NAME

figpar, **parse_config**, **get_config_option** - configuration file parsing library

LIBRARY

library "libfigpar"

SYNOPSIS

#include <figpar.h>

int
parse_config(*struct figpar_config options[], const char *path,*
*int (*unknown)(struct figpar_config *option, uint32_t line, char *directive, char *value),*
uint8_t processing_options);

*struct figpar_config **
get_config_option(*struct figpar_config options[], const char *directive*);

#include <string_m.h>

int
replaceall(*char *source*, *const char *find*, *const char *replace*);

unsigned int
strcount(*const char *source*, *const char *find*);

void
strexpand(*char *source*);

void
strexpandnl(*char *source*);

void
strtolower(*char *source*);

DESCRIPTION

The **figpar** library provides a light-weight, portable framework for parsing configuration files. The library uses open(2), read(2), and lseek(2) within the file pointed to by *path* to find directives and values which are then made available to the application.

Due to the fact that configuration files may have basic syntax differences, the library does not attempt to

impose any structure on the data but instead provides raw data to a set of callback functions. These callback functions can in-turn initiate abort through their return value, allowing custom syntax validation during parsing.

Configuration directives, types, and callback functions are provided through data structures defined in <*figpar.h*>:

```
struct figpar_config {
    enum figpar_cfgtype         type;           /* value type */
    const char                  *directive;     /* keyword */
    union figpar_cfgvalue       value;          /* value */

    /* Pointer to function used when directive is found */
    int (*action)(struct figpar_config *option, uint32_t line,
        char *directive, char *value);
};

enum figpar_cfgtype {
    FIGPAR_TYPE_NONE      = 0x0000, /* directives with no value */
    FIGPAR_TYPE_BOOL      = 0x0001, /* boolean */
    FIGPAR_TYPE_INT       = 0x0002, /* signed 32 bit integer */
    FIGPAR_TYPE_UINT      = 0x0004, /* unsigned 32 bit integer */
    FIGPAR_TYPE_STR       = 0x0008, /* string pointer */
    FIGPAR_TYPE_STRARRAY  = 0x0010, /* string array pointer */
    FIGPAR_TYPE_DATA1     = 0x0020, /* void data type-1 (open) */
    FIGPAR_TYPE_DATA2     = 0x0040, /* void data type-2 (open) */
    FIGPAR_TYPE_DATA3     = 0x0080, /* void data type-3 (open) */
    FIGPAR_TYPE_RESERVED1 = 0x0100, /* reserved data type-1 */
    FIGPAR_TYPE_RESERVED2 = 0x0200, /* reserved data type-2 */
    FIGPAR_TYPE_RESERVED3 = 0x0400, /* reserved data type-3 */
};

union figpar_cfgvalue {
    void    *data;      /* Pointer to NUL-terminated string */
    char    *str;       /* Pointer to NUL-terminated string */
    char    **strarray; /* Pointer to an array of strings */
    int32_t num;        /* Signed 32-bit integer value */
    uint32_t            u_num;      /* Unsigned 32-bit integer value */
    uint32_t            boolean:1;  /* Boolean integer value (0 or 1) */
};
```

The *processing_options* argument to **parse_config**() is a mask of bit fields which indicate various processing options. The possible flags are:

FIGPAR_BREAK_ON_EQUALS	An equals sign ('=') is normally considered part of the directive. This flag enables terminating the directive at the equals sign. Also makes equals sign optional and transient.
FIGPAR_BREAK_ON_SEMICOLON	A semicolon (';') is normally considered part of the value. This flag enables terminating the value at the semicolon. Also allows multiple statements on a single line separated by semicolon.
FIGPAR_CASE_SENSITIVE	Normally directives are matched case insensitively using fnmatch(3). This flag enables directive matching to be case sensitive.
FIGPAR_REQUIRE_EQUALS	If a directive is not followed by an equals, processing is aborted.
FIGPAR_STRICT_EQUALS	Equals must be part of the directive to be considered a delimiter between directive and value.

The *options* struct array pointer can be NULL and every directive will run the **unknown**() function argument.

The directive for each figpar_config item in the **parse_config**() options argument is matched against each parsed directive using fnmatch(3) until a match is found. If a match is found, the **action**() function for that figpar_config directive is run with the line number, directive, and value. Otherwise if no match, the **unknown**() function is run (with the same arguments).

If either *action* or *unknown* return non-zero, **parse_config**() aborts reading the file and returns the error value to its caller.

get_config_option() traverses the options-array and returns the option that matches via strcmp(3), or if no match a pointer to a static dummy struct is returned (whose values are all zero or NULL).

The use of *struct figpar_config* is entirely optional as-is the use of *enum figpar_cfgtype* or *union figpar_cfgvalue*. For example, a NULL pointer can be passed to **parse_config**() for the first argument while providing a simple *unknown* function based on queue(3) that populates a singly-linked list of a struct containing the *directive* and *value*.

In addition, miscellaneous string manipulation routines are provided by <*string_m.h*>:

replaceall() Replace all occurrences of *find* in *source* with *replace*.

strcount() Count the number of occurrences of one string that appear in the *source* string. Return value is the total count. An example use would be to show how large a block of memory needs to be for a **replaceall**() series.

strexpand() Expand escape sequences in a buffer pointed to by *source*.

strexpandnl() Expand only the escaped newlines in a buffer pointed to by *source*.

strtolower() Convert a string to lower case.

SEE ALSO

queue(3)

HISTORY

The **figpar** library first appeared in FreeBSD 10.2.

AUTHORS

Devin Teske <dteske@FreeBSD.org>

BUGS

This is the first implementation of the library, and the interface may be subject to refinement.