Print a horizontal line

snipplet: terminal, line
LastUpdate: 2010-07-31
Contributor: Jan Schampera, prince_jammys, ccsalvesen, others
type: snipplet

The purpose of this small code collection is to show some code that draws a horizontal line using as less external tools as possible (it's not a big deal to do it with AWK or Perl, but with pure or nearly-pure Bash it gets more interesting).

In general, you should be able to use this code to repeat any character or character sequence.

Table of Contents

The simple way: Just print it

Not a miracle, just to be complete here.

```
printf '%s\n' ----------
```

The iterative way

This one simply loops 20 times, always draws a dash, finally a newline

```
for ((x = 0; x < 20; x++)); do
  printf %s -
done
echo
```

The simple printf way

This one uses the `printf` command to print an **empty** field with a **minimum field width** of 20 characters. The text is padded with spaces, since there is no text, you get 20 spaces. The spaces are then converted to `-` by the `tr` command.

```
printf '%20s\n' | tr ' ' -
```

whitout an external command, using the (non-POSIX) substitution expansion and `-v` option:

```
printf -v res %20s
printf '%s\n' "${res// /-}"
```

A line across the entire width of the terminal

This is a variant of the above that uses `tput cols` to find the width of the terminal and set that number as the minimum field witdh.

```
printf '%*s\n' "${COLUMNS:-$(tput cols)}" '' | tr ' ' -
```

The more advanced printf way

This one is a bit tricky. The format for the `printf` command is `%.0s`, which specified a field with the **maximum** length of **zero**. After this field, `printf` is told to print a dash. You might remember that it's the nature of `printf` to repeat, if the number of conversion specifications is less than the number of given arguments. With brace expansion `{1..20}`, 20 arguments are given (you could easily write `1 2 3 4 … 20`, of course!). Following happens: The **zero-length field** plus the dash is repeated 20 times. A zero length field is, naturally, invisible. What you see is the dash, repeated 20 times.

```
# Note: you might see that as ''%.s'', which is a (less documented) shorthand for ''%.0s''
printf '%.0s-' {1..20}; echo
```

If the 20 is variable, you can use eval to insert the expansion (take care that using `eval` is potentially dangerous if you evaluate external data):

```
eval printf %.0s- '{1..'"${COLUMNS:-$(tput cols)}"'}'; echo
```

Or restrict the length to 1 and prefix the arguments with the desired character.

```
eval printf %.1s '-{1..'"${COLUMNS:-$(tput cols)}"'}'; echo
```

You can also do it the crazy ormaaj way™ following basically the same principle as this string reverse example. It completely depends on Bash due to its brace expansion evaluation order and array parameter parsing details. As above, the eval only inserts the COLUMNS expansion into the expression and isn't involved in the rest, other than to put the `_` value into the environment of the `_[0]` expansion. This works well since we're not creating one set of arguments and then editing or deleting them to create another as in the previous examples.

```
_=- command eval printf %s '"${_[0]"{0..'"${COLUMNS:-$(tput cols)}"'}"}"'; echo
```

The parameter expansion way

Preparing enough dashes in advance, we can then use a non-POSIX subscript expansion:

```
hr=----------------------------------------------------------------------\
--------------------------------------------------------------------------
printf '%s\n' "${hr:0:${COLUMNS:-$(tput cols)}}"
```

A more flexible approach, and also using modal terminal line-drawing characters instead of hyphens:

```
hr() {
  local start=$'\e(0' end=$'\e(B' line='qqqqqqqqqqqqqqqq'
  local cols=${COLUMNS:-$(tput cols)}
  while ((${#line} < cols)); do line+="$line"; done
  printf '%s%s%s\n' "$start" "${line:0:cols}" "$end"
}
```

Related articles

- The printf command

Discussion

Fernando Basso, 2011/12/27 10:10

What about having the horizontal line being displayed from withing PS1 so that it will show up after every command?

Jan Schampera, 2011/12/28 00:50, 2011/12/28 00:52

Something like this:

```
PROMPT_COMMAND="printf '%.0s-' {1..20};printf '\n'"
```

See PROMPT_COMMAND variable

Krim, 2012/02/10 03:01

Lol, can't resist: just as a joke: (and to be a annoying) my ansi escape solution to print a horizontal line at the top of a screen. (vt100 compatible terminal required)

```
echo $'\e#8\e[1B\e[J'

# #8   == print alignment test pattern
# [1B  == move cursor down 1 row
# [J   == clear screen from cursor downward
```

> /krim

onirix, 2015/07/07 10:12

Hello Krim,

Where can I found a complete list of commands that you pickup these ones ?

Thanks !

onirix, 2015/07/07 10:26

Okey I got it !

http://www.ccs.neu.edu/research/gpc/MSim/vona/terminal/VT100_Escape_Codes.html

Krim, 2012/02/10 03:36

On a slightly related note, my

```
PROMPT_COMMAND="echo $'\e7\e[2;800H\e[16D\e[7m' $(date +"%Y/%m/%d %H:%M") $'\e8'"

# \e7        = Store cursor position and attributes
# \e[2;800H  = Move cursor to top right (well, I don't have any > 800 col ttys)
# \e[16D     = Move cursor left 16 chars
# \e[7m      = Set reverse video
#            Then print a space and the date, and finish with a
# \e8        = Restore cursor position and attributes
```

luka, 2014/01/25 01:54

Not being a C-native, I prefer using a simple echo/tr combination to do the job:

<pre> echo -$nothing{1..80} | tr -d " " </pre>

Provided that $nothing is unset, bash parameter expansion will replace it with nothing - 80 times due to the sequence expression {1..80} Since brace expansion seperates each item with the first character of the IFS delimiter and this is a space by default, we need to delete those spaces using tr -d.

Mind you, I have no idea whether echoing an unset variable is portable or will lead to errors in other shells.

gianni, 2014/01/25 10:55

```
# print multi-${c}haracter ${w}ide Horizontal RoW on stdout hrw () {

  isi "$2" && w="$2" ; isi "$1" && w="$1"
  isc "$2" && c="$2" ; isc "$1" && c="$1"
  w="${w:-${COLUMNS:- tput cols }}"
  c="${c:-—}"
  printf '%*s\n' $w '' | tr ' ' "$c"
  unset w c
}

# IS Integer isi () { egrep -oq '^[0-9]+$' =< "$1" ; }

# IS Character isc () { grep -oq '^[punct]$' =< "$1" ; }
```

Mike, 2015/01/08 00:57

```
printf "%s\b\b\b\b" -{001..25}
```

You could leave a comment if you were logged in.