# Comment changer le titre d'un xterm

Ric Lister, ric@giccs.georgetown.edu
traduction Jean-Albert Ferrez, Jean-Albert.Ferrez@epfl.ch Dernière modification : 22.11.1999, v2.0

Ce document explique comment utiliser des séquences d'échappement pour modifier dynamiquement le titre de la fenêtre et de l'icône d'un xterm. Des exemples sont donnés pour plusieurs shells, et l'annexe donne les séquences pour d'autres types de terminaux.

## Contents

# 1 Où trouver ce document

Ce document fait désormais partie des *HOWTOs Linux* `http://sunsite.unc.edu/LDP/HOWTO/` et peut être trouvé à : `http://sunsite.unc.edu/LDP/HOWTO/mini/Xterm-Title.html` .

La dernière version en date se trouve à : `http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/` .

Ce document remplace le howto initial écrit par Winfried Trümper.

Ndt : La version française de ce document se trouve à :

`http://www.freenix.fr/linux/HOWTO/mini/Xterm-Title.html`

# 2 Titres statiques

Les titres des émulateurs de terminaux xterm, color-xterm ou rxvt peuvent être donnés sur la ligne de commande avec les options -T et -n :

```
xterm -T "Le titre de mon XTerm" -n "Le titre de son icône"
```

# 3 Titres dynamiques

Bon nombre de personnes trouvent utile de faire apparaître dans le titre de leur terminal une information qui change dynamiquement, telle que le nom du serveur sur lequel on est connecté, le répertoire courant, etc.

## 3.1 Les séquences d'échappement xterm

On peut changer le titre de la fenêtre et de l'icône dans un xterm lancé en utilisant les séquences d'échappement XTerm. Les séquences suivantes sont utiles dans ce but :

- `ESC]0;nomBEL` – Change le titre de la fenêtre et de l'icône
- `ESC]1;nomBEL` – Change le titre de l'icône
- `ESC]2;nomBEL` – Change le titre de la fenêtre

où ESC est le caractère escape (échappement, \033), et BEL est le caractère bell (bip, \007).

Afficher l'une de ces séquences dans un xterm causera le changement du titre de la fenêtre ou de l'icône.

**Note**: Ces séquences fonctionnent également avec la plupart des dérivés de xterm, tels que nxterm, color-xterm et rxvt. D'autres émulateurs de terminaux utilisent d'autres séquences ; quelques exemples sont donnés en annexe. La liste complète des séquences d'échappement est donnée dans le fichier *ctlseq2.txt* `http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/ctlseq2.txt`

de la distribution de xterm, ou dans le fichier *xterm.seq* `http://www.giccs.georgetown.edu/~ric/howto/Xterm-Title/xterm.seq` de la distribution de *rxvt* `http://www.rxvt.org/` .

### 3.2 Afficher les séquences d'échappement

Pour les informations qui ne changent pas au cours de l'éxécution du shell, telles que le serveur et le nom d'utilisateur, il suffit d'afficher les séquences depuis le fichier rc du shell :

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

devrait donner un titre du genre `nom@serveur`, pour autant que les variables `$USER` et `$HOST` soient correctes. Les options requises pour echo peuvent dépendre du shell (cf ci-dessous).

Pour les informations qui peuvent changer au cours de l'exécution du shell, telles que le répertoire courant, ces séquences doivent vraiment être données lors de chaque changement de l'invite. De cette façon, le titre est mis à jour lors de chaque commande et peut ainsi refléter des informations telles que le répertoire en cours, le nom d'utilisateur, le nom du serveur, etc. Certains shells offrent des fonctions spéciales pour y parvenir, d'autres pas : il faut dans ce cas insérer la chaîne directement dans le texte de l'invite.

## 4 Exemples pour quelques shells

Nous donnons ci-dessous des exemples pour les shells les plus courants. Nous commençons avec `zsh` car il offre des possibilités qui facilitent grandement notre tâche. Nous progresserons ensuite vers des exemples de plus en plus difficiles.

Dans tous les exemples ci-dessous, on teste la variable d'environnement TERM pour être certain de n'appliquer ces séquences que si l'on est dans un xterm (ou dérivé). Le test est fait sur TERM=xterm*, de manière à inclure au passage les variantes telles que TERM=xterm-color (défini par rxvt).

Encore une remarque au sujet des dérivés du C shell tels que `tcsh` et `csh`. Dans ces shells, les variables non-définies causent des erreurs fatales. Il est dès lors nécessaire avant de tester la valeur de la variable `$TERM`, de tester si elle existe pour ne pas interrompre un shell non-interactif. Pour y parvenir, il faut inclure les exemples ci-dessous dans quelque chose du genre :

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

(À notre avis, il s'agit d'une raison parmi beaucoup d'autres de ne pas utiliser les C shells. Voir Csh Programming Considered Harmful `http://language.perl.com/versus/csh.whynot` pour une discussion utile).

Pour utiliser les exemples suivants, placez-les dans le fichier d'initialisation du shell aproprié, c'est-à-dire un fichier lu lors du lancement d'un shell interactif. Le plus souvent il s'agit de `.`*`shell`*`rc` (ex : `.zshrc`, `.tcshrc`, etc.)

### 4.1 zsh

On utilise quelques fonctions et codes offerts par `zsh` :

```
precmd ()   fonction exécutée juste avant chaque invite
chpwd ()    fonction exécutée lors de chaque changement de répertoire
\e          code du caractère escape (ESC)
\a          code du caractère bip (BEL)
%n          code remplacé par $USERNAME
%m          code remplacé par le hostname jusqu'au premier '.'
%~          code remplacé par le répertoire, avec '~' à la place de $HOME
```

De nombreux autres codes sont disponibles, voir 'man zshmisc'.

Ainsi, le code suivant, mis dans ~/.zshrc, affiche "nom@serveur:répertoire" dans le titre de la fenêtre (et de l'icône).

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

On arrive au même résultat en utilisant `chpwd()` au lieu de `precmd()`. La commande interne `print` fonctionne comme `echo`, mais donne accès aux séquences `%`.

## 4.2 tcsh

`tcsh` offre des possibilités similaires à celles de `zsh` :

```
precmd ()   fonction exécutée juste avant chaque invite
chpwd ()    fonction exécutée lors de chaque changement de répertoire
%n          code remplacé par $USERNAME
%m          code remplacé par le hostname jusqu'au premier '.'
%~          code remplacé par le répertoire, avec '~' à la place de $HOME
```

Malheureusement, il n'y a pas d'équivalent à la fonction print de `zsh` qui permette d'utiliser les codes de l'invite dans la chaîne du titre ; le mieux que l'on puisse faire est d'utiliser les variables du shell (dans `~/.tcshrc`) :

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

mais on obtient alors le chemin complet du répertoire, sans '~'. Par contre, on peut mettre la chaîne dans l'invite :

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

ce qui donne "`tcsh%` " comme invite, et "*`nom@serveur: répertoire`*" dans le titre (et l'icône) de xterm. Les "%{...%}" doivent être placés autour des séquences d'échappement (et ne peuvent pas être le dernier élément de l'invite, 'man tcsh' donne plus de détails).

## 4.3 bash

`bash` offre la variable `PROMPT_COMMAND` qui contient une commande à exécuter avant d'afficher l'invite. Ce code (inséré dans `~/.bashrc`) affiche `nom@serveur: répertoire` dans le titre de la fenêtre (et de l'icône).

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

où `\033` est le caractère `ESC` et `\007 BEL`.

Il convient de noter que les guillemets jouent un rôle important : les variables entre `"..."` sont évaluées, alors que celles entre `"..."` ne le sont pas. Ainsi, `PROMPT_COMMAND` reçoit bien le nom des variables, ces dernières étant évaluées lorsque `PROMPT_COMMAND` est invoquée.

Cependant, `$PWD` donne le répertoire complet. Si l'on veut utiliser le raccourci `~`, il faut placer toute la séquence d'échappement dans l'invite pour avoir accès aux codes suivants :

```
\u        est remplacé par $USERNAME
\h        est remplacé par le hostname jusqu'au premier '.'
\w        est remplacé par le repertoire, avec '~' à la place de $HOME
\[...\]   signale une suite de caractère non-imprimables
```

Ainsi le code suivant produit l'invite `bash$` , et place `nom@serveur: répertoire` dans le titre (et l'icône) de xterm.

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

L'utilisation de `\[...\]` signale à `bash` la présence de caractères non-imprimables, information dont il a besoin lorsqu'il calcule la longueur de l'invite. Sans cette précaution, les commandes d'édition de ligne ne savent plus très bien où placer le curseur.

## 4.4 ksh

`ksh` n'offre pas grand chose en terme de fonctions et codes, il faut donc mettre notre chaîne dans l'invite pour qu'elle soit mise à jour dynamiquement. L'exemple suivant produit l'invite `ksh$` , et place `nom@serveur: répertoire` dans le titre (et l'icône) de xterm.

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

Cependant, `$PWD` donne le répertoire complet. On peut ôter le préfixe `$HOME/` en utilisant la construction `${...##...}`. De même, on peut tronquer le nom du serveur à l'aide de `${...%%...}`.

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

Les caractères `^[` et `^G` désignent `ESC` et `BEL` (ils peuvent être saisis dans emacs à l'aide de `C-q ESC` et `C-q C-g`).

### 4.5 csh

C'est assez difficile à réaliser avec `csh`. On finit par mettre ce qui suit dans le ~/.cshrc :

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

Il a fallu faire un alias de la commande `cd` pour mettre à jour l'invite. Les caractères `^[` et `^G` désignent `ESC` et `BEL` (ils peuvent être saisis dans emacs à l'aide de `C-q ESC` et `C-q C-g`).

Notes : sur certains systèmes `hostname -s` peut être utilisé pour obtenir le nom de la machine au lieu du nom qualifié. Les utilisateurs ayant des liens symboliques sur des répertoires trouveront `` `pwd` `` plus précis que `$cwd`.

## 5 Afficher le nom de la commande en cours d'exécution

Souvent un utilisateur lance une longue commande en avant plan telle que `top`, un éditeur, un lecteur de courrier électronique, etc, et voudrait que le nom de cette commande figure dans le titre de la fenêtre. C'est un problème délicat qui n'est facile à résoudre qu'avec `zsh`.

### 5.1 zsh

`zsh` offre une fonction idéale pour cet objectif :

```
preexec()   fonction exécutée juste avant qu'une commande soit exécutée
$*,$1,...   arguments passés à preexec()
```

On peut donc insérer le nom de la commande de la manière suivante :

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

Note: la fonction `preexec()` est apparue vers la version 3.1.2 de `zsh`, vous devrez peut-être mettre à jour votre ancienne version.

### 5.2 Autres shells

Ce n'est pas facile avec les autres shells qui n'ont pas l'équivalent de la fonction `preexec()`. Si quelqu'un a des exemples, merci de les communiquer par email à l'auteur.

## 6 Annexe : séquences d'échappement pour d'autres émulateurs de terminaux

De nombreux émulateurs de terminaux modernes sont des dérivés de `xterm` ou `rxvt` et acceptent les séquences d'échappement que nous avons utilisées jusqu'ici. Certains terminaux propriétaires fournis avec les diverses variantes d'unix utilisent leur propres séquences.

### 6.1 `aixterm` d'IBM

`aixterm` reconnaît les séquences d'échappement de `xterm`.

### 6.2 `wsh`, `xwsh` et `winterm` de SGI

Ces terminaux définissent `$TERM=iris-ansi` et utilisent :

- `ESCP1.y`*texte*`ESC\ Pour le titre de la fenêtre`
- `ESCP3.y`*texte*`ESC\ Pour le titre de l'icône`

La liste complète des séquences est donnée dans la page man `xwsh(1G)`.

Les terminaux d'Irix supportent également les séquences de `xterm` pour définir individuellement le titre de la fenêtre et de l'icône, mais pas celle pour définir les deux en même temps.

### 6.3 `cmdtool` et `shelltool` de Sun

`cmdtool` et `shelltool` définissent `$TERM=sun-cmd` et utilisent :

- `ESC]l`*texte*`ESC\ Pour le titre de la fenêtre`
- `ESC]L`*texte*`ESC\ Pour le titre de l'icône`

Ce sont des programmes vraiment horribles, il vaut mieux utiliser autre chose.

### 6.4 CDE dtterm

`dtterm` définit `$TERM=dtterm`. Il semble qu'il reconnaisse à la fois les séquences `xterm` standard ainsi que celles du `cmdtool` de Sun (testé sur Solaris 2.5.1, Digital Unix 4.0, HP-UX 10.20).

### 6.5 HPterm

`hpterm` définit `$TERM=hpterm` et utilise les séquences suivantes :

- `ESC&f0k`*longueur*`D`*texte* Donne le texte *texte* de longueur *longueur* comme titre de fenêtre
- `ESC&f-1k`*longueur*`D`*texte* Donne le texte *texte* de longueur *longueur* comme nom de l'icône

Un programme C simple pour calculer la longueur et afficher la bonne séquence ressemble à :

```
#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);
}
```

On peut également écrire un shell-script équivalent, utilisant `${#string}` (`zsh`, `bash`, `ksh`) ou `${%string}` (`tcsh`) pour obtenir la longueur d'une chaîne. L'exemple suivant est pour `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 Annexe : exemples dans d'autres langages

Il peut être utile d'écrire des bouts de codes pour changer le titre de la fenêtre à l'aide des séquences `xterm`. Voici quelques exemples :

## 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 Crédits

Merci aux personnes suivantes pour leur contribution à ce document.

Paul D. Smith <psmith@BayNetworks.COM> et Christophe Martin <cmartin@ipnl.in2p3.fr> ont tous les deux remarqué que j'avais interverti les guillemets dans le `PROMPT_COMMAND` pour `bash`. Les avoir dans le bon ordre garantit que les variables sont évaluées dynamiquement.

Paul D. Smith <psmith@BayNetworks.COM> a proposé de protéger les caractères non-imprimables dans l'invite de `bash`.

Christophe Martin <cmartin@ipnl.in2p3.fr> a donné la solution pour ksh.

Keith Turner <keith@silvaco.com> a donné les séquences d'échappement pour les `cmdtool` et `shelltool` de Sun.

Jean-Albert Ferrez <ferrez@dma.epfl.ch> a signalé un manque de cohérence dans l'utilisation de `"PWD"` et `"$PWD"`, ainsi que de `"\"` et `"\\"`.

Bob Ellison <papillo@hpellis.fc.hp.com> et Jim Searle <jims@broadcom.com> ont testé `dtterm` sur HP-UX.

Teng-Fong Seak <seak@drfc.cad.cea.fr> a suggéré l'option `-s` de `hostname`, l'utilisation de `` `pwd` ``, et de `echo` sous `csh`.

Trilia <trilia@nmia.com> a suggéré les exemples dans d'autres langages.

Brian Miller <bmiller@telstra.com.au> a fourni les séquences d'échappement et les exemples pour `hpterm`.

Lenny Mastrototaro <lenny@click3x.com> a expliqué l'utilisation des séquences xterm dans les émulateurs de terminaux Irix.

Paolo Supino <paolo@init.co.il> a suggéré l'utilisation de `\\$` dans le prompt de `bash`.