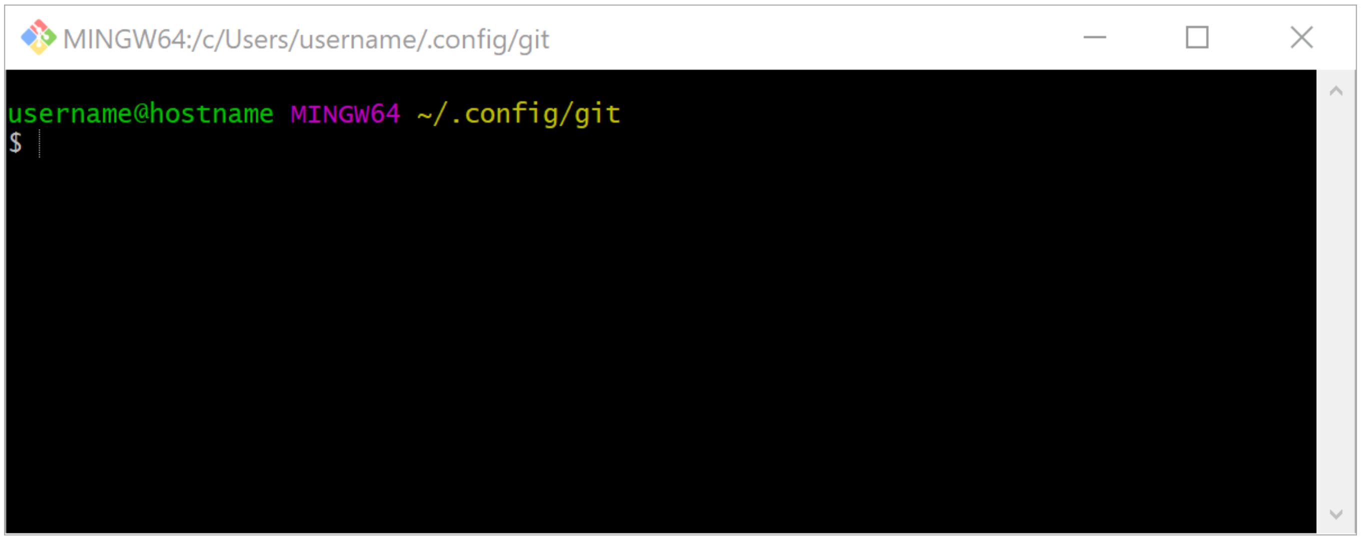
How to Customize the Git Bash Shell Prompt

So you want to hide your username@hostname in the Git Bash prompt? Here's how.

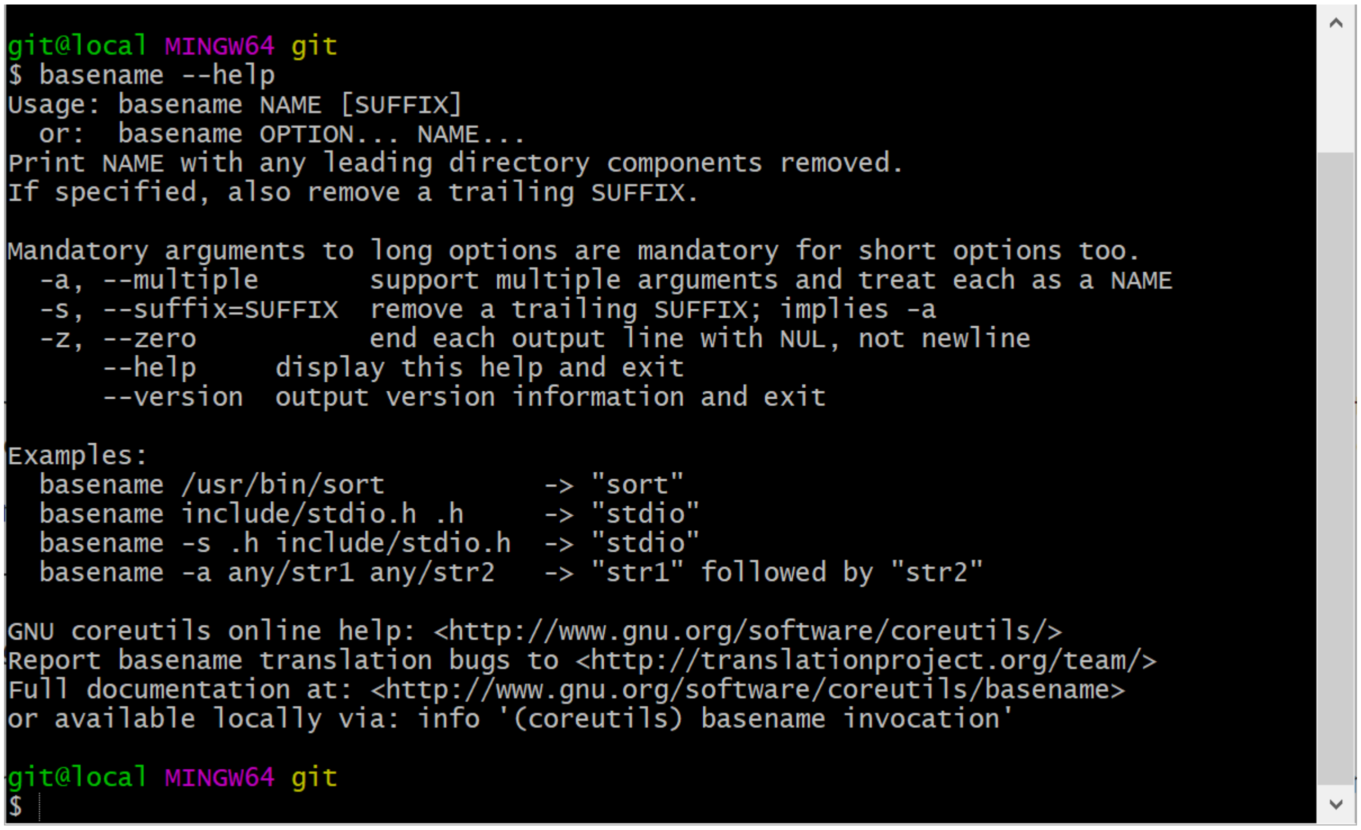
Git for Windows provides lots of native and lightweight tools, as well as a Bash emulation that brings shell experience to users. The default Bash terminal looks like the following.



```
username@hostname MINGW64 ~/.config/git
$
```

The terminal window title contains the full path of the current directory, and the prompt displays the current username, the host name, as well as the path of the current working directory. If the current working directory is a Git repository, the prompt will also show the Git branch name. We should appreciate that the terminal is very informative.

However, sometimes we don't want to show the full path of the current directory or our username and host name. For example, I don't want to expose my username and my machine name in the screenshots in my blog posts. Therefore, I configure the Git Bash terminal window to display a static string "`git@local`" as the username and host name, and only display the `basename` of the current working directory. In this way, I can hide some personal information. My terminal window looks like the following screenshot:



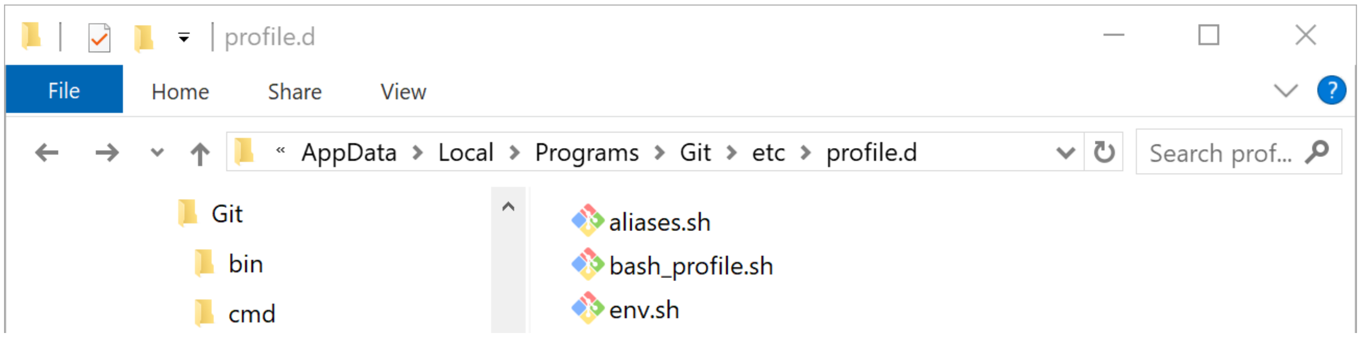
```
git@local MINGW64 git
$ basename --help
Usage: basename NAME [SUFFIX]
  or:  basename OPTION... NAME...
Print NAME with any leading directory components removed.
If specified, also remove a trailing SUFFIX.

Mandatory arguments to long options are mandatory for short options too.
  -a, --multiple       support multiple arguments and treat each as a NAME
  -s, --suffix=SUFFIX  remove a trailing SUFFIX; implies -a
  -z, --zero           end each output line with NUL, not newline
      --help     display this help and exit
      --version  output version information and exit

Examples:
  basename /usr/bin/sort          -> "sort"
  basename include/stdio.h .h     -> "stdio"
  basename -s .h include/stdio.h  -> "stdio"
  basename -a any/str1 any/str2   -> "str1" followed by "str2"

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Report basename translation bugs to <http://translationproject.org/team/>
Full documentation at: <http://www.gnu.org/software/coreutils/basename>
or available locally via: info '(coreutils) basename invocation'

git@local MINGW64 git
$
```

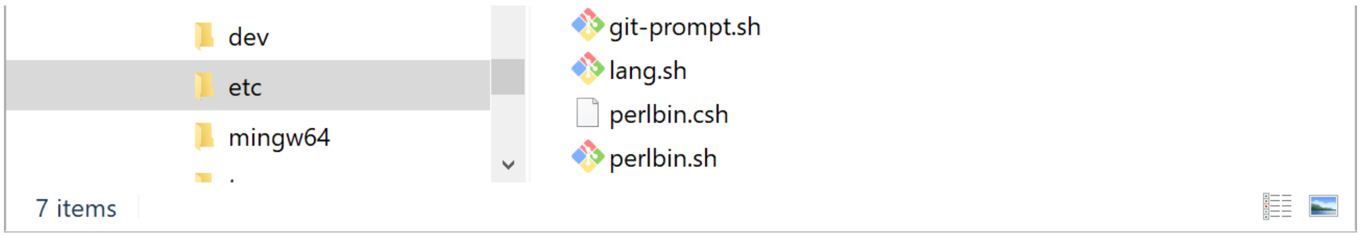
According to an article by Alan P. Barber (link), customizing the Bash prompt is actually very easy.

First, we need to identify the installation path of Git for Windows, and find the profile definition folder by executing the following commands in a Git Bash terminal.

```
cd /etc/profile.d/
explorer .
```

The commands above will open a folder in the file explorer, as shown in the screenshot below.





We are looking for the `git-prompt.sh` file in the `profile.d` folder. The `git-prompt.sh` file contains the configurations for the title of the Git Bash terminal and the Bash prompt string. We can change the colors of the prompt, and modify the prompt content based on our needs. The default `git-prompt.sh` file has the following content.

```
if test -f /etc/profile.d/git-sdk.sh
then
	TITLEPREFIX=SDK-${MSYSTEM#MINGW}
else
	TITLEPREFIX=$MSYSTEM
fi

if test -f ~/.config/git/git-prompt.sh
then
	. ~/.config/git/git-prompt.sh
else
	PS1='\[\033]0;$TITLEPREFIX:$PWD\007\]' # set window title
	PS1="$PS1"'\n'                 # new line
	PS1="$PS1"'\[\033[32m\]'       # change to green
	PS1="$PS1"'\u@\h '             # user@host<space>
	PS1="$PS1"'\[\033[35m\]'       # change to purple
	PS1="$PS1"'$MSYSTEM '          # show MSYSTEM
	PS1="$PS1"'\[\033[33m\]'       # change to brownish yellow
	PS1="$PS1"'\w'                 # current working directory
	if test -z "$WINELOADERNOEXEC"
	then
		GIT_EXEC_PATH="$(git --exec-path 2>/dev/null)"
		COMPLETION_PATH="${GIT_EXEC_PATH%/libexec/git-core}"
		COMPLETION_PATH="${COMPLETION_PATH%/lib/git-core}"
		COMPLETION_PATH="$COMPLETION_PATH/share/git/completion"
		if test -f "$COMPLETION_PATH/git-prompt.sh"
		then
			. "$COMPLETION_PATH/git-completion.bash"
			. "$COMPLETION_PATH/git-prompt.sh"
			PS1="$PS1"'\[\033[36m\]'  # change color to cyan
			PS1="$PS1"'`__git_ps1`'   # bash function
```

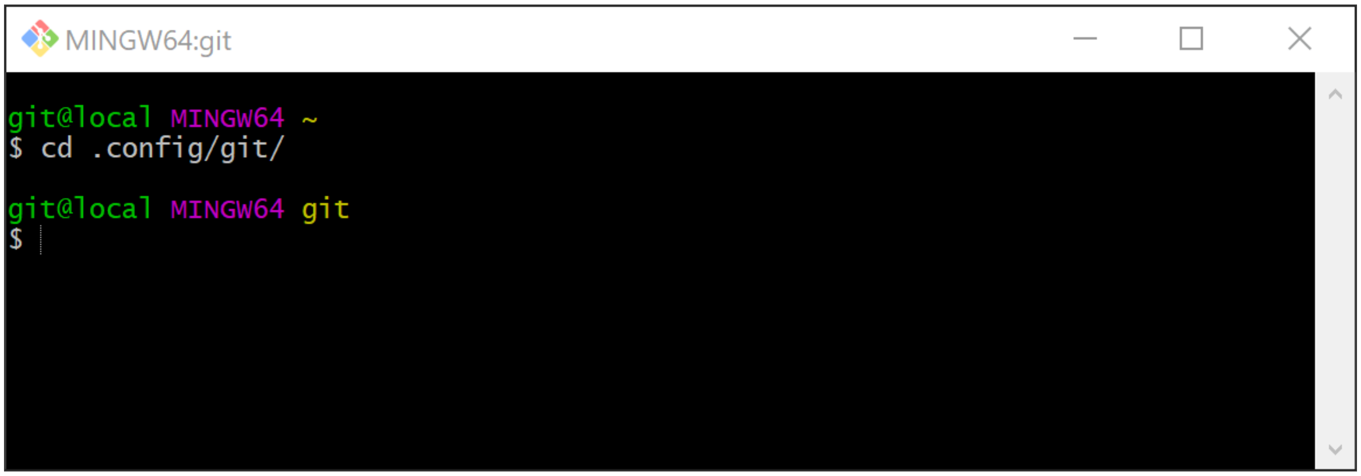
```
                fi
        fi
        PS1="$PS1"'\[\033[0m\]'        # change color
        PS1="$PS1"'\n'                 # new line
        PS1="$PS1"'$ '                 # prompt: always $
fi

MSYS2_PS1="$PS1"               # for detection by MSYS2 SDK's bash.basrc
```

The `git-prompt.sh` file has detailed comments, which greatly helps our customization. We can modify this file to customize our Git Bash terminal, or we can keep this file as it is and save our configurations to our home directory (see lines 8 to 11). I prefer the latter way because I can keep the original `git-prompt.sh` file pristine. Thus, I create a folder `~/.config/git`, and in the folder I create a file `~/.config/git/git-prompt.sh` which will take precedence over the default `git-prompt.sh` file. The following code snippet shows the content of my configuration file.

```
PS1='\[\033]0;$TITLEPREFIX:\W\007\]' # set window title
PS1="$PS1"'\n'                 # new line
PS1="$PS1"'\[\033[32m\]'       # change to green
PS1="$PS1"'git@local '         # user@host<space>
PS1="$PS1"'\[\033[35m\]'       # change to purple
PS1="$PS1"'$MSYSTEM '          # show MSYSTEM
PS1="$PS1"'\[\033[33m\]'       # change to brownish yellow
PS1="$PS1"'\W'                 # current working directory
if test -z "$WINELOADERNOEXEC"
then
        GIT_EXEC_PATH="$(git --exec-path 2>/dev/null)"
        COMPLETION_PATH="${GIT_EXEC_PATH%/libexec/git-core}"
        COMPLETION_PATH="${COMPLETION_PATH%/lib/git-core}"
        COMPLETION_PATH="$COMPLETION_PATH/share/git/completion"
        if test -f "$COMPLETION_PATH/git-prompt.sh"
        then
                . "$COMPLETION_PATH/git-completion.bash"
                . "$COMPLETION_PATH/git-prompt.sh"
                PS1="$PS1"'\[\033[36m\]'  # change color to cyan
                PS1="$PS1"'`__git_ps1`'   # bash function
        fi
fi
```

```
PS1="$PS1"'\[\033[0m\]'        # change color
PS1="$PS1"'\n'                 # new line
PS1="$PS1"'$ '                 # prompt: always $
```

With this file, my terminal prompt looks like this.



```
git@local MINGW64 ~
$ cd .config/git/

git@local MINGW64 git
$
```

We mainly need to customize the `PS1` string (primary prompt). The `PS1` string uses a set of Bash prompt backslash-escaped special characters to display dynamic content. For example, `\u` represents the current user's username, `\w` (lowercase) is the path of the current working directory, `\W` (uppercase) is the basename of the current working directory, and so on. The following table shows all the escape characters that can be used to configure the prompt.

```
\a      an ASCII bell character (07)
\d      the date in "Weekday Month Date" format (e.g., "Tue May 26")
\e      an ASCII escape character (033)
\h      the hostname up to the first '.'
\H      the hostname
\j      the number of jobs currently managed by the shell
\l      the basename of the shell's terminal device name
\n      newline
\r      carriage return
\s      the name of the shell, the basename of $0 (the portion following the final slash)
\t      the current time in 24-hour HH:MM:SS format
\T      the current time in 12-hour HH:MM:SS format
\@      the current time in 12-hour am/pm format
\u      the username of the current user
```

```
\v     the version of bash (e.g., 2.00)
\V     the release of bash, version + patchlevel (e.g., 2.00.0)
\w     the current working directory
\W     the basename of the current working directory
\!     the history number of this command
\#     the command number of this command
\$     if the effective UID is 0, a #, otherwise a $
\nnn   the character corresponding to the octal number nnn
\\     a backslash
\[     begin a sequence of non-printing characters
\]     end a sequence of non-printing characters
```

The construction of the `PS1` string also uses some ANSI color escape codes. The color escape code starts with `\[` and ends with `\]`. The color format is `\033[32m`, where `\033` represents the ASCII escape character, and the code `32` represents foreground color green. We can use the following color codes in the prompt string.

Code	Color
30	Black
31	Red
32	Green
33	Yellow
34	Blue
35	Magenta
36	Cyan
37	White