltool`

`cmdtool` e `shelltool` impostano entrambi `$TERM=sun-cmd` e usano i seguenti escape:

- `ESC]l`*`stringa`*`ESC\ Imposta il titolo della finestra a` *`stringa`*
- `ESC]L`*`stringa`*`ESC\ Imposta il titolo dell'icona a` *`stringa`*

Sono davvero programmi pessimi: si usi qualcos'altro.

6.4 CDE `dtterm`

`dtterm` imposta `$TERM=dtterm` e sembra riconoscere sia le sequenze di escape `xterm` standard che le sequenze di escape `cmdtool` della Sun (verificato su Solaris 2.5.1, Digital Unix 4.0, HP-UX 10.20).

6.5 HPterm

`hpterm` imposta `$TERM=hpterm` e utilizza i seguenti escape:

- `ESC&f0k`*lunghezza*`D`*stringa* Imposta il titolo della finestra a *stringa* di lunghezza *lunghezza*
- `ESC&f-1k`*lunghezza*`D`*stringa* Imposta il titolo dell'icona a *stringa* di lunghezza *lunghezza*

Un programma C base per calcolare la lunghezza e fare l'echo della stringa può essere questo:

```
#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);
}
```

Possiamo scrivere uno shell-script simile utilizzando l'espansione `${#stringa}` (`zsh`, `bash`, `ksh`) o l'espansione `${%stringa}` (`tcsh`) per trovare la lunghezza della stringa. Quanto segue è per `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 Appendice: esempi in altri linguaggi

Può essere utile scrivere un piccolo programma per stampare un argomento nel titolo utilizzando gli escape `xterm`. Sotto è fornito qualche esempio.

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 Ringraziamenti

Un ringraziamento alle seguenti persone che hanno fornito consigli, errata corrige ed esempi per questo documento.

Paul D. Smith <`psmith@BayNetworks.COM`> e Christophe Martin <`cmartin@ipnl.in2p3.fr`> hanno entrambi fatto notare che avevo le virgolette nel modo sbagliato nella `$PROMPT_COMMAND` della `bash`. Averle capite esattamente significa che le variabili *vengono* espanse dinamicamente.

Paul D. Smith <`psmith@BayNetworks.COM`> ha suggerito l'uso di `\[...\]` nel prompt della `bash` per includere caratteri non stampabili.

Christophe Martin <`cmartin@ipnl.in2p3.fr`> ha provveduto alla soluzione per `ksh`.

Keith Turner <`keith@silvaco.com`> ha fornito le sequenze di escape per `cmdtool` e `shelltool` della Sun.

Jean-Albert Ferrez <`ferrez@dma.epfl.ch`> ha messo in evidenza alcune incoerenze nell'uso di `PWD` e `$PWD` e nell'uso di \ in opposizione a \\.

Bob Ellison <`papillo@hpellis.fc.hp.com`> e Jim Searle <`jims@broadcom.com`> hanno verificato `dtterm` su HP-UX.

Teng-Fong Seak <`seak@drfc.cad.cea.fr`> ha suggerito l'opzione `-s` per `hostname`, l'uso di `` `pwd` `` e l'uso di `echo` nella `csh`.

Trilia <`trilia@nmia.com`> ha suggerito gli esempi in altri linguaggi.

Brian Miller <`bmiller@telstra.com.au`> ha fornito le sequenze di escape e gli esempi per `hpterm`.

Lenny Mastrototaro <`lenny@click3x.com`> ha spiegato l'uso nei terminali Irix delle sequenze di escape xterm.

Paolo Supino <`paolo@init.co.il`> ha suggerito l'uso di `\\$` nel prompt della `bash`.