eunix: echo

Eric Bailey

October 31, 2017 [1]

[1] Last updated December 6, 2023

A reimplementation of `echo` for my own edification.

Contents

The `main` Function 1

Include Headers 2

The `usage` Function 2

Processing Options 2

`-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 ⟨*Define the* `main` *function.* 1b⟩≡

```
int main(int argc, char *argv[])
{
    ⟨Process given options. 2e⟩

    ⟨Print each string, separated by a space. 4e⟩

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

    return 0;
}
```

This code is used in chunk 1a.
Defines:
`argc`, used in chunk 4.
`argv`, used in chunk 4.
`main`, never used.

1a ⟨* 1a⟩≡

```
⟨Include headers. 2a⟩

⟨Forward declarations. 2c⟩

⟨Define the main function. 1b⟩

⟨Define the usage function. 2d⟩
```

Root chunk (not used in this document).

Include Headers

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

"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

2a ⟨*Include headers.* 2a⟩≡
```
#include <getopt.h>
```
This definition is continued in chunk 2b.
This code is used in chunk 1a.
Defines:
`getopt`, used in chunk 4b.
`opterr`, used in chunk 2e.
`optind`, used in chunks 3d and 4e.
`optopt`, used in chunk 4b.

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

2b ⟨*Include headers.* 2a⟩+≡
```
#include <stdio.h>
```
This code is used in chunk 1a.
Defines:
`EOF`, used in chunk 4b.
`printf`, used in chunks 2d and 4d.
`putchar`, used in chunks 3a and 4c.

The `usage` Function

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

2c ⟨*Forward declarations.* 2c⟩≡
```
void usage();
```
This code is used in chunk 1a.
Uses `usage` 2d.

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` 2b.

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` 2a.

`echo` accepts `-n` and prints other options.

2f ⟨*known options* 2f⟩≡
```
n
```
This code is used in chunk 4b.

`-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` 2b.

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.

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

3d ⟨*Handle unknown options.* 3d⟩≡
```
case '?':
    optind--;
    break;
```
This code is used in chunk 4a.
Uses `optind` 2a.

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 4b.

Stop processing options when `optopt` is set. Otherwise, process each known option as `c` until `EOF`.

"When `getopt` encounters an unknown option character... it stores that option character in this variable." – GNU, 2017

4b ⟨*Process known options until* EOF. 4b⟩≡
```
optopt == '?' && (c = getopt(argc, argv, "⟨known options 2f⟩")) ≠ EOF
```
This code is used in chunk 4a.
Uses argc 1b, argv 1b, c 4a, EOF 2b, getopt 2a, and optopt 2a.

4c ⟨*print a space* 4c⟩≡
```
putchar(' ');
```
This code is used in chunk 4g.
Uses putchar 2b.

Echoing Strings

Loop through `argv`, starting at `optind`, and ⟨*print a space* 4c⟩ between each string.

4d ⟨*Print the current string.* 4d⟩≡
```
printf("%s", argv[index]);
```
This code is used in chunk 4e.
Uses argv 1b, index 4e, and printf 2b.

4e ⟨*Print each string, separated by a space.* 4e⟩≡
```
for (int index = optind; index < argc; index++) {
    ⟨Print the current string. 4d⟩
    ⟨Print a space unless the current string is the last argument. 4g⟩
}
```
This code is used in chunk 1b.
Defines:
index, used in chunk 4.
Uses argc 1b and optind 2a.

4f ⟨*the current string is not the last argument* 4f⟩≡
```
index < argc - 1
```
This code is used in chunk 4g.
Uses argc 1b and index 4e.

If `index` is less than `argc - 1` then ⟨*the current string is not the last argument* 4f⟩, so ⟨*print a space* 4c⟩.

4g ⟨*Print a space unless the current string is the last argument.* 4g⟩≡
```
if (⟨the current string is not the last argument 4f⟩)
    ⟨print a space 4c⟩
```
This code is used in chunk 4e.

Full Listing

```c
#include <getopt.h>
#include <stdio.h>

void usage();

int main(int argc, char *argv[])
{
    opterr = 0;

    int newline_flag = 1;

    int c;

    while (optopt == '?' && (c = getopt(argc, argv, "n")) != EOF) {
        switch (c) {
        case 'n':
            newline_flag = 0;
            break;
        case '?':
            optind--;
            break;
        }
    }

    for (int index = optind; index < argc; index++) {
        printf("%s", argv[index]);
        if (index < argc - 1)
            putchar(' ');
    }

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

    return 0;
}

void usage()
{
    printf("Usage: echo [-n] [string ...]\n");
}
```

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, 4b
⟨*Print a newline unless the* `-n` *option was given.* 3a⟩ 1b, 3a
⟨*print a space* 4c⟩ 4c, 4g
⟨*Print a space unless the current string is the last argument.* 4g⟩ 4e, 4g
⟨*Print each string, separated by a space.* 4e⟩ 1b, 4e
⟨*Print the current string.* 4d⟩ 4d, 4e
⟨*Process given options.* 2e⟩ 1b, 2e, 3c, 4a
⟨*Process known options until* `EOF`. 4b⟩ 4a, 4b
⟨*the current string is not the last argument* 4f⟩ 4f, 4g

Index

`argc`: 1b, 4b, 4e, 4f
`argv`: 1b, 4b, 4d
`c`: 4a, 4b
`EOF`: 2b, 4b
`getopt`: 2a, 4b
`index`: 4d, 4e, 4f
`main`: 1b
`newline_flag`: 3a, 3b, 3c
`opterr`: 2a, 2e
`optind`: 2a, 3d, 4e
`optopt`: 2a, 4b
`printf`: 2b, 2d, 4d
`putchar`: 2b, 3a, 4c
`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.