De titel van een xterm wijzigen

Ric Lister, ric@giccs.georgetown.edu,
Vertaald door: Ellen Bokhorst bokkie@nl.linux.org v2.0, 27 oktober 1999

In dit document wordt uitgelegd hoe escape sequences kunnen worden gebruikt om dynamisch de titel van een xterm-venster en een ikoon te wijzigen. Voor verscheidene shells wordt voorzien in voorbeelden, en in de appendix zijn escape sequences gegeven voor een aantal andere typen terminals.

Inhoudsopgave

1 Waar dit document is te vinden

Dit document maakt nu onderdeel uit van de *Linux HOWTO Index* `<http://sunsite.unc.edu/LDP/HOWTO/>` en het is te vinden op `<http://sunsite.unc.edu/LDP/HOWTO/mini/Xterm-Title.html>`.

De laatste versie is altijd in verscheidene formaten te vinden op `<http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/>`.

Dit document overvleugelt de oorspronkelijke howto geschreven door Winfried Trümper.

2 Statische titels

Een statische titel kan voor ieder van de terminals `xterm`, `color-xterm` of `rxvt` worden ingesteld, door gebruik te maken van de opties `-T` en `-n`:

```
xterm -T "Mijn XTerm Titel-n "Mijn XTerm Ikoon Titel"
```

3 Dynamische titels

Veel mensen vinden het handig om de titel van een terminal zo in te stellen dat het dynamisch informatie weergeeft, zoals de hostnaam waarop de gebruiker is ingelogd of de huidige werkdirectory, enz.

3.1 xterm escape sequences

Ikoon- en venstertitels van een draaiende xterm kunnen worden gewijzigd door gebruik te maken van XTerm escape sequences. In deze zin zijn de volgende sequences nuttig:

- `ESC]0;stringBEL` – Stel de naam van het ikoon en de venstertitel in op **string**
- `ESC]1;stringBEL` – Stel de naam van het ikoon in op **string**
- `ESC]2;stringBEL` – Stel de venstertitel in op **string**

`ESC` is hier het **escape**-teken (\033), en `BEL` is het **bell** teken (\007).

Het afdrukken van één van deze sequences zorgt dat de titel van het venster of de ikoon wordt gewijzigd.

Opmerking: deze sequences zijn van toepassing op de meeste afgeleiden van xterm, zoals `nxterm`, `color-xterm` en `rxvt`. Andere typen terminals maken vaak gebruik van andere escapes; zie de appendix voor voorbeelden. Zie het bestand *ctlseq2.txt* `<http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/ctlseq2.txt>`, Voor een volledige lijst met xterm escape sequences, welke met de xterm distributie wordt meegeleverd, of *xterm.seq* `<http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/xterm.seq>`, welke wordt meegeleverd met de *rxvt* `<http://www.rxvt.org/>` distributie.

3.2 Afdrukken van escape sequences

Voor informatie die gedurende de levensduur van deze shell gelijk blijft, zoals de host- en gebruikersnaam, volstaat een echo-commando door eenvoudigweg de escape string in het rc bestand van de shell te plaatsen:

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

zal een titel produceren zoals `username@hostname`, in de veronderstelling dat de shellvariabelen `$USER` en `$HOST` correct zijn ingesteld. De benodigde opties voor `echo` kunnen per shell variëren (zie de voorbeelden verderop).

Voor informatie die tijdens de levensduur kan wijzigen, zoals de huidige werkdirectory, moeten deze escapes echt, iedere keer dat de prompt wijzigt, worden aangepast. Zo wordt bij iedere opdracht die je aanroept de string gewijzigd en kan informatie worden bijgehouden zoals de huidige werkdirectory, naam van de gebruikers, hostnaam, enz. Een aantal shells voorziet in speciale functies voor dit doel, een aantal doet dit niet en we moeten de titel sequences direct in de promptstring voegen. Dit wordt in de volgende sectie geïllustreerd.

4 Voorbeelden voor verschillende shells

Hieronder wordt een set voorbeelden gegeven voor de wat meer gebruikelijke shells. We beginnen met `zsh` aangezien het diverse mogelijkheden biedt die onze taak er veel eenvoudiger op maken. We zullen vervolgens steeds moeilijkere voorbeelden doorwerken.

In alle voorbeelden testen we de omgevingsvariabele `$TERM` om er zeker van te zijn dat we de escapes alleen toepassen op xterms. We testen op `$TERM=xterm*`; het jokerteken wordt gebruikt omdat een aantal varianten (zoals rxvt) deze omgevingsvariabele in kan stellen op `$TERM=xterm-color`.

We zouden een extra opmerking over C-shellafgeleiden, zoals `tcsh` en `csh` moeten maken. In C shells, worden ongedefinieerde variabelen als fatale fouten beschouwd. Daarom is het nodig voor het testen van de variabele `$TERM`, te testen op het bestaan ervan. Om dit te bereiken moet je de voorbeelden hieronder wijzigen in zoiets als:

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

(Wij vinden dit één van de vele redenen om geen gebruik te maken van C-shells. Zie Csh Programming Considered Harmful `<http://language.perl.com/versus/csh.whynot>` voor een nuttige bespreking).

De voorbeelden hierna zouden kunnen worden gebruikt door ze te plaatsen in het van toepassing zijnde shell-initialisatiebestand; d.w.z. één die bij het opstarten door interactieve shells wordt ingelezen. In de meeste gevallen heeft deze een naam zoals in `.`*shell*`rc` (b.v. `.zshrc`, `.tcshrc`, enz).

4.1 zsh

`zsh` biedt de volgende functies en uitbreidingen, waar we gebruik van zullen maken:

```
precmd ()   een functie die vóór iedere prompt wordt uitgevoerd
chpwd ()    een functie die wordt uitgevoerd wanneer de directory wijzigt
\e          escape sequence voor escape (ESC)
\a          escape sequence voor bell (BEL)
%n          extraheert naar $USERNAME
%m          extraheert naar hostnaam tot aan de eerste '.'
%~          extraheert naar directory, $HOME wordt vervangen door '~'
```

Er zijn nog heel wat meer uitbreidingen beschikbaar: zie de `zshmisc` man page.

Dus het volgende zal de xterm titel instellen op *gebruikersnaam@hostnaam: directory@"gebruikersnaam@hostnaam: directory":*

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

Dit zou ook bewerkstelligd kunnen worden door gebruik te maken van chpwd() in plaats van precmd(). De ingebouwde opdracht print werkt net als echo, maar geeft ons ook nog eens toegang tot de % prompt escapes.

4.2 tcsh

tcsh heeft een aantal functies en uitbreidingen die vergelijkbaar zijn met die van zsh:

```
precmd ()   een functie die voor iedere prompt wordt uitgevoerd
cwdcmd ()   een functie die wordt uitgevoerd wanneer de directory wijzigt
%n          extraheert naar gebruikersnaam
%m          extraheert naar hostnaam
%~          extraheert naar directory, $HOME wordt vervangen door '~'
%#              breidt uit naar '>' voor gewone gebruikers, '#' voor
%root-gebruikers
%{...%}     voegt een string in als een letterlijke escape sequence
```

Helaas is er geen equivalente voor de opdracht print van zsh die het ons toestaat prompt escapes in de titelstring te gebruiken, dus het beste wat we kunnen doen is gebruik te maken van shellvariabelen (in ~/.tcshrc):

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

Hierdoor krijg je echter het volledige pad van de directory in plaats dat er gebruik wordt gemaakt van ~. In plaats daarvan kun je de string in de prompt plaatsen:

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

hiermee wordt de prompt ingesteld op % @", en een xterm-titel en ikoon gebruikersnaam@hostnaam: directory@"gebruikersnaam@hostnaam: directory". Houd in de gaten dat de %{...%}@"tussen escape sequences moet worden geplaatst (en niet het laatste item in de prompt kan zijn: zie de manpage van tcsh voor details).

4.3 bash

bash voorziet in de variabele $PROMPT_COMMAND waaraan de opdracht is toegekend welke vóór de weergave van de prompt wordt uitgevoerd. Dit voorbeeld stelt de titel in op gebruikersnaam@hostnaam: directory:

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

\033 is hier de tekencode voor ESC, en \007 voor BEL.

De aanhalingstekens zijn hier belangrijk: variabelen worden geëxtraheerd naar "..."

, en niet naar '...'. Dus$PROMPT_COMMAND wordt op een niet geëxtraheerde waarde ingesteld, en de variabelen binnen de "..."

worden geëxtraheerd wanneer $PROMPT_COMMAND wordt gebruikt.

$PWD produceert echter het volledige pad van de directory. Als we de verkorte vorm ~ willen, moet we de escape string in de prompt opnemen, wat maakt dat we voordeel kunnen hebben van de door in de shell voorziene uitbreidingen op de prompt:

```
\u          extraheert naar $USERNAME
\h          extraheert naar hostnaam tot aan de eerste '.'
\w          extraheert naar directory, $HOME wordt vervangen door '~'
\$              extraheert voor gewone gebruikers naar '$', '#' voor root
\[...\]     sluit een reeks in met niet afdrukbare tekens
```

Dus het volgende produceert een prompt bash$, en een xterm titel gebruikersnaam@hostnaam: directory:

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

Let op het gebruik van \[...\], waarmee aan bash wordt aangegeven de niet afdrukbare controletekens te negeren, wanneer de breedte van de prompt wordt berekend. Anders onstaat er verwarring bij het plaatsen van de cursor bij regelediting opdrachten.

4.4 ksh

ksh voorziet in weinig functies en uitbreidingen, dus moeten we de escapestring in de prompt voegen, om het dynamisch te laten bijwerken. Dit voorbeeld produceert een titel met gebruikersnaam@hostnaam: directory en een prompt ksh$.

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

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

$PWD produceert echter het volledige pad van de directory. We kunnen het voorvoegsel $HOME/ van de directory verwijderen door gebruik te maken van de constructie ${...##...}. We kunnen ook gebruik maken van ${...%%...} om de hostnaaam af te kappen:

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

De ^[en ^G in de promptstring zijn enkele tekens voor ESC en BEL (kan in emacs door het invoeren van C-q ESC en C-q C-g).

4.5 csh

Dit is in csh inderdaad erg moeilijk, en we komen ongeveer hierop uit:

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

waarbij we een alias voor de opdracht cd opdracht gebruiken om de escape sequence te sturen. De ^[en ^G in de string zijn enkele tekens voor ESC en BEL (kunnen in emacs worden ingevoerd met C-q ESC en C-q C-g).

Opmerkingen: op een aantal systemen kan hostname -s worden gebruikt om een afgekorte versie, in plaats van de fully-qualified hostnaam te verkrijgen. Een aantal gebruikers met symlinked directory's bemerken mogelijk dat `pwd` (achterwaartse aanhalingstekens voor het uitvoeren van de opdracht pwd) een accurater pad teruggeeft dan $pwd.

5 Afdrukken van de huidige jobnaam

Vaak zal een gebruiker een voorgrondtaak opstarten zoals top, een editor, een emailclient, enz, en willen dat de naam van de job in de titel wordt weergegeven. Dit is een wat neteliger probleem en kan alleen eenvoudig worden bewerkstelligd met zsh.

5.1 zsh

zsh voorziet in een ideale ingebouwde functie voor dit doel:

```
preexec()   een functie die net voor uitvoering van een opdracht wordt
uitgevoerd
$*,$1,...   argumenten doorgegeven aan preexec()
```

Dus we kunnen als volgt de jobnaam in de titel voegen:

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

Opmerking: de preexec() functie verscheen zo rond versie 3.1.2 van zsh, dus wellicht moet je een eerdere versie upgraden.

5.2 Andere shells

Dit is in andere shells niet eenvoudig door het ontbreken van een equivalente opdracht als de preexec() functie. Als iemand anders voorbeelden heeft, email deze dan alsjeblieft naar de auteur.

6 Appendix: escapes voor andere typen terminals

Veel moderne terminals zijn afgeleid van xterm of rxvt en ondersteunen de tot nu toe gebruikte escape sequences. Een aantal eigen terminals die met diverse varianten van unix worden meegeleverd maken gebruik van eigen escape sequences.

6.1 IBM `aixterm`

aixterm herkent de xterm escape sequences.

6.2 SGI `wsh`, `xwsh` en `winterm`

Deze terminals zetten de termvariabele in als $TERM=iris-ansi en gebruiken de volgende escapes:

- *ESCP1.ystringESC\ Stel de venstertitel in op string*
- *ESCP3.ystringESC\ Stel de ikoontitel in op string*

Zie de manpage van xwsh(1G) voor een volledige lijst met xwsh escapes.

De Irix terminals ondersteunen ook de xterm escapes om afzonderlijk te venstertitel en ikoontitel in te stellen, maar niet de escape om ze beiden in te stellen.

6.3 Sun `cmdtool` en `shelltool`

cmdtool en shelltool stellen stellen de termvariabele beiden in als $TERM=sun-cmd en maken gebruik van de volgende escapes:

- *ESC]lstringESC\ Stel de venster titel in op string*

- *ESC]LstringESC\ Stel de titel van het ikoon in op string*

Dit zijn werkelijk afgrijselijke programma's: gebruik iets anders.

6.4 CDE dtterm

dtterm stelt de termvariabele in op $TERM=dtterm, en blijkt standaard zowel de standaard xterm als de Sun cmdtool sequences te herkennen (getest onder Solaris 2.5.1, Digital Unix 4.0, HP-UX 10.20).

6.5 HPterm

hpterm stelt de termvariabele in als $TERM=hpterm en gebruikt de volgende escapes:

- *ESC&f0klengthDstring Stel venstertitel in op string van lengte length*
- *ESC&f-1klengthDstring Stel ikoontitel in op string met de lengte length*

Een basis C-programma om de lengte van de string te berekenen en deze terug te geven, ziet er ongeveer zo uit:

```
#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);
}
```

We kunnen een vergelijkbaar script schrijven door gebruik te maken van de ${#string} (zsh, bash, ksh) of ${%string} (tcsh)'s uitbreiding om de lengte van de string te achterhalen. Het volgende geldt voor 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 Appendix: voorbeelden in andere talen

Het kan handig zijn een klein programma te schrijven waarbij een argument naar de titel wordt afgedrukt door gebruik te maken van de xterm escapes. Hierna wordt een aantal voorbeelden gegeven.

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 Krediet

Met dank aan de volgende mensen die voorzagen in advies, correcties op fouten en voorbeelden voor dit document.

Paul D. Smith <psmith@BayNetworks.COM> en Christophe Martin <cmartin@ipnl.in2p3.fr> wezen er beiden op dat ik de aanhalingstekens verkeerd om had in $PROMPT_COMMAND van bash. Ze juist plaatsen betekent dat variabelen dynamisch worden geëxtraheerd.

Paul D. Smith <psmith@BayNetworks.COM> deed de suggestie voor het gebruik van \[...\] in de bash prompt voor het opnemen van niet afdrukbare tekens.

Christophe Martin <cmartin@ipnl.in2p3.fr> leverde de oplossing voor ksh.

Keith Turner <keith@silvaco.com> leverde de escape sequences voor Sun cmdtool en shelltool.

Jean-Albert Ferrez <ferrez@dma.epfl.ch> wees op een aantal inconsequenties in het gebruik van "PWDën $PWD@"$PWD", en in het gebruik van \@"vs \\@".

Bob Ellison <papillo@hpellis.fc.hp.com> en Jim Searle <jims@broadcom.com> testte dtterm onder HP-UX.

Teng-Fong Seak <seak@drfc.cad.cea.fr> deed de suggestie voor het gebruik van de -s optie voor hostname, het gebruik van 'pwd', en echo onder csh.

Trilia <trilia@nmia.com> deed de suggestie voor voorbeelden in andere talen.

Brian Miller <bmiller@telstra.com.au> leverde de escape sequences en voorbeelden aan voor hpterm.

Lenny Mastrototaro <lenny@click3x.com> gaf een uitleg van Irix terminal's gebruik van xterm escape sequences.

Paolo Supino <paolo@init.co.il> deed de suggestie voor het gebruik van \\$ in de bash prompt.