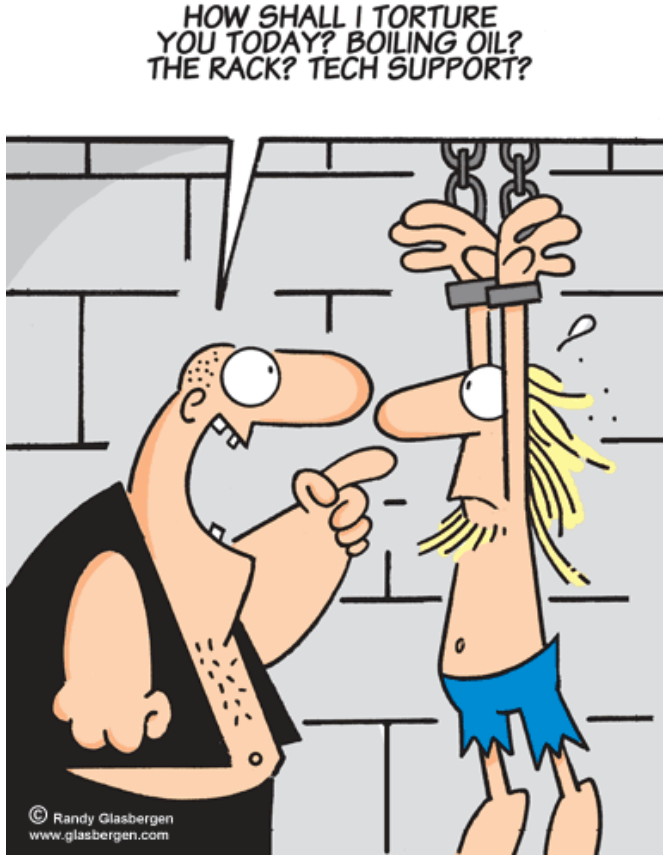
Applied Programming

Command line parsing with

getopt & getopt_long_only

Modern Command line Parsing

- The standard “C” command line parsing tools are torture to use due to all the error checking required!
  - int main(int argc, char *argv[])



http://s-media-cache-ak0.pinimg.com/originals/72/08/80/720880586aca8c04f14d2c7ca7979cae.gif

- There must be a better way
  - Use standard libraries
  - GNU getopt
  - GNU getopt_long_only

getopt()

- Getopt() is an **iterative** **single character** command line option parsing function utilizing global variables and short hand function call options. It supports boolean arguments, optional arguments with parameters and arbitrary additional arguments.

- E.g: myprog -a –b –c my.txt foo.txt

-a, -b	a & b are boolean arguments
-c my.txt	my.txt is the argument for “c”
foo.txt	Is an arbitrary argument

getopt() global variables

Predefined Variable	Description
int opterr	How to handle unknown or invalid parameters. If non-zero - print an error message to the standard error (default) If zero - Don't print any messages, but return the character '?' to indicate an error.
int optopt	If an invalid option is specified or a required option is missing, this variable will contain the offending character option. (note: "int" not "char")
int optind	The index of the next argv element to be processed. This variable can be used to determine where the remaining non-option arguments begin. The initial value is 1.
char *optarg	pointer to the option argument (if any)

Requires: unistd.h.

getopt() function call

int getopt (int argc, const char *argv[], const char *options)

Parameter	Description
argc and argv	The normal "main" parameters
options	A string that specifies the valid option characters for this program. An option character can be followed by a colon **":"** to indicate that it takes a **required** argument. An option character can be followed by two colons **"::"** indication the argument is **optional** e.g. "abc:" - There are 3 parameters, -a –b & -c. -c has a required argument "abc::" - There are 3 parameters, -a –b & -c. -c has a optional argument

Using int getopt()

- Normally called in a while() loop
- Returns the option character processed OR -1 when all options have been process.
  - Note: There may still be non-option arguments, compare optind against argc to check.
- If an option has an argument it is returned in optarg.
  - This will point to argv[] or **NULL** for none.
- If getopt finds an invalid option it returns '?' and sets the optopt to the invalid option character.

Getopt example 1/4

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main (int argc, char **argv) {
  int aflag = 0;                /* My boolean variables */
  int bflag = 0;
  char *cvalue = NULL;          /* Pointer to a parameter */
  int index;
  int rc;                       /* Result from getopt() */

  opterr = 0;           /* Disable automatic error reporting */
```

Getopt example 2/4

```
/* Parse until all parameters have been processed.
   “c” has an required parameter */
while ((rc = getopt (argc, argv, "abc:")) != -1) {
    printf(“getopt() returned =‘%c’\n”,  rc);
    switch (rc)   {
      case 'a':                  /* Boolean variable */
        aflag = 1;
        break;

      case 'b':                   /* Boolean variable */
        bflag = 1;
        break;

      case 'c':                  /* Copy the required C parameter */
        cvalue = optarg;     /* Safe to copy just the pointer because it  */
        break;                      /* really points to argv[]  */
```

Getopt example 3/4

```
case '?':                              /* Handle the error cases */
  if (optopt == 'c') {
    fprintf (stderr, "Option -%c requires an argument.\n", optopt);}

  else if (isprint (optopt)) {  /* character is printable */
    fprintf (stderr, "Unknown option `-%c'.\n", optopt);}

  else {                               /* Character is NOT printable   */
    fprintf (stderr,  "Unknown option character '%x'.\n", optopt); }
  return 1;

default:                               /* oops,  unexpected result */
  fprintf (stderr, “Unexpected result %xX at line %d\n”,
                                                    rc, __LINE__);
  exit(99);
}}
```

Getopt example 4/4

```
/* Print out the final results */
  printf ("aflag = %d, bflag = %d, cvalue = %s\n",
                  aflag, bflag, cvalue);

/* account for any extra arguments */
  for (index = optind; index < argc; index++) {
    printf ("Non-option argument %s\n",
                                        argv[index]); }
  return 0;}
```

Getopt examples

./getopt -a -c asdf

getopt() returned ='a'
getopt() returned ='c'
aflag = 1, bflag = 0, cvalue = asdf

./getopt -a -c

getopt() returned ='a'
getopt() returned ='?'
Option -c requires an argument.

Error message must be generate by our code because opterr was set to 0

./getopt -a -c asdf qwer

getopt() returned ='a'
getopt() returned ='c'
aflag = 1, bflag = 0, cvalue = asdf
Non-option argument qwer

See appendix for the full code

Exercise 1

- What would **getopt(argc, argv, “dog::")** be expected to parse?

- Look for the Boolean arguments “d” & “o” and the an optional argument after “g”

Exercise 2

- What would **getopt(argc, argv, “c:a:t::")** be expected to parse?

- Look for the required arguments after “c” & “a” and the an optional argument after “t”

getopt_long_only()

- getopt() only handles SINGLE character parameters which is not user friendly.
  - E.g ./getopt –a
- getopt_long_only() extends the getopt() function to enable short AND **LONG command line parameters** with the same syntax style as getopt()
  - E.g ./getopt_long_only –add

Requires “getopt.h”

Note: getopt_long() uses a “--” syntax

getopt_long “struct option”

- Describes a single long option name.

Element	Description
const char *name	The name of the option
int has_arg	Indicate the option takes an argument. Choices are: **no_argument**, **required_argument** and **optional_argument.**
int *flag int val	These two fields control how to report or act on an option when it occurs. If flag **is null** then the val is a value which identifies this option. If flag **is non-null** then it must be the address of an int variable which is the flag for this option. The value in val is the value to store in the flag to indicate that the option was seen.

getopt_long_only() function call

int getopt_long_only (int argc, const char *argv[], const char *shortopts, const struct option *longopts, int *indexptr)

Parameter	Description
argc and argv	The normal "main" parameters
shortopts	Same as getopt "options". E.g. "abc:"
struct option *longopts	Pointer to an array of "struct option" objects with the last element identified with the value "0,0,0,0"
int *indexptr	Pointer to a variable containing the longopts[] index. getopt_long_only stores the index of the longopts definition in indexptr. You can get the name of the option with longopts[*indexptr].name. Note: May not be valid if function returns non-zero
Return value	The shortopts character found OR 0 if there is ONLY a longopts parameter

getopt_long_only example 1/5

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

int main (int argc, char **argv) {
  int rc;

  /*-----------------------------------------------------------------
    These variables are used to control the getOpt_long_only
     command line parsing utility.
  -----------------------------------------------------------------*/
  /* getopt_long stores the option index here. */
  int option_index;

  /* Flag set by -verbose & -brief, note initialized to -1 to track usage */
  int verbose_flag = -1;
```

getopt_long_only example 2/5

```
 /* This contains the short command line parameters list   In general
  they SHOULD match the long parameter but DONT HAVE TO     e.g:  verbose  AND  g    */
char *getoptOptions = "abc:d:f:g";

/* This contains the long command line parameter list, it should mostly   match the short list  */
struct option long_options[] = {
 /* These options set the same flag. */
 {"verbose", no_argument,       &verbose_flag, 1},
 {"verb",    no_argument,       &verbose_flag, 1},
 {"brief",   no_argument,       &verbose_flag, 0},

 /* These options don't set a flag.
    We distinguish them by their indices. */
 {"add",       no_argument,         0, 'a'},
 {"append",  no_argument,         0, 'b'},
 {"delete",   required_argument, 0, 'd'},
 {"del",      required_argument,  0, 'd'},
 {"create",   required_argument,  0, 'c'},
 {"file",      required_argument,  0, 'f'},
 {0, 0, 0, 0} /* Terminate!! */
};
```

You can use more than one string for then same option

← CRITICAL!

getopt_long_only example 3/5

```
while ((rc = getopt_long_only(argc, argv, getoptOptions, long_options,
&option_index)) != -1)  {
     printf("getopt_long_only() returned ='%c' index = '%d'\n",  rc, option_index);
    switch (rc) {
     case 0:  /* long option only */
       /* If this option set a flag, do nothing else now. */
       if (long_options[option_index].flag != 0) {
       break; }
       printf ("option %s", long_options[option_index].name);
       if (optarg) {
       printf (" with arg %s", optarg);}
       printf ("\n");
       break;

     case 'a':
       puts ("option -a\n");
       break;
```

getopt_long_only example 4/5

```
case 'b':
  puts ("option -b\n");
  break;

case 'c':
  printf ("option -c with value `%s'\n", optarg);
  break;

case 'd':
  printf ("option -d with value `%s'\n", optarg);
  break;

case 'f':
  printf ("option -f with value `%s'\n", optarg);
  break;

case 'g':
  printf ("option -g\n");
  break;
```

getopt_long_only example 5/5

```
      case '?':
        /* getopt_long_only already printed an error message.
           because  opterr was NOT set to  0*/
        break;

      default:
        printf ("error: undefined option %0xX\n", rc);
      } } // End switch

/* Verbose status */
printf("verbose flag is %d\n", verbose_flag);

/* Print any remaining command line arguments (not options). */
if (optind < argc) {
    printf ("non-option ARGV-elements: ");
    while (optind < argc) {
    printf ("%s ", argv[optind++]);}
    putchar ('\n');
  } // End if

exit (0); }
```

getopt_long_only example 1

/gol -g
getopt_long_only() returned ='g' index = '0'
option -g
verbose flag is -1

Notice the index is zero but that is really “verbose”

Using a sentential value is a handy way to identify unused parameters

./gol -g -verb
getopt_long_only() returned ='g' index = '0'
option -g
getopt_long_only() returned ='' index = '1'
verbose flag is 1

Notice the index is correct

See appendix for the full code

getopt_long_only example 2

./gol -add

getopt_long_only() returned ='a' index = '3'

option -a

verbose flag is -1

./gol -a

getopt_long_only() returned ='a' index = '0'

option -a

verbose flag is -1

See appendix for the full code

getopt_long_only example 3

See appendix for the full code

./gol -asdf

getopt_long_only() returned ='a' index = '0'

option -a

./gol: unrecognized option '-sdf'

System generated error opterr was not set to zero

getopt_long_only() returned ='?' index = '0'

./gol -add something

getopt_long_only() returned ='a' index = '3'

option -a

non-option ARGV-elements: something

Subtle bug

- You are given the following code fragment:

```
char *getoptOptions = "s:";
struct option long_options[] =
{"step",  required_argument, 0, 's'},
case 's':   size = atof(optarg);
```

  - You enter: -step 2 or -s 2 size will be set to 2
  - You enter: **-stepper 2** size will be 0!! **Why?**

- "stepper" does not match the long form but does match "–s:" so the invalid command is evaluated as **–stepper** and atof("**tepper**") is zero!

- If you checked (**optind < argc**) you would find an "extra" argument of "**2**"

Appendix

Full getopt() example

```
/*---------------------------------------------------------------------------
  Example of getop function call
  07/11/2017    R. Repka    First release
---------------------------------------------------------------------------*/
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main (int argc, char **argv) {
  int aflag = 0;        /* My boolean variables */
  int bflag = 0;
  char *cvalue = NULL;  /* Pointer to a parameter */
  int index;
  int rc;               /* Result from getopt() */

  opterr = 0;           /* Disable automatic error reporting */
/* Parse until all parameters have been processed.   “c” has an required parameter */
while ((rc = getopt (argc, argv, "abc:")) != -1) {
    printf("getopt() returned ='%c'\n",  rc);
    switch (rc)   {
      case 'a':    /* Boolean variable */
        aflag = 1;
        break;

      case 'b':                         /* Boolean variable */
        bflag = 1;
        break;

      case 'c':     /* Copy the required C parameter */
        cvalue = optarg;     /* Safe to copy just the pointer because it  */
        break;                          /* really points to argv[]  */

      case '?':     /* Handle the error cases */
        if (optopt == 'c') {
          fprintf (stderr, "Option -%c requires an argument.\n", optopt);}

        else if (isprint (optopt)) {  /* character is printable */
          fprintf (stderr, "Unknown option `-%c'.\n", optopt);}

        else {      /* Character is NOT printable   */
          fprintf (stderr,  "Unknown option character '%x'.\n", optopt); }
        return 1;

      default:      /* oops,  unexpected result */
        fprintf (stderr, "Unexpected result %xX at line %d\n", rc, __LINE__);
        exit(99);
      }}

  /* Print out the final results */
  printf ("aflag = %d, bflag = %d, cvalue = %s\n",
                aflag, bflag, cvalue);

 /* account for any extra arguments */
  for (index = optind; index < argc; index++) {
    printf ("Non-option argument %s\n",
                                      argv[index]); }
  return 0;}
```

Full getopt_long_only() example

```
/*----------------------------------------------------------------------------
  Example of getop_long_only function call
  07/11/2017   R. Repka   First release
----------------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>

int main (int argc, char **argv) {
  int rc;

  /*----------------------------------------------------------------------------
    These variables are used to control the getOpt_long_only command line
    parsing utility.
   ----------------------------------------------------------------------------*/
  /* getopt_long stores the option index here. */
  int option_index;

  /* Flag set by -verbose & -brief, note initialized to -1 to track usage */
  int verbose_flag = -1;

  /* This contains the short command line parameters list   In general
     they SHOULD match the long parameter but DONT HAVE TO
     e.g:  verbose  AND  g    */
  char *getoptOptions = "abc:d:f:g";

  /* This contains the long command line parameter list, it should mostly
     match the short list                                                  */
  struct option long_options[] = {
    /* These options set the same flag. */
    {"verbose", no_argument,       &verbose_flag, 1},
    {"verb",    no_argument,       &verbose_flag, 1},
    {"brief",   no_argument,       &verbose_flag, 0},

    /* These options don't set a flag.
       We distinguish them by their indices. */
    {"add",     no_argument,       0, 'a'},
    {"append",  no_argument,       0, 'b'},
    {"delete",  required_argument, 0, 'd'},
    {"del",     required_argument, 0, 'd'},
    {"create",  required_argument, 0, 'c'},
    {"file",    required_argument, 0, 'f'},
    {0, 0, 0, 0} /* Terminate */
  };

  while ((rc = getopt_long_only(argc, argv, getoptOptions, long_options, &option_index)) != -1)  {
      printf("getopt_long_only() returned ='%c' index = %d\n",  rc, option_index);

     switch (rc) {
       case 0:
         /* If this option set a flag, do nothing else now. */
         if (long_options[option_index].flag != 0) {
         break; }
         printf ("option %s", long_options[option_index].name);
         if (optarg) {
         printf (" with arg %s", optarg);}
         printf ("\n");
         break;

       case 'a':
         puts ("option -a\n");
         break;

       case 'b':
         puts ("option -b\n");
         break;

       case 'c':
         printf ("option -c with value `%s'\n", optarg);
         break;

       case 'd':
         printf ("option -d with value `%s'\n", optarg);
         break;

       case 'f':
         printf ("option -f with value `%s'\n", optarg);
         break;

       case 'g':
         printf ("option -g\n");
         break;

       case '?':
         /* getopt_long already printed an error message.
              because  opterr was NOT set to  0*/
         break;

       default:
         printf ("error: undefined option %0xX\n", rc);
       } } // End switch

  /* Verbose status */
  printf("verbose flag is %d\n", verbose_flag);

  /* Print any remaining command line arguments (not options). */
  if (optind < argc) {
     printf ("non-option ARGV-elements: ");
     while (optind < argc) {
     printf ("%s ", argv[optind++]);}
     putchar ('\n');
   } // End if

  exit (0);
}
```