C++ version of the GNU getopt function

The GetOpt class provides an efficient and structured mechanism for processing command-line options from an application program. The sample program fragment below illustrates a typical use of the GetOpt class for some hypothetical application program:

```
#include <stdio.h>
#include <GetOpt.h>
//...
int debug_flag, compile_flag, size_in_bytes;

int
main (int argc, char **argv)
{
  // Invokes ctor `GetOpt (int argc, char **argv,
  //                       char *optstring);'
  GetOpt getopt (argc, argv, "dcs:");
  int option_char;

  // Invokes member function `int operator ()(void);'
  while ((option_char = getopt ()) != EOF)
    switch (option_char)
      {
         case 'd': debug_flag = 1; break;
         case 'c': compile_flag = 1; break;
         case 's': size_in_bytes = atoi (getopt.optarg); break;
         case '?': fprintf (stderr,
                            "usage: %s [dcs<size>]\n", argv[0]);
      }
}
```

Unlike the C library version, the libg++ GetOpt class uses its constructor to initialize class data members containing the argument count, argument vector, and the option string. This simplifies the interface for each

subsequent call to member function **int operator ()(void)**.

The C version, on the other hand, uses hidden static variables to retain the option string and argument list values between calls to **getopt**. This complicates the **getopt** interface since the argument count, argument vector, and option string must be passed as parameters for each invocation. For the C version, the loop in the previous example becomes:

```
while ((option_char = getopt (argc, argv, "dcs:")) != EOF)
  // ...
```

which requires extra overhead to pass the parameters for every call.

Along with the GetOpt constructor and **int operator ()(void)**, the other relevant elements of class GetOpt are:

char *optarg

Used for communication from **operator ()(void)** to the caller. When **operator ()(void)** finds an option that takes an argument, the argument value is stored here.

int optind

Index in **argv** of the next element to be scanned. This is used for communication to and from the caller and for communication between successive calls to **operator ()(void)**.

When **operator ()(void)** returns EOF, this is the index of the first of the non-option elements that the caller should itself scan.

Otherwise, **optind** communicates from one call to the next how much of **argv** has been scanned so far.

The libg++ version of GetOpt acts like standard UNIX **getopt** for the calling routine, but it behaves differently for the user, since it allows the user to intersperse the options with the other arguments.

As GetOpt works, it permutes the elements of **argv** so that, when it is done, all the options precede everything else. Thus all application programs are extended to handle flexible argument order.

Setting the environment variable _POSIX_OPTION_ORDER disables permutation. Then the behavior is completely standard.