

How to change your shell prompt.

In my last [article](#) I discussed how to change your shell. Now let's change your prompt to make it helpful.

Here are what some of my prompts look like:

```
brendhan@TheBaron>$ <--My Bourne shell
James@TheBaron:/home/James> <-- My Korn shell
[bhorne@TheBaron:bhorne, 10:00 PM, Wed Nov 27] $ <-- My BASH shell
TheBaron darcon /home/darcon[11:17PM]$ <-- My Z shell
[10:02pm] TheBaron[/root] > <--My TC shell
louis@TheBaron /home/louis -> <-- My C shell
```

Making a custom shell prompt is not too difficult. Some shells have a lot more options than others. First find out what type of shell you have:

```
echo $SHELL
```

```
/bin/sh : This is the Bourne shell.
/bin/ksh93 : This is the Korn shell.
/bin/bash : This is the Bash shell.
/bin/zsh : This is the Z shell.
/bin/csh : This is the C Shell.
/bin/tcsh : This is the TC Shell
```

Now let us deal with the files you will need to edit:

```
For Bourne it is: .shrc
For Korn it is: .kshrc or .profile
For Bash it is: .bashrc or .bash_profile
For Z it is: .zshrc
For C it is: .cshrc
For TC it is: .tcshrc or .cshrc
```

Okay now let's deal a little with the shells.

All edits of files shown here take place in the home directory. If you are not sure if you are in your home directory do this:

```
cd ~
```

That will take you to your home directory. You can then do this:

```
pwd
```

To see what is your home directory.

The quick link section. You can fast forward right to the part you want if you really don't want to read the whole thing.

[Bourne Shell](#)

[Korn Shell](#)

[BASH](#)

[Z Shell](#)

[TC Shell](#)

[C Shell](#)

[The Prompt Table](#)

[Color Prompts](#)

[ANSI Color Codes](#)

[Bourne Color](#)

[Korn Color](#)

[BASH Color](#)

[Z Color](#)

[TC Color](#)

[C Color](#)

[Lazy Shell Change](#)

[Bourne Shell](#)

[Korn Shell](#)

[BASH](#)

[Z Shell](#)

[TC Shell](#)

[C Shell](#)

[Lazy Prompt Change](#)

[Bourne Shell](#)

[Korn Shell](#)

[BASH](#)

[Z Shell](#)

[TC/C Shell](#)

The Bourne Shell:

The most limited of the shells. You can only create one set of options for it. Edit your .profile file and you will see this:

```
## set prompt: ``username@hostname$ "  
# PS1=""whoami`@`hostname | sed 's/\..*/"/>"  
# case `id -u` in  
# 0) PS1="${PS1}# ";;  
# *) PS1="${PS1}$ ";;  
# esac
```

Remove the comment marks(#) and make it look like this:

```
## set prompt: ``username@hostname$ "  
PS1=""whoami`@`hostname | sed 's/\..*/"/>"  
case `id -u` in  
0) PS1="${PS1}# ";;  
*) PS1="${PS1}$ ";;  
esac
```

Your Bourne shell will resemble this:

```
whoami@hostname>$
```

the next time you log in. That is about it for the Bourne shell. I have been researching it and there is something called "xcd" that you can use to make it more custom but it is dependant on what version of the Bourne shell you have. Finding out what version you have is really difficult. I did not try this "xcd" setup. If you can help explain xcd in human terms so that I can understand it [I may add it](#)

The Korn Shell:

You will want to edit either your .profile or .kshrc file. If you do not have a .kshrc in your home directory don't worry. You can always create one. Korn shell will go to the .profile if you do a full login and goes to .kshrc if you are just using Korn as an interactive shell (This statement is confusing and I will explain more [later](#)).

Now you can leave your .profile pretty much as is and just change the existing PS1 line to look like this:

```
PS1=`logname`@`hostname -s`:'$PWD>' # set the prompt to display the current directory
```

Your Korn shell will resemble this:

```
whoami@hostname:current working directory>
```

BASH:

You will want to edit either your `.bash_profile` or `.bashrc` file(Interactive non login shell want `.bashrc` explain more on this [later](#)).You will want to add this line if you don't already have it,

```
PS1="[u@\h:\W, \@, \d]> " #custom prompt options
```

```
whoami@hostname\current working directory\time\date
```

Z Shell:

You will want to edit your `.zshrc`. If you don't have one you can make one:

```
ee .zshrc
```

You can add this line to it. Hell it is the only line in mine.

```
PROMPT='%m %n %/[%t]$\` # Custom Prompt settings
```

```
hostname whoami home directory [time]$\`
```

The TC Shell:

You will want to edit your `.tcshrc` file. If you don't have one you can create one. And it is really nice because all you have to do is copy your `.cshrc` file:

```
cp .cshrc .tcshrc
```

Ok now add this line after the line that says "# An interactive shell -- set some stuff up":

```
set prompt = " %B[%@]%\b %m[%/] >" #Custom Prompt settings
```

```
start bold[time] end bold hostname [current working directory] >
```

The C Shell:

You will want to edit your `.cshrc` file. You will want to add after the line that says "# An interactive shell -- set some stuff up":

```
set prompt="`whoami` @ `hostname -s` $cwd ->" #Custom Prompt settings
```

```
whoami @ hostname current working directory ->
```

Finish file edit:

After you have edited the files logout and then log back in. This should show you your custom prompt.

Now a small note FreeBSD does not separate the binaries for TC or C. If you place a `.tcshrc` file in your home directory it will always take precedence over a `.cshrc` file. Basically you can run a `cs` command for a prompt but what you are really doing is running TC with limited parameters. If you want to run a true (ok a close to true) C shell do not place a `.tcshrc` file in the home directory.

Not my style:

What you don't like the options I showed you for your prompt. Fine be that way! I am taking my ball and going home. But I will leave you this really nice table that shows you a lot of the options. Please note the Bourne shell does not have a column because it is a finicky narrow minded shell(and I am not so smart as to be able to figure all of it's unique issues yet).

The Table

:

Description	csch*	ksh	bash	tcsh*	zsh
Current working directory	\$PWD	\$PWD	\w	%/	%/
Current working directory, with one's home directory by `~`	\$PWD:t	\$PWD##*/	\W	%~	%~
Full hostname	'uname -n'	'uname -n'	N/A	%M	%M
Hostname up to the first '.'	`hostname -s`	`hostname -s`	\h	%m	%m
Start (stop) boldfacing mode	%B (or %b)	N/A	N/A	%B (or %b)	%B (or %b)
Start (stop) standout mode	%S (or %s)	N/A	N/A	%S (or %s)	%S (or %s)
Start (stop) underline mode	%U (or %u)	N/A	N/A	%U (or %u)	%U (or %u)
User name	`whoami`	`logname`	\u	%n	%n
The shell's tty that the user is logged in on	%l	N/A	N/A	%	%
The current history number	%h	N/A	\!	%h (or %!)	%h (or %!)
Name of the shell	N/A	N/A	\s	N/A	N/A
Time of day in 12-hour hh:mm AM/PM	%t	N/A	\@	%t (or %@)	%t (or %@)
Time of day in 24-hour hh:mm	%T	N/A	\A	%T	%T
Time of day in 12-hour with seconds hh:mm:ss AM/PM	%p	N/A	\T	%p	N/A
Time of day in 24-hour with seconds hh:mm:ss	%P	N/A	\t	%P	%*
The day in 'dd' format	%D	N/A	N/A	%D	N/A
The month in 'Mon' format	%w	N/A	N/A	%w	N/A
The month in 'mm' format	%W	N/A	N/A	%W	N/A
The year in 'yy' format	%y	N/A	N/A	%y	N/A
The year in 'yyyy' format	%Y	N/A	N/A	%Y	N/A
The date in "Weekday Month Date" format	N/A	N/A	\d	N/A	N/A
The date in day-dd format	N/A	N/A	N/A	N/A	%w
The date in Mon/dd/yy format	N/A	N/A	N/A	N/A	%W
The date in yy-mm-dd format	N/A	N/A	N/A	N/A	%D
The weekday in 'Day' format	%d	N/A	N/A	%d	N/A
Description	csch*	ksh	bash	tcsh*	zsh

*TC and C shell use the same binaries in FreeBSD that means there really isn't a true C shell but that is a whole different topic. Read further in this article you will get a better explanation.

Okay this list is not complete by any means what so ever but I am one person who has to work for a living.If you know of a variable that can be added let me know by adding a comment [here](#). Remember this is for newbies.

Okay so now you have the basic idea behind how to customize a prompt but it can get even fancier.

Color Prompts

Well this will certainly test my HTML to transfer ANSI color codes to an HTML page. But enough of my babbling. Lets give that prompt some color.

```
PS1="\[\033[42m]\[\033[30m][\[\033[31m]\u@\[\033[1;33m]\h:\[\033[1;35m]\W,\[\033[1;34m]\@,\[\033[1;37m]\d\[\033[30m]\[\033[0m]>"
```

Place that puppy in your .bash_profile as one long line and login and don't call me about how you went blind. And no I don't know how to color coordinate but that prompt shows you some of the fun you can have with a prompt if you are working in an environment that supports colors.

```
[bhome@TheBaron:bhome, 10:56 PM Sun Dec 01] >$
```

ANSI Color Codes

Color	Code	Sample
Black	0;30	PROMPT
Red	0;31	PROMPT
Green	0;32	PROMPT
Brown	0;33	PROMPT
Blue	0;34	PROMPT
Purple	0;35	PROMPT
Cyan	0;36	PROMPT
Light Gray	0;37	PROMPT
Dark Gray	1;30	PROMPT
Light Red	1;31	PROMPT
Light Green	1;32	PROMPT
Yellow	1;33	PROMPT
Light Blue	1;34	PROMPT
Pink	1;35	PROMPT
Light Cyan	1;36	PROMPT
White	1;37	PROMPT

Text Background Colors

Color	Code	Sample
Red	0;41	PROMPT
Green	0;42	PROMPT
Brown	0;43	PROMPT
Blue	0;44	PROMPT

Purple	0;45	PROMPT
Cyan	0;46	PROMPT
Gray	0;47	PROMPT

Attributes

Attribute	Code	Sample
Normal Display	0	PROMPT
*Bold	1	PROMPT
Italics	3	<i>PROMPT</i>
Underline	4	<u>PROMPT</u>
*Blink on	5	
*Black Outline	6	PROMPT
Reverse Video on	7	PROMPT
*Non display	8	This is suppose to make text invisible on a colored background
*Strike through	9	PROMPT
*Bold off	22	Read Below
*Italics off	23	Read Below
*Underline off	24	Read Below
*Blink off	25	Read Below
*Inverse off	27	Read Below
*Strike through off	29	Read Below

- * Bold : The effect is not as dramatic as shown it basically makes the prompt a little brighter.
- * Blink on: Warning people find this extremely obnoxious. Don't do it on a public machine.
- * Black Outline: This is so faint you really have to look to be able to see it It does not show up like you see in the image. Also it only works with colored backgrounds.
- * Non display: The official definition is this: The 8m invisible video sequence does not change the background color. In the ANSI.sys driver, this escape sequence sets the foreground to black and the background to black. In ANSIPLUS, the current foreground is set to whatever the current background is, making text invisible on a potentially colored background. Okay but I can't get it to work right for me. If you come up with a cool prompt that can show this let me know here.
- * Strikethrough : The table shows what it is *suppose* to do. I have not been able to get the effect to work. Let me know if you are able to here.
- * Bold off: Turns off bold if you don't want it to go through your whole prompt.
- * Italics off: This turns off italics so it doesn't go through your whole prompt.
- * Underline off: Turns off the underline feature so it doesn't go through your whole prompt.
- * Blink off: Turns off the blink feature so your whole prompt doesn't blink.
- Inverse off: Turns off inverse colors so your whole prompt isn't reversed.
- * Strikethrough off: Ends the strikethrough line so your whole prompt doesn't have a line through it. At least that is the idea Hell I couldn't get the line in the first place.

You will probably note there are a lot of numbers not in the chart. I did to. I started looking around and found out why. They won't work in a prompt. ANSI hasn't even set up all of the missing numbers with features yet and some of those missing numbers are for really advanced features that will serve absolutely no purpose here. Also there are a couple of features like shadow that just don't do a damn thing. I will list all of the pages of stuff I went through for this and if you can make a prompt work with it and keep it simple I will add it. Educate me [here](#).

The Bourne Shell:

The Bourne shell does support colors. But it isn't easy. You have to use the vi editor to make it work. Since giving a vi tutorial is almost a subject on it's own I will not go into a lot of detail here on how to use vi. Understand vi stands for very intimidating editor. But you can't get the symbols you want in the way they need to be to work. I am a newbie and love ee. It is as it's name describes a Easy Editor. Now on to adding colors to your prompt. To color your prompt you will need to place this symbol in your prompt `^[Xm` Replace X with the number of the color you want. Now before I go to the example a few things about why this is a pain in the ass. `^[` Those symbols are made by going into vi and editing and typing `ctrl +v` (pressed at the same time) and then pressing the Esc button. You will note another `[` after the first one you get that by pressing the key above the ' key (qwerty style keyboard). So the keying is very weird. And having to be forced to do it in vi is like being lead through The Twilight Zone. Oh yeah doing a copy/ paste doesn't work either. At least for the multiple times I tried it. Now for the example:

```
PS1="^[31m`whoami`@:`hostname | sed 's/\..*/`^[0m>" # Custom prompt settings
```

`PS1="red text whoami @ hostname normal display>` So basically you have told the computer all text from here to be red. After that you have told the computer to place the hostname and whoami in the prompt. Now return the display to a normal setting. This is important because forgetting to tell it to go back to normal means everything will be red, or worse whatever your last color command is. The point being when you are done make sure you tell it to return to normal.

Okay more than one color:

```
PS1="^[31m`whoami`@^[[1;31m:`hostname | sed 's/\..*/`^[0m>" #Custom Prompt settings
```

```
PS1 = "red text whoami yellow text hostname normal display>
```

Okay background color:

```
PS1="^[46;31m`whoami`@^[[1;31m:`hostname | sed 's/\..*/`^[0m>" #Custom Prompt settings
```

`PS1 = "Cyan background red text whoami yellow text hostname normal display>"` You can have multiple colors in the brackets. Separate them with a semicolon. Also don't ask for two colors to cover the same text. Your computer will get mad at you.

Certain colors need a semicolon in order to appear. Yellow as the above example shows is `1;33` do not use just `33` or it will come out brown. If you have a `0;31` you don't need to place the 0.

These are just some of things you can do. Don't be afraid to experiment. Post your Bourne shell prompt [here](#).

Also there is the issue of [tput](#) but it is also very new to me and I haven't used it. But I have had people make the suggestion of using that. Hey if it works for you go for it.

The Korn Shell:

The Korn shell does support colors. To color the prompt you will want to place this symbol in your prompt. `^\E[Xm'`. Replace X with the number of the color you want. Example:

```
PS1=$^\E[31m`logname`@`hostname -s`:$^\E[0m>' #Custom prompt settings.
```

```
PS1 = "red text whoami hostname normal display>"
```

So basically you have told the computer all text from here to be red. After that you have told the computer to place the hostname and whoami in the prompt. Now return the display to a normal setting. This is important because forgetting to tell it to go back to normal means everything will be red, or worse whatever your last color command is. The point being when you are done make sure you tell it to return to normal.

Okay more than one color:

```
PS1=$^\E[31m`logname`@$^\E[1;33m`hostname -s`:$^\E[0m>' #Custom prompt settings
```

```
PS1 = "red text whoami yellow text hostname normal display>
```

Okay background color:

```
PS1=$'\E[46;31m" logname` @${'\E[1;33m"hostname -s`:${'\E[0m>' #Custom prompt settings
```

PS1="Cyan background red text whoami yellow text hostname normal display>" You can have multiple colors in the brackets. Separate them with a semicolon. Also don't ask for two colors to cover the same text. Your computer will get mad at you.

Certain colors need a semicolon in order to appear. Yellow as the above example shows is 1;33 do not use just 33 or it will come out brown. If you have a 0;31 you don't need to place the 0.

These are just some of things you can do. Don't be afraid to experiment. Post your Korn shell prompt [here](#).

BASH:

Bash does support colors in the prompt. If you used the prompt I showed earlier you know that. Now how you can do it. Place this symbol in front of the section you want to colonize. `\[\033[Xm\]`. Replace the "X" with the number of the color you want. Example:

```
PS1="\[\033[31m\][\u@\h]\[\033[0m\]>"
```

The translation the computer does is this:

```
PS1="red text begin[whoami@hostname]normal display>
```

So basically you have told the computer all text from here to be red. After that you have told the computer to place the username and hostname in the prompt. Now return the display to a normal setting. This is important because forgetting to tell it to go back to normal means everything will be red, or worse whatever your last color command is. The point being when you are done make sure you tell it to return to normal.

Okay more than one color:

```
PS1="\[\033[31m\][\u@\[\033[1;33m\]\h]\[\033[0m\]>"
```

```
PS1="red text[whoami@ yellow text hostname] normal display>
```

Okay background color:

```
PS1="\[\033[46;31m\][\u@\[\033[1;33m\]\h]\[\033[0m\]>"
```

PS1="Cyan background red text [whoami@ yellow text hostname] normal display> You can have multiple colors in the brackets. Separate them with a semicolon. Also don't ask for two colors to cover the same text. Your computer will get mad at you.

Certain colors need a semicolon in order to appear. Yellow as the above example shows is 1;33 do not use just 33 or it will come out brown. If you have a 0;31 you don't need to place the 0.

These are just some of things you can do. Don't be afraid to experiment. Post your BASH prompt [here](#).

Z Shell:

The Z shell does support colors. To color the prompt you will want to do place this symbol in your prompt. `%{\e[Xm%}` Replace X with the number you want for the color you want. Example:

```
PROMPT=${'\e[0;31m% }%m %n %{\e[0m% }$'
```

```
PROMPT= red text hostname whoami normal display
```

So basically you have told the computer all text from here to be red. After that you have told the computer to place the whoami and hostname in the prompt. Now return the display to a normal setting. This is important because forgetting to tell it to go back to normal means everything will be red, or worse whatever your last color command is. The point being when you are done make sure you tell it to return to normal.

Okay more than one color:


```
PROMPT=${'\e[0;31m'}%m %{\e[1;33m'}%n %{\e[0m%}'$'
```

```
PROMPT="red text hostname yellow text whoami normal display>
```

Okay background color:

```
PROMPT=${'\e[0;46;31m'}%m %{\e[1;33m'}%n %{\e[0m%}'$'
```

PROMPT="Cyan background red text hostname yellow text whoami normal display\$" You can have multiple colors in the brackets. Separate them with a semicolon. Also don't ask for two colors to cover the same text. Your computer will get mad at you.

With the Z shell you must include all 0's or 1's in your brackets however you only need to do it once. As in %{\e[0;46;31m%}. Don't be afraid to experiment. Post your Z shell [here](#).

Addendum to Z Shell Color Prompt

If the above example does not work for you, you can try it this way. It is a bit more complicated but should work. Add this to your .zshrc:

```
#Prompt Color Table Z shell
```

```
fg_black=${'\e[0;30m'
```

```
fg_red=${'\e[0;31m'
```

```
fg_green=${'\e[0;32m'
```

```
fg_brown=${'\e[0;33m'
```

```
fg_blue=${'\e[0;34m'
```

```
fg_purple=${'\e[0;35m'
```

```
fg_cyan=${'\e[0;36m'
```

```
fg_lgray=${'\e[0;37m'
```

```
fg_dgray=${'\e[1;30m'
```

```
fg_lred=${'\e[1;31m'
```

```
fg_lgreen=${'\e[1;32m'
```

```
fg_yellow=${'\e[1;33m'
```

```
fg_lblue=${'\e[1;34m'
```

```
fg_pink=${'\e[1;35m'
```

```
fg_lcyan=${'\e[1;36m'
```

```
fg_white=${'\e[1;37m'
```

```
#Text Background Colors
```

```
bg_red=${'\e[0;41m'
```

```
bg_green=${'\e[0;42m'
```

```
bg_brown=${'\e[0;43m'
```

```
bg_blue=${'\e[0;44m'
```

```
bg_purple=${'\e[0;45m'
```

```
bg_cyan=${'\e[0;46m'
```

```
bg_gray=${'\e[0;47m'
```

```
#Attributes
```

```
at_normal=${'\e[0m'
```

```
at_bold=${'\e[1m'
```

```
at_italics=${'\e[3m'
```

```
at_underl=${'\e[4m'
```

```
at_blink=${'\e[5m'
```

```
at_outline=${'\e[6m'
```

```
at_reverse=${'\e[7m'
```

```
at_nondisp=${'\e[8m'
```

```
at_strike=${'\e[9m'
```

```
at_boldoff=${'\e[22m'
```

```
at_italicsoff=${'\e[23m'
```

```
at_underloff=${'\e[24m'
```

```
at_blinkoff=${'\e[25m'
```

```
at_reverseoff=${'\e[27m'
```

```
at_strikeoff=${'\e[29m'
```

To color the prompt you will want to do place the name in your prompt. `{X}` Replace X with the name you want for the color you want. Example:

```
PS1="${fg_red}%m %n ${at_normal}$"
```

```
PS1= red text hostname whoami normal display
```

Two colors:

```
PS1="${fg_red}%m ${fg_yellow}%n ${at_normal}$"
```

```
PS1= red text hostname yellow text whoami normal display
```

Background color:

Well now you have issues. Things like:

```
PS1="${bg_cyan;fg_red}%m ${fg_yellow}%n ${at_normal}$" won't work properly.
```

What you have to do is create a line that mimics what you want. For example:

```
#Mixed Colors
```

```
mx_color0=$'\e[46;31m'
```

```
mx_color1=$'\e[46;1;33m'
```

```
'PS1="${mx_color0}%M ${mx_color1}%n ${at_normal}$"
```

You would add those lines to your `.zshrc`. That would give you the:

```
PS1="Cyan background red text hostname yellow text whoami normal display$"
```

Now basically you can create those lines in any combo you want and call them anything you want. I just set them up like you see for an easy to understand setup. But you could do this:

```
fore_black=$'\e[0;30m'
```

```
black0=$'\e[0;30m'
```

```
darkness=$'\e[0;30m'
```

You get the idea as long as you put the name you created in the parenthesis and you don't make duplicates it can be called anything you want.

TC Shell:

The TC shell does support colors. To color the prompt you will want to place this symbol in your prompt. `%{\033[Xm%}`. Also right after your "set prompt = " you will want to place a `\n` if you don't the color prompts will not work. Example: set prompt = "`\n%{\033[31m%}%m %n %{\033[0m%}>`"

```
set prompt = "red text hostname whoami normal display>"
```

So basically you have told the computer all text from here to be red. After that you have told the computer to place the hostname and whoami in the prompt. Now return the display to a normal setting. This is important because forgetting to tell it to go back to normal means everything will be red, or worse whatever your last color command is. The point being when you are done make sure you tell it to return to normal.

Okay more than one color:

```
set prompt = "\n%{\033[31m%}%m %{\033[1;33m%}%n %{\033[0m%}>"
```

```
set prompt = "red text hostname yellow text whoami normal display>"
```

Okay background color:

```
set prompt = "\n%{\033[46;31m%}%m %{\033[1;33m%}%n %{\033[0m%}>"
```

set prompt = "Cyan background red text hostname yellow text whoami normal display>" You can have multiple colors in the brackets. Separate them with a semicolon. Also don't ask for two colors to cover the same text. Your computer will get mad at you.

Certain colors need a semicolon in order to appear. Yellow as the above example shows is 1;33 do not use just 33 or it will come out brown. If you have a 0;31 you don't need to place the 0.

These are just some of things you can do. Don't be afraid to experiment. Post your TC shell prompt [here](#).

Addendum to TC Shell Color Prompt

I have been pointed out another prompt variable that will also work in TC if the first one doesn't. %^[Xm%} replace X with the color code you want. You will have to go into the vi editor to create ^[this is the ctrl+v keys then the Esc key. You can read more about it [here](#)

C Shell:

The C shell does support colors. To color the prompt you will want to place this symbol in your prompt.%{\033[Xm%}. Also right after your "set prompt = " you will want to place a \n if you don't the color prompts will not work.Example:

```
set prompt = "\n%{\033[31m%}%m %n %{\033[0m%}>"
```

```
set prompt = "red text hostname whoami normal display>"
```

So basically you have told the computer all text from here to be red. After that you have told the computer to place the hostname and whoami in the prompt. Now return the display to a normal setting. This is important because forgetting to tell it to go back to normal means everything will be red, or worse whatever your last color command is. The point being when you are done make sure you tell it to return to normal.

Okay more than one color:

```
set prompt = "\n%{\033[31m%}%m %{\033[1;33m%}%n %{\033[0m%}>"
```

```
set prompt = "red text hostname yellow text whoami normal display>"
```

Okay background color:

```
set prompt = "\n%{\033[46;31m%}%m %{\033[1;33m%}%n %{\033[0m%}>"
```

```
set prompt ="Cyan background red text hostname yellow text whoami normal display>"
```

You can have multiple colors in the brackets. Separate them with a semicolon. Also don't ask for two colors to cover the same text. Your computer will get mad at you.

Certain colors need a semicolon in order to appear. Yellow as the above example shows is 1;33 do not use just 33 or it will come out brown. If you have a 0;31 you don't need to place the 0.

These are just some of things you can do. Don't be afraid to experiment. Post your C shell prompt [here](#).

The Lazy Shell Change

The interactive non login shell.

Changing you shell part 2.

So you don't want to have to log in under a different name to change your shell. Okay. you can do that. Just type in the name of the shell you want to. This part starts from you login shell.

Some notes that are important here. Korn and Bourne share some of the same files when you try to do a non login interactive shell request.To get it to read your .shrc or .kshrc file will depend on a few extra steps. It may seem scary at first. Don't worry you are not breaking anything and none of the changes are permanent.

Bourne Shell:

Okay you log into Bourne and you want to change to a:

Korn Shell: Place a .kshrc file in your home directory. Place your prompt value in that. Now edit your .shrc file and add these two lines at the

bottom.

```
ENV=~/.kshrc # set up Korn shell option
export ENV
```

You should log out and back in the first time you do this. You shouldn't have to after the first time. Then at the prompt type:

```
ksh93
```

You will then have a interactive non login Korn shell. To exit out of it type:

```
exit
```

BASH: Place a .bashrc in your home directory and place your prompt settings in the .bashrc file. then at the prompt type:

```
bash
```

You will then have a interactive non login BASH shell. To exit out of it type:

```
exit
```

Z shell: Place a .zshrc in your home directory and place your prompt settings in the .zshrc file. then at the prompt type:

```
zsh
```

You will then have a interactive non login Z shell. To exit out of it type:

```
exit
```

TC/C Shell: Place a .tcshrc (.cshrc) in your home directory and place your prompt settings in the .tcshrc file. then at the prompt type:

```
tcsh or csh
```

You will then have a interactive non login TC/C shell. To exit out of it type:

```
exit
```

Now a small note FreeBSD does not separate the binaries for TC or C. If you place a .tcshrc file in your home directory it will always take precedence over a .cshrc file. Basically you can run a csh command for a prompt but what you are really doing is running TC with limited parameters.

Korn Shell:

Okay you log into Korn and you want to change to a:

Bourne Shell: Place a .shrc file in your home directory if you don't already have one. Place the prompt in the file as shown earlier in the document. It should already have a PS1 line in there just needed to be uncommented. At the prompt type this:

```
ENV=~/.shrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
sh
```

You will then have a interactive non login Bourne shell. To exit out of it type:

```
exit
```

BASH: Place a .bashrc in your home directory and place your prompt settings in the .bashrc file. then at the prompt type:

bash

You will then have a interactive non login BASH shell. To exit out of it type:

exit

Z Shell: Place a .zshrc in your home directory and place your prompt settings in the .zshrc file. then at the prompt type:

zsh

You will then have a interactive non login Z shell. To exit out of it type:

exit

TC/C Shell: Place a .tcshrc (.cshrc) in your home directory and place your prompt settings in the .tcshrc file. then at the prompt type:

tcsh or csh

You will then have a interactive non login TC/C shell. To exit out of it type:

exit

Now a small note FreeBSD does not separate the binaries for TC or C. If you place a .tcshrc file in your home directory it will always take precedence over a .cshrc file. Basically you can run a csh command for a prompt but what you are really doing is running TC with limited parameters.

BASH:

Okay you log into BASH and you want to change to a:

Bourne Shell: Place a .shrc file in your home directory if you don't already have one. Place the prompt in the file as shown earlier in the document. It should already have a PS1 line in there just needed to be uncommented. At the prompt type this:

```
ENV=~/.shrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
sh
```

You will then have a interactive non login Bourne shell. To exit out of it type:

exit

Korn Shell: Place a .kshrc in your home directory and place your prompt settings in the .kshrc file. then at the prompt type:

```
ENV=~/.kshrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
ksh93
```

You will then have a interactive non login Korn shell. To exit out of it type:

exit

Z shell: Place a .zshrc in your home directory and place your prompt settings in the .zshrc file. then at the prompt type:

```
zsh
```

You will then have a interactive non login Z shell. To exit out of it type:

exit

TC/C Shell: Place a .tcshrc (.cshrc) in your home directory and place your prompt settings in the .tcshrc file. then at the prompt type:

tcsh or csh

You will then have a interactive non login TC/C shell. To exit out of it type:

exit

Now a small note FreeBSD does not separate the binaries for TC or C. If you place a .tcshrc file in your home directory it will always take precedence over a .cshrc file. Basically you can run a csh command for a prompt but what you are really doing is running TC with limited parameters.

Z Shell:

Okay you log into Z shell and you want to change to a:

Bourne Shell: Place a .shrc file in your home directory if you don't already have one. Place the prompt in the file as shown earlier in the document. It should already have a PS1 line in there just needed to be uncommented. At the prompt type this:

```
ENV=~/.shrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
sh
```

You will then have a interactive non login Bourne shell. To exit out of it type:

exit

Korn Shell: Place a .kshrc in your home directory and place your prompt settings in the .kshrc file. then at the prompt type:

```
ENV=~/.kshrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
ksh93
```

You will then have a interactive non login Korn shell. To exit out of it type:

exit

BASH: Place a .bashrc in your home directory and place your prompt settings in the .bashrc file. then at the prompt type:

```
bash
```

You will then have a interactive non login BASH shell. To exit out of it type:

exit

TC/C Shell: Place a .tcshrc (.cshrc) in your home directory and place your prompt settings in the .tcshrc file. then at the prompt type:

tcsh or csh

You will then have a interactive non login TC/C shell. To exit out of it type:

exit

TC Shell:

Okay you log into TC and you want to change to a:

Bourne Shell: Place a .shrc file in your home directory if you don't already have one. Place the prompt in the file as shown earlier in the document. It should already have a PS1 line in there just needed to be uncommented. At the prompt type this:

```
ENV=~/.shrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
sh
```

You will then have a interactive non login Bourne shell. To exit out of it type:

exit

Korn Shell: Place a .kshrc in your home directory and place your prompt settings in the .kshrc file. then at the prompt type:

```
ENV=~/.kshrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
ksh93
```

You will then have a interactive non login Korn shell. To exit out of it type:

exit

BASH:Place a .bashrc in your home directory and place your prompt settings in the .bashrc file. then at the prompt type:

```
bash
```

You will then have a interactive non login BASH shell. To exit out of it type:

exit

Z shell:Place a .zshrc in your home directory and place your prompt settings in the .zshrc file. then at the prompt type:

```
zsh
```

You will then have a interactive non login Z shell. To exit out of it type:

exit

C Shell: You really want to do this. Really. I don't know why but okay. at the prompt type:

```
csh
```

You will then be place in a illusionary C shell. TC and C shells both read the same binaries so it is always going to read the .tcshrc file. You are still working in a TC shell just with limited parameters.

C Shell:

Okay you log into C and you want to change to a:

Bourne Shell: Place a `.shrc` file in your home directory if you don't already have one. Place the prompt in the file as shown earlier in the document. It should already have a `PS1` line in there just needed to be uncommented. At the prompt type this:

```
ENV=~/.shrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
sh
```

You will then have a interactive non login Bourne shell. To exit out of it type:

```
exit
```

Korn Shell: Place a `.kshrc` in your home directory and place your prompt settings in the `.kshrc` file. then at the prompt type:

```
ENV=~/.kshrc
```

Then the prompt will return and type this:

```
export ENV
```

Then the prompt will return and type this:

```
ksh93
```

You will then have a interactive non login Korn shell. To exit out of it type:

```
exit
```

BASH:Place a `.bashrc` in your home directory and place your prompt settings in the `.bashrc` file. then at the prompt type:

```
bash
```

You will then have a interactive non login BASH shell. To exit out of it type:

```
exit
```

Z shell: Place a `.zshrc` in your home directory and place your prompt settings in the `.zshrc` file. then at the prompt type:

```
zsh
```

You will then have a interactive non login Z shell. To exit out of it type:

```
exit
```

TC Shell: Okay if you do this you can mess things up a little. If you have only a `.cshrc` file in your home directory that is what is read when you log in. However in order for you to have a TC shell you will need a `.tcshrc` file. The problem is once you put that in. You `.csh` is left in the dust. Every time you log in that is where it's reading, `.tcshrc` has precedence over `.cshrc`. Okay now for the really bad news. If all you have is a `.cshrc` it doesn't really matter. The TC/C shell are basically the same binaries. C shell just is TC shell with few less functions. Kind of redundant isn't it.

The Lazy Prompt Change

Okay so you don't want to edit your files. Damn that is lazy. Or maybe you just need your prompt to be different for a little while. Well here you go.

Bourne Shell: Okay at your prompt type this:


```
PS1="-->"
```

This will change your prompt to look like this:

```
-->
```

You can always type another style of prompt for what ever floats your boat.

```
PS1="What is thy bidding Master?"
```

What is thy bidding Master?

The changes aren't permanent so if you log out and back in you will have your original prompt back. That's why you edit your files. Because a prompt placed in a file is what you system defaults to unless you change it at the prompt.

Korn: Okay at your prompt type this:

```
PS1="-->"
```

This will change your prompt to look like this:

```
-->
```

You can always type another style of prompt for what ever floats your boat.

```
PS1="What is thy bidding Master?"
```

What is thy bidding Master?

The changes aren't permanent so if you log out and back in you will have your original prompt back. That's why you edit your files. Because a prompt placed in a file is what you system defaults to unless you change it at the prompt.

BASH: Okay at your prompt type this:

```
PS1="-->"
```

This will change your prompt to look like this:

```
-->
```

You can always type another style of prompt for what ever floats your boat.

```
PS1="What is thy bidding Master?"
```

What is thy bidding Master?

The changes aren't permanent so if you log out and back in you will have your original prompt back. That's why you edit your files. Because a prompt placed in a file is what you system defaults to unless you change it at the prompt.

Z Shell: Okay at your prompt type this:

```
PROMPT="-->"
```

This will change your prompt to look like this:

```
-->
```

You can always type another style of prompt for what ever floats your boat.

```
PROMPT="What is thy bidding Master?"
```

What is thy bidding Master?

The changes aren't permanent so if you log out and back in you will have your original prompt back. That's why you edit your files. Because a

prompt placed in a file is what you system defaults to unless you change it at the prompt.

TC/C Shell: Okay at your prompt type this:

```
set prompt="-->"
```

This will change your prompt to look like this:

```
-->
```

You can always type another style of prompt for what ever floats your boat.

```
set prompt="What is thy bidding Master?"
```

What is thy bidding Master?

The changes aren't permanent so if you log out and back in you will have your original prompt back. That's why you edit your files. Because a prompt placed in a file is what you system defaults to unless you change it at the prompt.

The Stuff Near the End

Well damn was that long enough. This the end of the article for now. And to think I originally wanted to look at including X term in it also. Well there definitely was a lot of work researching this puppy. I hope this help you customize your prompts. There were a lot of people and websites that helped me to bring this thing together. Here is that list.

[Jason Neuman](#) For putting up with my articles and not deleting them.

[David Korn](#) For actually replying to my emails with my novice level questions.

[Net_Fish](#) For letting me know that I could customize my prompt.

[drdink](#) For helping me even if it was like pulling teeth to get answers from him.

[comp.unix.bsd.freebsd.misc](#) for letting me post my ridiculous questions to their list. Some of them even got answered.

Special Thanks to the following Jens Schweikhardt, Kristoffer Persson, and jpd of the newsgroup who helped me with the Bourne Shell issue.

[Basalisk](#) in #teknik for being really patient with me in trying to understand the responses I got on the Bourne shell issue.

Umberto Quايا (from the newsgroup) for helping me tackle the echo issue. Which ultimately I edited out because of it's complexity. And for helping me with a bunch of other questions that came up as I was working on this. Michal from comp.unix.bsd.freebsd.misc who helped with the addendums to Z and TC shell.

The list of websites that I found helpful in no particular order:

The ANSI Stuff:

<http://www.bluesock.org/~willg/dev/ansi.html> A little advanced for me but still helpful.

http://pueblo.sourceforge.net/doc/manual/ansi_color_codes.html Very nice with some good explanations about the Bold issue.

[Kristofer Sweger](#) has a great page on ANSI color codes and functions.

[Joe Smith](#) has the really technical stuff about ANSI buried deep in this page. It went way over my head.

[The dircolors](#) man page. Since I couldn't bring it up from my computer. From the [Linuxgazette](#).

Some of the Prompts I came across:

[The Bash Power User prompt](#) So much of this went way beyond me.

[Steve Parker](#) This guy is smart. Here is his tutorial on the Bourne Shell.

[About.com](#) is a great tutorial website . To bad they have that lousy pop up. This page is about shell variables.

[J. Yoon](#) has the best site for custom prompt options. This is the page that inspired me.

[kung-foo.tv](#) had this thing buried deep in their website. It shows some of the TC colored prompt options.

[Daemon News](#) has a page of custom prompts. They are great examples of some very cool very overdone prompts.

[IBM](#) even got into the picture and the site has a really nice color graphic of the ANSI codes. Written by Daniel Robbins.

[Dotfiles.com](#) has a really nice page of prompts.

[Jon C. LeBlanc](#) wrote this article on Generic Unix Interactive Prompts.

[Kuro5hin](#) has a page here asking people for samples of their unique prompts.

[Linuxlookup](#) has a nice how to on BASH with the ANSI color chart.

[Mac OS X Hints](#) has a page for TC shell prompts.

[BSDWiki](#) has a page on custom shell prompts.

[FreeBSD Diary](#) has a page on the BASH prompt.

[FreeBSD Diary](#) has another page here on the TC shell prompt.


That is my list of some of the better webpages I came across. Please check them out. And if you have a comment or a suggestion or a prompt you want to show do so [here](#).

Some notes on page viewing. I have tested this page with IE 5.5, Netscape 4.7, Konqueror and all render fine. Opera does so if the font is Times New Roman and the browser is set to 100%. Lynx does not render the <table> parameters. I have tried to adjust the placement in the tables for clearer viewing, but I can't clear up all of them. So some of the tables look funky under lynx.

Well there is always one jerk in every crowd. When I was trying to find out more about the Bourne shell, there were very many helpful people. And many answered that they did not know or couldn't help me. Which was fine. This guy is so smart that he decided I was clueless. Take a look [here](#).

Valid HTML
4.0!

[Home](#)

 Here is the article as a .pdf

