read

Read one line from the standard input, (or from a file) and assign the word(s) to variable name(s).

```
Syntax
      read [-ers] [-a aname] [-p prompt] [-t timeout]
              [-n nchars] [-d delim] [name...]

Key
   -a aname
             The words are assigned to sequential indices of the array variable aname, starting at 0.
             aname is unset before any new values are assigned.  Other name arguments are ignored.

   -d delim
             The first character of delim is used to terminate the input line, rather than newline.

   -e        If the standard input is coming from a terminal, readline is used to obtain the line.

   -n nchars
             read returns after reading nchars characters rather than waiting for a complete line of
             input.
   -p prompt
             Display prompt on standard error, without a trailing newline, before attempting to read
             any input. The prompt is displayed only if input is coming from a terminal.

   -r        Backslash  does not act as an escape character.  The backslash is considered to be part
             of the line. In particular, a backslash-newline pair can not be used as a line continuation.

   -s        Silent mode. If input is coming from a terminal, characters are not echoed.

   -t timeout
             Cause read to time out and return failure if a complete line of input is not read
             within timeout seconds. This option has no effect if read is not reading input from
             the terminal or a pipe.

   -u fd     Read input from file descriptor fd.
```

This is a BASH shell builtin.

One line is read from the standard input, and the first word is assigned to the first *name*, the second word to the second *name*, and so on, with leftover words and their intervening separators assigned to the last *name*.

If there are fewer words read from the standard input than names, the remaining names are assigned empty values.

The characters in the value of the `IFS` variable are used to split the line into words.

The backslash character `` `\' `` can be used to remove any special meaning for the next character read and for line continuation.

If no names are supplied, the line read is assigned to the variable `REPLY`. The return code is zero, unless end-of-file is encountered or `read` times out.

Examples

```
#!/bin/bash
read var_year
echo "The year is: $var_year"

echo -n "Enter your name and press [ENTER]: "
read var_name
echo "Your name is: $var_name"
```

"Programs are meant to be read by humans and only incidentally for computers to execute" ~ Donald Knuth

Related:

select - Accept keyboard input
Equivalent Windows commands: SET /P - Prompt for user input