Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Análisis de opciones en C con getopt.h (24227 lectures)

Per **Jesús Roncero Franco**, ***golan*** (http://www.roncero.org)

Creado el 26/04/2002 04:54 modificado el 26/04/2002 04:54

Cuando programamos para consola, muchas veces queremos pasarle opciones al programa, como, por ejemplo, que aumente el nivel de mensajes por pantalla (el típico verbose). La función getopt() nos ofrece una manera fácil de analizar las opciones en nuestros programas en C.

Con este truco veremos cómo se usa mediante un pequeño ejemplo en C.

La biblioteca getopt sirve para analizar los argumentos que se le pasan a un programa en la linea de comando para separar las opciones de los verdaderos parámetros del programa. Es decir, si tenemos un programa que se llama prueba y ejecutamos :

```
$ prueba -v ./directorio/
```

getopt() nos va a separar el argumento (**./directorio/**) de la opción (**-v** o **--verbose**). Veamos cómo:

Como siempre, os recomiendo que os leais la página man de getopt(). Hay que consultar la página man número 3 de getopt, '**man 3 getopt**', ya que la que invocamos normalmente es la del programa getopt que nos permite, al igual que la función, analizar las opciones en nuestros shell scripts.
En ella encontrareis la explicación de todos los parámetros y opciones de la función getopt(). La explicación de la página man es muy críptica y es por eso que vamos a explicar aquí las opciones más comunes y poner un ejemplo para que se entienda bien.

Lo primero que tenemos que hacer es incluir el archivo de cabecera **getopt.h**. El funcionamiento de getopt() es el siguiente:

- Llamamos a la función getopt_long() y le pasamos como parámetros los típicos argc y argv, que contienen el número de argumentos en la llamada al programa, así como una cadena y una estructura en la que vamos a definir las opciones que queremos que tenga el programa.
- getopt_long() devolverá, por cada llamada a la función, la opción que que se haya incluido, hasta que devuelva -1, momento en el cual ya no quedarán opciones por procesar.

El proceso consistirá en repetir la llamada a la función hasta que devuelva -1

Así mismo, getopt_long() reordena *argv* para que, una vez finalizado el análisis de opciones, argv tenga al final todos los argumentos pasados al programa.
De esta manera podemos ir recorriendo todas las opciones del programa, ir actuando en consecuencia, y despues procesar los argumentos del programa como si hubiese sido lo único que se le haya pasado al programa.

Como sé que es un poco lioso de entender, veámoslo con un ejemplo y así lo aclararemos. Este ejemplo es una modificación del programa de ejemplo que viene en la página man en castellano.
Supongamos que tenemos un programa que realice una cierta operación con una serie de ficheros. A este programa queremos que tenga como opciones las siguientes:

- -v ó --verbose : Para añadir más información.
- -o ó --output fichero : Para que genere un registro externo en el fichero indicado
- -h ó --help : Para mostrar el mensaje de ayuda

Bien, lo primero que haremos es crear una variable dónde vamos a recibir lo que devuelve getopt(), una cadena que incluye el formato de opción corta (-h) y una estructura que define el formato de las opciones largas (--help).

```
/* Recibe la opción siguiente */
   int siguiente_opcion;

/* Una cadena que lista las opciones cortas válidas */
   const char* const op_cortas = "hvo:";

/* Un array describiendo los valores largos correctos. */
   const struct option op_largas[] =
   {
     { "help",       0,  NULL,   'h'},
     { "verbose",    0,  NULL,   'v' },
     { "output",     1,  NULL,   'o'},
     { NULL,         0,  NULL,   0   }
   };
```

Veamos: **siguiente_opcion** va a contener el valor que devuelva getopt() con el que sabremos de qué opción se trata.

op_cortas es una cadena de caracteres en la que cada caracter indica cuales van a ser las opciones cortas (-h -o y -v). El caracter que lleve dos puntos detrás **:** va a ser una opción con argumentos. En este caso, va a ser la opción -o que indicará el fichero en el que, hipotéticamente, el programa grabará un registro.

op_largas es una estructura de tipo *option* que va a contener la configuración de las opciones largas. Crearemos un vector por cada opción larga que queramos, donde el primer elemento del vector indicará la opción larga que queramos usar. El segundo indicará: No lleva argumentos esta opción (0), lleva un argumento obligatorio(1), lleva un argumento opcional(2). Para el tercer elemento del vector usaremos NULL, que indicará que lo que queremos que devuelva getopt() sea el cuarto elemento del vector.
Así **{ "output", 1, NULL, 'o'}** indica que la opción larga será **output**, que llevará un argumento obligatorio y que lo que queremos que devuelva getopt() sea **'o'**. Indicamos que ya no tenemos más opciones con una linea especial con NULL y 0.

Veamos el código fuente en C de un programa de ejemplo y pasaremos a comentarlo a contínuación para ver cómo funciona:

```
/* prueba.c */
#include <stdio.h>
#include <getopt.h>
#include <stdlib.h>

const char* nombre_programa;

void imprime_uso ()
{
    printf("Uso: %s opciones [ argumentos ...]\n", nombre_programa);
    printf("    -h  --help                  Enseña esta ayuda\n"
           "    -o  --output fichero        Escribe la salida al fichero\n"
           "    -v  --verbose               Imprime más información a la salida\n");
}

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

  /* Una cadena que lista las opciones cortas válidas */
  const char* const op_cortas = "hvo:" ;

  /* Una estructura de varios arrays describiendo los valores largos */
  const struct option op_largas[] =
  {
      { "help",         0,  NULL,   'h'},
      { "verbose",      0,  NULL,   'v'},
      { "output",       1,  NULL,   'o'},
      { NULL,           0,  NULL,   0  }
```

```
  };

  /* El nombre del fichero que recibe la salida del programa */
  const char* fichero_salida = NULL ;

  /* Bandera a activar si hay que imprimir más información
   * en el modo verbose */
  int verbose = 0;

  /* Guardar el nombre del programa para incluirlo a la salida */
  nombre_programa = argv[0];

  /* Si se ejecuta sin parámetros ni opciones */
  if (argc == 1)
  {
      imprime_uso();
      exit(EXIT_SUCCESS);
  }

  while (1)
  {
      /* Llamamos a la función getopt */
      siguiente_opcion = getopt_long (argc, argv, op_cortas, op_largas, NULL);

      if (siguiente_opcion == -1)
          break; /* No hay más opciones. Rompemos el bucle */

      switch (siguiente_opcion)
      {
          case 'h' : /* -h o --help */
              imprime_uso();
              exit(EXIT_SUCCESS);

          case 'v' : /* -v o --verbose */
              verbose = 1 ;
              break;

          case 'o' : /* -o ó --output */
              fichero_salida = optarg; /* optarg contiene el argumento de -o */
              break;

          case '?' : /* opción no valida */
              imprime_uso(); /* código de salida 1 */
              exit(1);

          case -1 : /* No hay más opciones */
              break;

          default : /* Algo más? No esperado. Abortamos */
              abort();
          }
  }

  if (verbose && optind <argc)
  {
      printf ("elementos de ARGV que no son opciones:\n");
      while (optind  <argc)
          printf ("Argumento %s\n", argv[optind++]);
  }

  /* Imprimimos el fichero de salida */
  if (fichero_salida != NULL )
      printf("Fichero para la salida: %s\n",fichero_salida);

  /* Salida del programa principal */
  return 0;
}
```

Viendo el programa, nos fijamos que lo que hace es un bucle *while* infinito dónde vamos a ir llamando a getopt_long(). Cuando **siguiente_opcion** sea diferente de -1 entonces rompemos el bucle, ya que una vez que sea igual a -1 significa que ya no tenemos más opciones que procesar.
Vemos que dentro del bucle, hacemos un *switch* sobre **siguiente_opcion**, que contiene la opción que estamos procesando en este momento, lo que nos permite actuar en consecuencia, activando las variables oportunas.

Lo que si merece pararnos a señalar es el uso de las variables **optind** y **optarg**.

- optind indica a partir de qué elemento de argv comienzan los argumentos para el programa. Nótese que getopt_long() reordena **argv** colocando al final los argumentos del programa.
- optarg es usado cuando una opción lleva un parámetro, siendo este un puntero a la cadena que contiene ese parámetro.

Comentar también que el programa no hace nada en partícular. Sólo al final, imprimimos por pantalla todos los argumentos del programa si se ha especificado la opción --verbose y se imprime también el nombre del fichero de salida si se ha usado la opción --output fichero.

Visto esto, si compilamos el programa y lo ejecutamos de la siguiente forma obtendremos:

```
./prueba -v -o salida.txt a b c d
elementos de ARGV que no son opciones:
Argumento a
Argumento b
Argumento c
Argumento d
Fichero para la salida: salida.txt
```

Por último, no olvideis consultar la página man si quereis conocer todas las opciones, compatibilidad con POSIX, etc...

E-mail del autor: jesus _ARROBA_ roncero.org
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1290