NAME

getallargs() - parses all the flag-type arguments

SYNOPSIS

#include <schily/getargs.h>

```
int getallargs(pac, pav, fmt, a1, ..., an)
        int *pac;       /* pointer to arg count */
        char *(*pav)[]; /* pointer to address of arg vector */
        char *fmt;      /* format string */
        type *a1;       /* pointer to result 1 */
                        /* (corresponding to the */
                        /* first descriptor in fmt) */
        type *an;       /* pointer to result n */
                        /* (corresponding to the */
                        /* nth descriptor in fmt) */

int getlallargs(pac, pav, props, fmt, a1, ..., an)
        int *pac;       /* pointer to arg count */
        char *(*pav)[]; /* pointer to address of arg vector */
        struct ga_props *props; /* control properties */
        char *fmt;      /* format string */
        type *a1;       /* pointer to result 1 */
                        /* (corresponding to the */
                        /* first descriptor in fmt) */
        type *an;       /* pointer to result n */
                        /* (corresponding to the */
                        /* nth descriptor in fmt) */

int getvallargs(pac, pav, props, vfmt)
        int *pac;       /* pointer to arg count */
        char *(*pav)[]; /* pointer to address of arg vector */
        struct ga_props *props; /* control properties */
        struct ga_flags *vfmt;  /* array of formats and args */
```

DESCRIPTION

getallargs() is part of the advanced option parsing interface together with the **getargs**() and **getfiles**() family.

getallargs() parses all flag (option) arguments (anywhere on the command line). It does not return until all the arguments have been parsed correctly (returning 0), or an error has occurred (returning < 0).

getlallargs() is similar to **getallargs**() but it implements an additional **ga_props** parameter that must be initialized with **getarginit**() before it is passed.

getvallargs() is similar to **getlallargs**() but uses a structure **ga_flags** instead of a format string and a variable arg list with pointers. The array of structures **ga_flags**:

```
struct ga_flags {
  const char  *ga_format; /* Comma separated list for one flag */
  void        *ga_arg;    /* Ptr. to variable to fill for flag */
  getpargfun  ga_funcp;   /* Ptr. for function to call (&/~)   */
};
```

is terminated by an element with **ga_format == NULL**. For a **ga_format** that does not expect a function pointer, **ga_funcp** is **NULL**.

See **getargs**() for a more detailed description of the parameter matching.

RETURNS

NOARGS 0	All arguments have been successfully examined.
BADFLAG -1	A bad flag (option) argument was supplied to the program. The argument ***pav** contains the offending command line argument.
BADFMT -2	A bad format descriptor string has been detected. This means an error in the calling program, not a user input data error.

General rules for the return code:

> 0	A file type argument was found.
0	All arguments have been parsed.
< 0	An error occurred or not a file type argument.

Flag and file arg processing should be terminated after getting a return code <= 0.

SEE ALSO

getargs(3), **getargerror**(3), **getfiles**(3).

NOTES

getallargs() must be called with the address of a count of items in the vector and the address of a pointer to the vector. Both addresses must already have been properly treated in order to skip over the first parameter which is the name of the program. [e.g. **--ac; ++av**].

Since **getallargs**() will destroy these values, copies should be made for later use in the program. If an error occurs, **av[0]** points to the unmatched argument.

The special argument, **"--"**, is ignored, but the following argument in the command line is treated as a literal filename argument. This way, filenames beginning with ’-’, ’+’, or containing ’=’ can be passed to the routine.

BUGS

None currently known.

Mail bugs and suggestions to **schilytools@mlists.in-berlin.de** or open a ticket at **https://codeberg.org/schilytools/schilytools/issues**.

The mailing list archive may be found at:

https://mlists.in-berlin.de/mailman/listinfo/schilytools-mlists.in-berlin.de.

AUTHOR

Joerg Schilling and the schilytools project authors.