Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'stdio.h.0p' command

$ man stdio.h.0p

stdio.h(0P) POSIX Programmer's Manual stdio.h(0P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

stdio.h ? standard buffered input/output

SYNOPSIS

#include <stdio.h>

DESCRIPTION

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see the System Interfaces volume of POSIX.1?2017, Section 2.2, The Compilation Environment) to enable the visibility of these symbols in this header.

The <stdio.h> header shall define the following data types through typedef:

FILE A structure containing information about a file.

fpos_t A non-array type containing all information needed to specify uniquely every position within a file.

off_t As described in <sys/types.h>.

size_t As described in <stddef.h>.

ssize_t As described in <sys/types.h>.

va_list As described in <stdarg.h>.

The <stdio.h> header shall define the following macros which shall ex?

pand to integer constant expressions:

BUFSIZ Size of <stdio.h> buffers. This shall expand to a posi?

tive value.

L_ctermid Maximum size of character array to hold ctermid() output.

L_tmpnam Maximum size of character array to hold tmpnam() output.

The <stdio.h> header shall define the following macros which shall ex?

pand to integer constant expressions with distinct values:

_IOFBF Input/output fully buffered.

_IOLBF Input/output line buffered.

_IONBF Input/output unbuffered.

The <stdio.h> header shall define the following macros which shall ex?

pand to integer constant expressions with distinct values:

SEEK_CUR Seek relative to current position.

SEEK_END Seek relative to end-of-file.

SEEK_SET Seek relative to start-of-file.

The <stdio.h> header shall define the following macros which shall ex?

pand to integer constant expressions denoting implementation limits:

{FILENAME_MAX}

Maximum size in bytes of the longest pathname that the

implementation guarantees can be opened.

{FOPEN_MAX} Number of streams which the implementation guarantees can

be open simultaneously. The value is at least eight.

{TMP_MAX} Minimum number of unique filenames generated by tmpnam().

Maximum number of times an application can call tmpnam()

reliably. The value of {TMP_MAX} is at least 25.

On XSI-conformant systems, the value of {TMP_MAX} is at

least 10000.

The <stdio.h> header shall define the following macro which shall ex?

pand to an integer constant expression with type int and a negative

value:

EOF End-of-file return value.

The <stdio.h> header shall define NULL as described in <stddef.h>.

The <stdio.h> header shall define the following macro which shall ex?

pand to a string constant:

P_tmpdir Default directory prefix for tempnam().

The <stdio.h> header shall define the following macros which shall ex?

pand to expressions of type ``pointer to FILE'' that point to the FILE

objects associated, respectively, with the standard error, input, and

output streams:

stderr Standard error output stream.

stdin Standard input stream.

stdout Standard output stream.

The following shall be declared as functions and may also be defined as

macros. Function prototypes shall be provided.

```
void     clearerr(FILE *);
char    *ctermid(char *);
int      dprintf(int, const char *restrict, ...)
int      fclose(FILE *);
FILE    *fdopen(int, const char *);
int      feof(FILE *);
int      ferror(FILE *);
int      fflush(FILE *);
int      fgetc(FILE *);
int      fgetpos(FILE *restrict, fpos_t *restrict);
char    *fgets(char *restrict, int, FILE *restrict);
int      fileno(FILE *);
void     flockfile(FILE *);
FILE    *fmemopen(void *restrict, size_t, const char *restrict);
FILE    *fopen(const char *restrict, const char *restrict);
int      fprintf(FILE *restrict, const char *restrict, ...);
int      fputc(int, FILE *);
int      fputs(const char *restrict, FILE *restrict);
size_t   fread(void *restrict, size_t, size_t, FILE *restrict);
```

```
FILE    *freopen(const char *restrict, const char *restrict,
            FILE *restrict);
int     fscanf(FILE *restrict, const char *restrict, ...);
int     fseek(FILE *, long, int);
int     fseeko(FILE *, off_t, int);
int     fsetpos(FILE *, const fpos_t *);
long    ftell(FILE *);
off_t   ftello(FILE *);
int     ftrylockfile(FILE *);
void    funlockfile(FILE *);
size_t  fwrite(const void *restrict, size_t, size_t, FILE *restrict);
int     getc(FILE *);
int     getchar(void);
int     getc_unlocked(FILE *);
int     getchar_unlocked(void);
ssize_t getdelim(char **restrict, size_t *restrict, int,
            FILE *restrict);
ssize_t getline(char **restrict, size_t *restrict, FILE *restrict);
char    *gets(char *);
FILE    *open_memstream(char **, size_t *);
int     pclose(FILE *);
void    perror(const char *);
FILE    *popen(const char *, const char *);
int     printf(const char *restrict, ...);
int     putc(int, FILE *);
int     putchar(int);
int     putc_unlocked(int, FILE *);
int     putchar_unlocked(int);
int     puts(const char *);
int     remove(const char *);
int     rename(const char *, const char *);
int     renameat(int, const char *, int, const char *);
void    rewind(FILE *);
```

```
int     scanf(const char *restrict, ...);
void    setbuf(FILE *restrict, char *restrict);
int     setvbuf(FILE *restrict, char *restrict, int, size_t);
int     snprintf(char *restrict, size_t, const char *restrict, ...);
int     sprintf(char *restrict, const char *restrict, ...);
int     sscanf(const char *restrict, const char *restrict, ...);
char    *tempnam(const char *, const char *);
FILE    *tmpfile(void);
char    *tmpnam(char *);
int     ungetc(int, FILE *);
int     vdprintf(int, const char *restrict, va_list);
int     vfprintf(FILE *restrict, const char *restrict, va_list);
int     vfscanf(FILE *restrict, const char *restrict, va_list);
int     vprintf(const char *restrict, va_list);
int     vscanf(const char *restrict, va_list);
int     vsnprintf(char *restrict, size_t, const char *restrict,
            va_list);
int     vsprintf(char *restrict, const char *restrict, va_list);
int     vsscanf(const char *restrict, const char *restrict, va_list);
```

Inclusion of the <stdio.h> header may also make visible all symbols from <stddef.h>.

The following sections are informative.

APPLICATION USAGE

Since standard I/O streams may use an underlying file descriptor to ac? cess the file associated with a stream, application developers need to be aware that {FOPEN_MAX} streams may not be available if file descrip? tors are being used to access files that are not associated with streams.

RATIONALE

There is a conflict between the ISO C standard and the POSIX definition of the {TMP_MAX} macro that is addressed by ISO/IEC 9899:1999 standard, Defect Report 336. The POSIX standard is in alignment with the public record of the response to the Defect Report. This change has not yet

been published as part of the ISO C standard.

FUTURE DIRECTIONS

None.

SEE ALSO

<stdarg.h>, <stddef.h>, <sys_types.h>

The System Interfaces volume of POSIX.1?2017, Section 2.2, The Compila?
tion Environment, clearerr(), ctermid(), fclose(), fdopen(), feof(),
ferror(), fflush(), fgetc(), fgetpos(), fgets(), fileno(), flockfile(),
fmemopen(), fopen(), fprintf(), fputc(), fputs(), fread(), freopen(),
fscanf(), fseek(), fsetpos(), ftell(), fwrite(), getc(), getchar(),
getc_unlocked(), getdelim(), getopt(), gets(), open_memstream(),
pclose(), perror(), popen(), putc(), putchar(), puts(), remove(), re?
name(), rewind(), setbuf(), setvbuf(), stdin, system(), tempnam(), tmp?
file(), tmpnam(), ungetc(), vfprintf(), vfscanf()

COPYRIGHT