NAME

fieldread, fieldmake, fieldwrite, fieldfree − field access package

SYNTAX

```
#include "fields.h"

typedef struct {
    int nfields;
    int hadnl;
    char *linebuf;
    char **fields;
} field_t;

#define FLD_RUNS        0x0001
#define FLD_SNGLQUOTES  0x0002
#define FLD_BACKQUOTES  0x0004
#define FLD_DBLQUOTES   0x0008
#define FLD_SHQUOTES    0x0010
#define FLD_STRIPQUOTES 0x0020
#define FLD_BACKSLASH   0x0040

extern field_t *fieldread (FILE * file, char * delims,
                        int flags, int maxf);
extern field_t *fieldmake (char * line, int allocated,
                        char * delims, int flags, int maxf);
extern int fieldwrite (FILE * file, field_t * fieldp,
                    int delim);
extern void fieldfree (field_t * fieldp);

extern unsigned int field_line_inc;
extern unsigned int field_field_inc;
```

DESCRIPTION

The fields access package eases the common task of parsing and accessing information which is separated into fields by whitespace or other delimiters. Various options can be specified to handle many common cases, including selectable delimiters, runs of delimiters, and quoting.

fieldread reads one line from a file, parses it into fields as specified by the parameters, and returns a **field_t** structure describing the result. *fieldmake* performs the same process on a buffer already in memory. *fieldwrite* creates an output line from a **field_t** structure and writes it to an output file. *fieldfree* frees a **field_t** structure and any associated memory allocated by the package.

The **field_t** structure describes the fields in a parsed line. A well-behaved should only access the **nfields**, **fields**, and **hadnl** elements; all other elements are used internally by the package and are not guaranteed to remain the same even though they are documented here. **Nfields** gives the number of fields in the parsed line, just like the **argc** argument to a C program; **fields** is a pointer to an array of string pointers, just like the **argv** argument to a C program. As in C, the last field pointer is followed by a null pointer, although the field count is the preferred method of accessing fields. The user may alter **nfields** by decreasing it, and may replace any pointer in **fields** without harm. This is often useful in replacing a single field with a calculated value preparatory to output. The **hadnl** element is nonzero if the original line was terminated with a newline when it was parsed; this is used to accurately reproduce the input when *fieldwrite* is called.

The **linebuf** element contains a pointer to an internal buffer allocated by *fieldread* or provided to *fieldmake*. This buffer is *not* guaranteed to contain anything sensible, although in the current implementation all of the field contents can be found therein.

fieldread reads a single line of arbitrary length from **file**, allocating as much memory as necessary to hold it, and then parses the line according to its remaining arguments. A pointer to the parsed **field_t** structure is returned, with **NULL** returned if an error occurs or if **EOF** is reached on the input file. Fields in the input line are considered to be separated by any of the delimiters in the **delims** parameter. For example, if delimiters of ":.;" are specified, a line containing "a:b;c.d" would be considered to have four fields.

The default parsing of fields considers each delimiter to indicate a separate field, and does not allow any quoting. This is similar to the parsing done by *cut*(1). This behavior can be modified by specifying flags. Multiple flags may be OR'ed together. The available flags are:

FLD_RUNS
Consider runs of delimiters to be the same as a single delimiter, suppressing all null fields. This is similar to the way utilities like *awk*(1) and *sort*(1) treat whitespace, but it is not limited to whitespace. A run does not have to consist of a single type of delimiter; if both semicolon and colon are delimiters, ";::;" is a run.

FLD_SNGLQUOTES
Allow field contents to be quoted with single quotes. Delimiters and other quotes appearing within single quotes are ignored. This may appear in combination with other quote options.

FLD_BACKQUOTES
Allow field contents to be quoted with reverse single quotes. Delimiters and other quotes appearing within reverse single quotes are ignored. This may appear in combination with other quote options.

FLD_DBLQUOTES
Allow field contents to be quoted with single quotes. Delimiters and other quotes appearing within double quotes are ignored. This may appear in combination with other quote options.

FLD_SHQUOTES
Allow shell-style quoting. In the absence of this option, quotes are only recognized at the beginning of a field, and characters following the close quote are removed from the field (and are thus lost from the input line). If this option is specified, quotes may appear within a field, in the same way as they are handled by *sh*(1). Multiple quoting styles may be used in the same field. If none of **FLD_SNGLQUOTES**, **FLD_BACKQUOTES**, or **FLD_DBLQUOTES** is specified with **FLD_SHQUOTES**, all three options are implied.

FLD_STRIPQUOTES
Remove quotes and backslash sequences from the field while parsing, converting backslash sequences to their proper ASCII equivalent. The C sequences \a, \b, \f, \n, \r, \v, \x*nn*, and *nnn* are supported. Any other sequence is simply converted to the backslashed character, as in *sh*(1).

FLD_BACKSLASH
Accept standard C-style backslash sequences. The sequence will be converted to an ASCII equivalent if **FLD_STRIPQUOTES** is specified (q.v.).

FLD_NOSHRINK
Don't shrink allocated memory using *realloc*(3) before returning. This option can have a significant effect on performance, especially when *fieldfree* is going to be called soon after *fieldread* or *fieldmake*. The disadvantage is that slightly more memory will be occupied until the field structure is freed.

The *maxf* parameter, if nonzero, specifies the maximum number of fields to be generated. This may enhance performance if only the first few fields of a long line are of interest to the caller. The actual number of fields returned is one greater than *maxf*, because the remainder of the line will be returned as a single contiguous (and uninterpreted, **FLD_STRIPQUOTES** or **FLD_BACKSLASH** is specified) field.

fieldmake operates exactly like *fieldread*, except that the line parsed is provided by the caller rather than being read from a file. If the *allocated* parameter is nonzero, the memory pointed to by the *line* parameter will automatically be freed when *fieldfree* is called; otherwise this memory is the caller's responsibility. The memory pointed to by *line* is destroyed by *fieldmake*. All other parameters are the same as for *fieldread.*

fieldwrite writes a set of fields to the specified *file*, separating them with the delimiter character *delim* (note that this is a character, not a string), and appending a newline if specified by the *hadnl* element of the structure. The field structure is not freed. *fieldwrite* will return nonzero if an I/O error is detected.

fieldfree frees the **field_t** structure passed to it, along with any associated auxiliary memory allocated by the package (or passed to *fieldmake*). The structure may not be accessed after *fieldfree* is called.

field_line_inc (default 512) and **field_field_inc** (default 20) describe the increments to use when expanding lines as they are read in and parsed. *fieldread* initially allocates a buffer of **field_line_inc** bytes and, if the input line is larger than that, expands the buffer in increments of the same amount until it is large enough. If input lines are known to consistently reach a certain size, performance will be improved by setting **field_line_inc** to a value larger than that size (larger because there must be room for a null byte). **field_field_inc** serves the same purpose in both *fieldread* and *fieldmake*, except that it is related to the number of fields in the line rather than to the line length. If the number of fields is known, performance will be improved by setting **field_field_inc** to at least one more than that number.

RETURN VALUES

fieldread and *fieldmake* return **NULL** if an error occurs or if **EOF** is reached on the input file. *fieldwrite* returns nonzero if an output error occurs.

BUGS

Thanks to the vagaries of ANSI C, the **fields.h** header file defines an auxiliary macro named **P**. If the user needs a similarly-named macro, this macro must be undefined first, and the user's macro must be defined after **fields.h** is included.