Using getopt to Parse Arguments in C

Using getop in C to Read Arguments

Having seen our hello world program in a number of forms so far such as the simple hello and then using if we will now extend this further. Using getopt in the C programming language will allow us to pass options to the program in any order. Thinking for a moment about how we compile the source code we use the option -o to specify the output file. Using either of the following examples would work:

```
gcc -o hello hello.c  gcc hello.c -o hello
```

Additionally, we can add in other options, also in any order:

```
gcc -o hello -Wall hello.c  gcc hello.c -o hello -Wall
```

//
//

Using options in a program is a lot easier than having the pass arguments in the correct order from the command line and we will now see how we achieve this in C using getopt.

Using Loops

Viewing the previous code extract you can see that we make use of a looping structure in C, the while loop. Looping in programming allows us to iterate or go though each item one be one. A simple while loop extract is shown below:

```
int a = 10;  while (a
```

```
We can see much like the if statement that we looked at within the
 previous code, the while loop is also a conditional statement. The
 parenthesis enclose the condition that must evaluate to true. Looping
 whilst is true we use printf the print the value of the variable a,
 noting, that the placeholder for an integer is %d. Ensuring the we don't
 loop forever the last line of the loop increments the value of the
 variable, a++. Printing will start at 10 and then continue through to
 19.
```

Using getopt

```
You may look at the while loop we employ and think it is a little more
 complicated, alternatively you may see that it does just a little more.
 Firstly, be sure to include the header for getopt.
```

```
#include
```

```
Each option that we supplied can be in one or more argument to the
program and those arguments are still in the array argv. Consider the
following command:
```

```
gcc -o hello hello.c
```

- **argv[0]** = gcc

- **argv[1]** = -o
- **argv[2]** = hello
- **argv[3]** = hello.c

We need a variable to store the current index value as we iterate through each argument looking for options.

```
int option_index = 0
```

The name or identifier that we assign to the variable is our choice.

When using getop an option, such as -o in the gcc program, can take a value. In the extract of code used in the above screenshot we can provide an argument to the option -u. We need to store that somewhere. For this we setup the user_name array. Both of these variables, option_index and user_name we initialize with a value to prevent warnings from the compiler.

```
char *user_name = NULL
```

To assign the true value to the variable option_index we return the index from argv using the function getopt. The first argument is the argument count, acting as the highest number we need at return. Then the array to iterate through, argv followed by the possible options. Viewing the screenshot we specify the option u and g. The option u takes and argument as it is preceded with the colon, (u:). Our program requires that we state the user name with the option -u.

```
while (( option_index - getopt(argc, argv, "u:g")) != -1)
```

The braces enclose the code block for the while loop.

Case Statements

The last element of our new main function is the case statement. The case statement starts with the keyword switch but we subsequently examine the

value of the option_index use the keyword case from which it is commonly known.

Using the case statement is a great alternative to adding in many elsif statements.

```
switch(option_index) {    case 'u':      user_name = optarg;      printf( "User is selected\n");      printf("This user is: %s\n", user_name);  break;    case 'g':      printf("Group is selected\n");      break;    d efault;      printf("Invalid option\n");  }
```

Viewing the code extract you can see that we search for both the option u and g. It is possible that both may be set. Using the option -u we look for the argument and populate the user_name variable with the option supplied. We use the keyword break to close the case for u before searching for the option h.

Whilst this is still a very simple example as we start looking at our own calculations functions this becomes a valuable building block. We will look at next creating a program to convert Centigrade to Fahrenheit but later will add options similar to this so we can covert in either direction based on the option used.

Complete Source Code

```
#include   #include   int main ( int argc, char **argv) {   int option_in dex = 0;   char *user_name = NULL;   while (( option_index = getopt(argc,  argv, "u:g")) != -1){     switch (option_index) {       case 'u':  user_name = optarg;         printf("User is selected\n");         printf ("The user is %s\n",user_name);         break;       case 'g':         pr intf("Group is selected\n");         break;       default:        printf( "Option incorrect\n");        return 1;       } //end block for switch   }  //end block for while   return 0;  } // end main block
```

The video follows: