NAME

echo – echo arguments

SYNOPSIS

/usr/bin/echo [*string . . .* **]**

AVAILABILITY

SUNWcsu

DESCRIPTION

The **echo** utility writes its arguments, separated by BLANKs and terminated by a NEWLINE, to the standard output. If there are no arguments, only the **NEWLINE** character will be written.

echo is useful for producing diagnostics in command files, for sending known data into a pipe, and for displaying the contents of environment variables.

The C shell, the Korn shell, and the Bourne shell each have an **echo** built-in command, which, by default, will have precedence, and will be invoked if the user calls **echo** without a full pathname. See **shell_builtins**(1). **sh**'s **echo**, **ksh**'s **echo**, and **/usr/bin/echo** understand the black-slashed escape characters, except that **sh**'s **echo** does not understand **\ a** as the alert character; however, these commands do not have a –**n** option. **csh**'s **echo** and **/usr/ucb/echo**, on the other hand, have a –**n** option, but do not understand the back-slashed escape characters.

OPERANDS

The following operands are supported:

string A string to be written to standard output. If any operand is "–**n**", it will be treated as a string, not an option. The following character sequences will be recognized within any of the arguments:

\ a	alert character
\ b	backspace
\ c	print line without new-line
\ f	form-feed
\ n	new-line
\ r	carriage return
\ t	tab
\v	vertical tab
****	backslash
\ 0*n*	where *n* is the 8-bit character whose ASCII code is the 1-, 2- or 3-digit octal number representing that character.

USAGE

Portable applications should not use –**n** (as the first argument) or escape sequences.

The **printf**(1) utility can be used portably to emulate any of the traditional behaviours of the **echo** utility as follows:

- The Solaris 2.*x* **/usr/bin/echo** is equivalent to:

  printf "%b\n" "$*"

- The **/usr/ucb/echo** is equivalent to:

```
if [ "X$1" = "X-n" ]
then
      shift
      printf "%s" "$*"
else
      printf "%s\n" "$*"
fi
```

New applications are encouraged to use **printf** instead of **echo**.

EXAMPLES

You can use **echo** to determine how many subdirectories below the root directory (**/**) is your current directory, as follows:

- echo your current-working-directory's full pathname
- pipe the output through **tr** to translate the path's embedded slash-characters into space-characters
- pipe that output through **wc -w** for a count of the names in your path.

example% /usr/bin/echo $PWD | tr '/' ' ' | wc -w

See **tr**(1) and **wc**(1) for their functionality.

Below are the different flavors for echoing a string without a NEWLINE:

/usr/bin/echo **% /usr/bin/echo "$USER's current directory is $PWD\c"**

sh/ksh shells **$ echo "$USER's current directory is $PWD\c"**

csh shell **% echo -n "$USER's current directory is $PWD"**

/usr/ucb/echo **% /usr/ucb/echo -n "$USER's current directory is $PWD"**

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **echo**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following error values are returned:

0 Successful completion.

>0 An error occurred.

SEE ALSO

echo(1B), **printf**(1), **shell_builtins**(1), **ascii**(5), **environ**(5)

NOTES

When representing an 8-bit character by using the escape convention **\0***n*, the *n* must *always* be preceded by the digit zero (0).

For example, typing: **echo ´WARNING:\07´** will print the phrase **WARNING:** and sound the "bell" on your terminal. The use of single (or double) quotes (or two backslashes) is required to protect the " \" that precedes the "07".

Following the **\0**, up to three digits are used in constructing the octal output character. If, following the **\0***n*, you want to echo additional digits that are not part of the octal representation, you must use the full 3-digit *n*. For example, if you want to echo "ESC 7" you must use the three digits "033" rather than just the two digits "33" after the **\0**.

2 digits	Incorrect:	**echo "\0337" | od -xc**	
	produces:	**df0a**	(hex)
		337	(ascii)
3 digits	Correct:	**echo "\00337" | od -xc**	
	produces:	**lb37 0a00**	(hex)
		033 7	(ascii)

For the octal equivalents of each character, see **ascii**(5).