

eunix: echo

Eric Bailey

October 31, 2017¹

¹ Last updated November 5, 2017

A reimplementaion of echo for my own edification.

1a $\langle * 1a \rangle \equiv$
 $\langle \text{Include headers. } 2a \rangle$

Contents

$\langle \text{Forward declarations. } 2c \rangle$

The main Function 1

$\langle \text{Define the main function. } 1b \rangle$

Include Headers 2

The usage Function 2

$\langle \text{Define the usage function. } 2d \rangle$

Processing Options 2

Root chunk (not used in this document).

-n (do not print a trailing newline) 3

Handling Unknown Options 3

Looping Through Given Options 4

Echoing Strings 4

Full Listing 5

Chunks 6

Index 6

The main Function

1b $\langle \text{Define the main function. } 1b \rangle \equiv$

```
int main(int argc, char *argv[])
{
     $\langle \text{Process given options. } 2e \rangle$ 

     $\langle \text{Print each string, separated by a space. } 4f \rangle$ 

     $\langle \text{Print a newline unless the -n option was given. } 3a \rangle$ 

    return 0;
}
```

This code is used in chunk 1a.

Defines:

argc, used in chunk 4.
argv, used in chunk 4.
main, never used.

Include Headers

Include the core input and output functions from the C standard library.

2a *⟨Include headers. 2a⟩*≡
`#include <stdio.h>`

This definition is continued in chunk 2b.

This code is used in chunk 1a.

Defines:

EOF, used in chunk 4c.

printf, used in chunks 2d and 4e.

putchar, used in chunks 3a and 4d.

Include the GNU **getopt** function from the GNU C Library.

2b *⟨Include headers. 2a⟩*+≡
`#include <getopt.h>`

This code is used in chunk 1a.

Defines:

getopt, used in chunk 4c.

opterr, used in chunk 2e.

optind, used in chunks 3d and 4f.

optopt, used in chunk 4b.

“The **getopt** function gets the next option argument from the argument list specified by the **argv** and **argc** arguments. Normally these values come directly from the arguments received by **main**.” – GNU, 2017

The usage Function

Define the **usage** function, which displays information about how to use **echo**, including *⟨known options 2f⟩*.

2d *⟨Define the usage function. 2d⟩*≡
`void usage()
{
 printf("Usage: echo [-n] [string ...]\n");
}`

This code is used in chunk 1a.

Defines:

usage, used in chunk 2c.

Uses printf 2a.

2c *⟨Forward declarations. 2c⟩*≡
`void usage();`

This code is used in chunk 1a.

Uses usage 2d.

Processing Options

Set **opterr** to 0 to tell **getopt** not to print an error message upon encountering un*⟨known options 2f⟩*.

2e *⟨Process given options. 2e⟩*≡
`opterr = 0;`

This definition is continued in chunks 3c and 4a.

This code is used in chunk 1b.

Uses opterr 2b.

echo accepts -n and prints other options.

2f *⟨known options 2f⟩*≡
`n`

This code is used in chunk 4c.

-n (do not print a trailing newline)

Declare a variable `newline_flag` to determine whether or not to print a newline after printing the rest of the given strings.

3a *⟨Print a newline unless the -n option was given. 3a⟩*≡

```
    if (newline_flag)
        putchar('\n');
```

This code is used in chunk 1b.

Uses `newline_flag` 3c and `putchar` 2a.

When the `-n` option is given, set `newline_flag` to 0, thereby disabling the printing of the trailing newline.

3b *⟨Handle -n. 3b⟩*≡

```
    case 'n':
        newline_flag = 0;
        break;
```

This code is used in chunk 4a.

Uses `newline_flag` 3c.

By default, print a trailing newline.

3c *⟨Process given options. 2e⟩*+≡

```
    int newline_flag = 1;
```

This code is used in chunk 1b.

Defines:

`newline_flag`, used in chunk 3.

Handling Unknown Options

If the user gives an unknown option, i.e. one not included in the *⟨known options 2f⟩*, decrement `optind` by 1 in order to print it later.

3d *⟨Handle unknown options. 3d⟩*≡

```
    case '?':
        optind--;
        break;
```

This code is used in chunk 4a.

Uses `optind` 2b.

“This variable is set by `getopt` to the index of the next element of the `argv` array to be processed.” – GNU, 2017

Looping Through Given Options

```

4a  <Process given options. 2e>+≡
    int c;

    while (<Process known options until EOF. 4b>) {
        switch (c) {
            <Handle -n. 3b>
            <Handle unknown options. 3d>
        }
    }

```

This code is used in chunk 1b.

Defines:

c, used in chunk 4c.

Stop processing options when **optopt** is non-zero.

```

4b  <Process known options until EOF. 4b>≡
    !optopt

```

This definition is continued in chunk 4c.

This code is used in chunk 4a.

Uses **optopt** 2b.

Otherwise, process each known option as **c** until **EOF**.

```

4c  <Process known options until EOF. 4b>+≡
    && (c = getopt(argc, argv, "<known options 2f>")) != EOF

```

This code is used in chunk 4a.

Uses **argc** 1b, **argv** 1b, **c** 4a, **EOF** 2a, and **getopt** 2b.

“When **getopt** encounters an unknown option character... it stores that option character in this variable.” – GNU, 2017

Echoing Strings

Loop through **argv**, starting at **optind**, and <print a space 4d> between 4e each string.

```

4f  <Print each string, separated by a space. 4f>≡
    for (int index = optind; index < argc; index++) {
        <Print the current string. 4e>
        <Print a space unless the current string is the last argument. 4h>
    }

```

This code is used in chunk 1b.

Defines:

index, used in chunk 4.

Uses **argc** 1b and **optind** 2b.

If **index** is less than **argc** - 1 then <the current string is not the last argument 4g>, so <print a space 4d>.

```

4h  <Print a space unless the current string is the last argument. 4h>≡
    if (<the current string is not the last argument 4g>)
        <print a space 4d>

```

This code is used in chunk 4f.

```

4d  <print a space 4d>≡
    putchar(' ');

```

This code is used in chunk 4h.

Uses **putchar** 2a.

```

<Print the current string. 4e>≡
    printf("%s", argv[index]);

```

This code is used in chunk 4f.

Uses **argv** 1b, **index** 4f, and **printf** 2a.

```

4g  <the current string is not the last argument 4g>≡
    index < argc - 1

```

This code is used in chunk 4h.

Uses **argc** 1b and **index** 4f.

Full Listing

```

1  #include <stdio.h>
2  #include <getopt.h>
3
4
5  void usage();
6
7
8  int main(int argc, char *argv[])
9  {
10     opterr = 0;
11
12     int newline_flag = 1;
13
14     int c;
15
16     while (!optopt && (c = getopt(argc, argv, "n")) != EOF) {
17         switch (c) {
18             case 'n':
19                 newline_flag = 0;
20                 break;
21             case '?':
22                 optind--;
23                 break;
24         }
25     }
26
27     for (int index = optind; index < argc; index++) {
28         printf("%s", argv[index]);
29         if (index < argc - 1)
30             putchar(' ');
31     }
32
33     if (newline_flag)
34         putchar('\n');
35
36     return 0;
37 }
38
39
40 void usage()
41 {
42     printf("Usage: echo [-n] [string ...]\n");
43 }

```

Chunks

⟨* [1a](#)⟩ [1a](#)
 ⟨Define the `main` function. [1b](#)⟩ [1a](#), [1b](#)
 ⟨Define the `usage` function. [2d](#)⟩ [1a](#), [2d](#)
 ⟨Forward declarations. [2c](#)⟩ [1a](#), [2c](#)
 ⟨Handle `-n`. [3b](#)⟩ [3b](#), [4a](#)
 ⟨Handle unknown options. [3d](#)⟩ [3d](#), [4a](#)
 ⟨Include headers. [2a](#)⟩ [1a](#), [2a](#), [2b](#)
 ⟨known options [2f](#)⟩ [2f](#), [4c](#)
 ⟨Print a newline unless the `-n` option was given. [3a](#)⟩ [1b](#), [3a](#)
 ⟨print a space [4d](#)⟩ [4d](#), [4h](#)
 ⟨Print a space unless the current string is the last argument. [4h](#)⟩ [4f](#), [4h](#)
 ⟨Print each string, separated by a space. [4f](#)⟩ [1b](#), [4f](#)
 ⟨Print the current string. [4e](#)⟩ [4e](#), [4f](#)
 ⟨Process given options. [2e](#)⟩ [1b](#), [2e](#), [3c](#), [4a](#)
 ⟨Process known options until EOF. [4b](#)⟩ [4a](#), [4b](#), [4c](#)
 ⟨the current string is not the last argument [4g](#)⟩ [4g](#), [4h](#)

Index

`argc`: [1b](#), [4c](#), [4f](#), [4g](#)
`argv`: [1b](#), [4c](#), [4e](#)
`c`: [4a](#), [4c](#)
`EOF`: [2a](#), [4c](#)
`getopt`: [2b](#), [4c](#)
`index`: [4e](#), [4f](#), [4g](#)
`main`: [1b](#)
`newline_flag`: [3a](#), [3b](#), [3c](#)
`opterr`: [2b](#), [2e](#)
`optind`: [2b](#), [3d](#), [4f](#)
`optopt`: [2b](#), [4b](#)
`printf`: [2a](#), [2d](#), [4e](#)
`putchar`: [2a](#), [3a](#), [4d](#)
`usage`: [2c](#), [2d](#)

References

GNU. The GNU C Library: Using the `getopt` function. https://www.gnu.org/software/libc/manual/html_node/Using-Getopt.html, 2017. Accessed: 2017-11-05.