

ARGZ_ADD(3)

ARGZ_ADD(3) Linux Programmer's Manual ARGZ_ADD(3)

NAME

argz_add, argz_add_sep, argz_append, argz_count, argz_create, argz_create_sep, argz_delete, argz_extract, argz_insert, argz_next, argz_replace, argz_stringify - functions to handle an argz list

SYNOPSIS

```
#include <argz.h>

error_t argz_add(char **argz, size_t *argz_len, const char *str);

error_t argz_add_sep(char **argz, size_t *argz_len,
                    const char *str, int delim);

error_t argz_append(char **argz, size_t *argz_len,
                   const char *buf, size_t buf_len);

size_t argz_count(const char *argz, size_t argz_len);

error_t argz_create(char * const argv[], char **argz,
                  size_t *argz_len);

error_t argz_create_sep(const char *str, int sep, char **argz,
                      size_t *argz_len);

error_t argz_delete(char **argz, size_t *argz_len, char *entry);

void argz_extract(char *argz, size_t argz_len, char **argv);
```

```

error_t argz_insert(char **argz, size_t *argz_len, char *before,
                    const char *entry);

char *argz_next(char *argz, size_t argz_len, const char *entry);

error_t argz_replace(char **argz, size_t *argz_len, const char *str,
                     const char *with, unsigned int *replace_count);

void argz_stringify(char *argz, size_t len, int sep);

```

DESCRIPTION

These functions are glibc-specific.

An argz vector is a pointer to a character buffer together with a length. The intended interpretation of the character buffer is an array of strings, where the strings are separated by null bytes (`'\0'`). If the length is nonzero, the last byte of the buffer must be a null byte.

These functions are for handling argz vectors. The pair (NULL,0) is an argz vector, and, conversely, argz vectors of length 0 must have NULL pointer. Allocation of nonempty argz vectors is done using **malloc**(3), so that **free**(3) can be used to dispose of them again.

argz_add() adds the string str at the end of the array *argz, and updates *argz and *argz_len.

argz_add_sep() is similar, but splits the string str into substrings separated by the delimiter delim. For example, one might use this on a UNIX search path with delimiter `'.'`.

argz_append() appends the argz vector (buf, buf_len) after (*argz, *argz_len) and updates *argz and *argz_len. (Thus, *argz_len will be increased by buf_len.)

argz_count() counts the number of strings, that is, the number of null bytes (`'\0'`), in (argz, argz_len).

argz_create() converts a UNIX-style argument vector argv, terminated by (char *) 0, into an argz vector (*argz, *argz_len).

argz_create_sep() converts the null-terminated string str into an argz vector (*argz, *argz_len) by breaking it up at every occurrence of the separator sep.

argz_delete() removes the substring pointed to by entry from the argz vector (*argz, *argz_len) and updates *argz and *argz_len.

argz_extract() is the opposite of **argz_create**(). It takes the argz vector (argz, argz_len) and fills the array starting at argv with pointers to the substrings,

and a final NULL, making a UNIX-style argv vector. The array `argv` must have room for `argz_count(argv, argz_len) + 1` pointers.

argz_insert() is the opposite of **argz_delete()**. It inserts the argument `entry` at position `before` into the argz vector (`*argz, *argz_len`) and updates `*argz` and `*argz_len`. If `before` is NULL, then `entry` will be inserted at the end.

argz_next() is a function to step through the argz vector. If `entry` is NULL, the first entry is returned. Otherwise, the entry following is returned. It returns NULL if there is no following entry.

argz_replace() replaces each occurrence of `str` with `with`, reallocating argz as necessary. If `replace_count` is non-NULL, `*replace_count` will be incremented by the number of replacements.

argz_stringify() is the opposite of **argz_create_sep()**. It transforms the argz vector into a normal string by replacing all null bytes (`'\0'`) except the last by `sep`.

RETURN VALUE

All argz functions that do memory allocation have a return type of `error_t`, and return 0 for success, and **ENOMEM** if an allocation error occurs.

CONFORMING TO

These functions are a GNU extension. Handle with care.

BUGS

Argz vectors without a terminating null byte may lead to Segmentation Faults.

SEE ALSO

envz_add(3)

COLOPHON

This page is part of release 3.54 of the Linux `man-pages` project. A description of the project, and information about reporting bugs, can be found at <http://www.kernel.org/doc/man-pages/>.

2007-05-18
