NAME

subgetopt − get option character from command line

SYNTAX

#include <subgetopt.h>

char ***sgoptarg**;
int **sgoptind**;
int **sgoptpos**;
int **sgoptdone**;
int **sgoptproblem**;

int **sgopt(***argc,argv,opts***)**;

int *argc*;
char ***argv*;
char **opts*;

DESCRIPTION

sgopt returns the next valid command-line option character from *argv*.

Valid option characters are listed in the *opts* string. *opts* may be empty. A character in *opts* may be followed by a colon, in which case it takes an *option argument*. Avoid using the characters ?, :, and − as option characters.

Below *option argument* is abbreviated as *optarg* and *command-line argument* is abbreviated as *cmdarg*.

Options are listed in cmdargs which begin with a minus sign. Several options which do not take optargs may be combined into one cmdarg.

An option which takes an optarg may be handled in two ways. If it appears at the very end of a cmdarg, then the entire next cmdarg is the optarg. But if there are any characters in the cmdarg after the option character, then those characters form the optarg. The optarg is returned in **sgoptarg**. Next time **sgopt** looks at the cmdarg which follows the optarg.

If a cmdarg does not begin with a hyphen, or if it is a lone hyphen not followed by any characters, or if it begins with two hyphens, then it terminates option processing, and **sgopt** returns an appropriate code. If there are two hyphens, **sgopt** will advance attention to the next cmdarg, so it can be called again to read further options.

PROPER USAGE

sgoptproblem should be used only when **sgopt** returns ?. **sgoptind** and **sgoptpos** are defined all the time. **sgoptarg** is defined all the time; it is null unless **sgopt** has just returned an option with optarg.

sgopt is typically used as follows.

```
#include <subgetopt.h>

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

while ((opt = sgopt(argc,argv,"a:s")) != sgoptdone)
  switch(opt) {
    case 'a':
      printf("opt a with optarg %s\n",sgoptarg); break;
    case 's':
      printf("opt s with no optarg\n"); break;
```

```
  case '?':
    if (argv[sgoptind] && (sgoptind < argc))
      printf("illegal opt %c\n",sgoptproblem);
    else
      printf("missing arg, opt %c\n",sgoptproblem);
    exit(1);
 }

argv += sgoptind;
while (*argv) printf("argument %s\n",*argv++);
exit(0);
}
```

The end of the command line is marked by either *argc*, or a null pointer in *argv*, whichever comes first. Normally these two markers coincide, so it is redundant to test for both *argv*[**sgoptind**] and **sgoptind <** *argc*. The above code shows both tests as an illustration.

Multiple option sets: One useful technique is to call **sgopt** with a primary *opts* until it returns EOF, then call **sgopt** with a secondary *opts* until it returns EOF. The user can provide primary options, then a double hyphen, and then secondary options. No special handling is needed if some or all of the options are omitted. The same technique can be used for any number of option sets in series.

Multiple command lines: Before parsing a new **argv**, make sure to set **sgoptind** and **sgoptpos** back to 1 and 0.

PARSING STAGES

sgopt keeps track of its position in *argv* with **sgoptind** and **sgoptpos**, which are initialized to 1 and 0. It looks at *argv*[**sgoptind**][**sgoptpos**] and following characters.

sgopt indicates that no more options are available by returning **sgoptdone**, which is initialized to **SUBGETOPTDONE**, which is defined as −1.

sgopt begins by setting **optarg** to null.

Ending conditions: If *argv* is null, or **sgoptind** is larger than *argc*, or the current cmdarg *argv*[**sgoptind**] is null, then **sgopt** returns **optdone**.

Stage one: If the current character is zero, **sgopt** moves to the beginning of the next cmdarg. It then checks the ending conditions again.

Stage two: If the current position is the begining of the cmdarg, **sgopt** checks whether the current character is a minus sign. If not it returns **optdone**. It then moves to the next character. If that character is zero, **sgopt** moves back to the beginning of the cmdarg, and returns **sgoptdone**. If the character is a minus sign, **sgopt** moves to the beginning of the next cmdarg, and returns **sgoptdone**.

Stage three: sgopt records the current character, *c*, and moves to the next character. There are three possibilities: (1) *c* is an option character without optarg in *opts*, or (2) *c* is an option character with optarg in *opts*, or (3) *c* does not appear in *opts*.

(1) If *c* appears as an option character without optarg in *opts*, **sgopt** returns *c*.

(2) If *c* appears as an option character with optarg in *opts*, **sgopt** sets **sgoptarg** to the current position, and moves to the next cmdarg. If **sgoptarg** is nonempty, **sgopt** returns *c*.

Then **sgopt** sets **sgoptarg** to the current cmdarg. If the current cmdarg is null, or past *argc*, **sgopt** sets

sgoptproblem to *c* and returns ?. Otherwise **sgopt** moves to the next argument and returns *c*.

(2) If *c* does not appear in *opts*, **sgopt** sets **sgoptproblem** to *c* and returns ?.

SYNTAX NOTE

sgopt is actually a macro abbreviation for **subgetopt**. The external **sg** variables are also macros for **subget**. These macros are defined in **<subgetopt.h>**, unless **SUBGETOPTNOSHORT** is defined when **<subgetopt.h>** is included.

VERSION

subgetopt version 0.9, 931129.

AUTHOR

Placed into the public domain by Daniel J. Bernstein.