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 IEEE Std 1003.1-2001, Section 2.2, The Compilation Environment) to enable the visibility of these symbols in this header.

The <*stdio.h*> header shall define the following macros as positive integer constant expressions:

BUFSIZ
: Size of <*stdio.h*> buffers.

_IOFBF
: Input/output fully buffered.

_IOLBF
: Input/output line buffered.

_IONBF
: Input/output unbuffered.

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

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

SEEK_CUR
: Seek relative to current position.

SEEK_END
: Seek relative to end-of-file.

SEEK_SET
: Seek relative to start-of-file.

The following macros shall be defined as positive integer constant expressions which denote implementation limits:

{FILENAME_MAX}
: Maximum size in bytes of the longest filename string 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 following macro name shall be defined as a negative integer constant expression:

EOF End-of-file return value.

The following macro name shall be defined as a null pointer constant:

NULL Null pointer.

The following macro name shall be defined as a string constant:

P_tmpdir
Default directory prefix for *tempnam*().

The following shall be defined as 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 data types shall be defined 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.

va_list As described in <*stdarg.h*> .

size_t As described in <*stddef.h*> .

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     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   *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);

char    *gets(char *);

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 *);
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, int ...);

char    *tempnam(const char *, const char *);

FILE    *tmpfile(void);
char    *tmpnam(char *);
int     ungetc(int, FILE *);
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 arg);
```

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

The following sections are informative.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<*stdarg.h*>, <*stddef.h*>, <*sys/types.h*>, the System Interfaces volume of IEEE Std 1003.1-2001, *clearerr*(), *ctermid*(), *fclose*(), *fdopen*(), *fgetc*(), *fgetpos*(), *ferror*(), *feof*(), *fflush*(), *fgets*(), *fileno*(), *flockfile*(), *fopen*(), *fputc*(), *fputs*(), *fread*(), *freopen*(), *fseek*(), *fsetpos*(), *ftell*(), *fwrite*(), *getc*(), *getc_unlocked*(), *getwchar*(), *getchar*(), *getopt*(), *gets*(), *pclose*(), *perror*(), *popen*(), *printf*(), *putc*(), *putchar*(), *puts*(), *putwchar*(), *remove*(), *rename*(), *rewind*(), *scanf*(), *setbuf*(), *setvbuf*(), *sscanf*(), *stdin*, *system*(), *tempnam*(), *tmpfile*(), *tmpnam*(), *ungetc*(), *vfscanf*(), *vscanf*(), *vprintf*(), *vsscanf*()

COPYRIGHT