stdin, stdout, stderr

Defined in header <stdio.h>

```
#define stdin  /* implementation-defined */    (1)
#define stdout /* implementation-defined */    (2)
#define stderr /* implementation-defined */    (3)
```

Three text streams are predefined. These streams are implicitly opened and unoriented at program startup.

1) Associated with the *standard input* stream, used for reading conventional input. At program startup, the stream is fully buffered if and only if the stream can be determined to not refer to an interactive device.
2) Associated with the *standard output* stream, used for writing conventional output. At program startup, the stream is fully buffered if and only if the stream can be determined to not refer to an interactive device.
3) Associated with the *standard error* stream, used for writing diagnostic output. At program startup, the stream is not fully buffered.

What constitutes an interactive device is implementation-defined.

These macros are expanded to expressions of type `FILE*`.

Notes

Although not mandated by POSIX, the UNIX convention is that `stdin` and `stdout` are line-buffered if associated with a terminal and `stderr` is unbuffered.

These macros may be expanded to modifiable lvalues. If any of these `FILE*` lvalue is modified, subsequent operations on the corresponding stream result in unspecified or undefined behavior.

Example

This example shows a function equivalent to `printf`.

Run this code

```
#include <stdarg.h>
#include <stdio.h>

int my_printf(const char * restrict fmt, ...)
{
    va_list vl;
    va_start(vl, fmt);
    int ret = vfprintf(stdout, fmt, vl);
    va_end(vl);
    return ret;
}

int main(void)
{
    my_printf("Rounding:\t%f %.0f %.32f\n", 1.5, 1.5, 1.3);
    my_printf("Padding:\t%05.2f %.2f %5.2f\n", 1.5, 1.5, 1.5);
    my_printf("Scientific:\t%E %e\n", 1.5, 1.5);
    my_printf("Hexadecimal:\t%a %A\n", 1.5, 1.5);
}
```

Possible output:

```
Rounding:       1.500000 2 1.30000000000000004440892098500626
Padding:        01.50 1.50  1.50
Scientific:     1.500000E+00 1.500000e+00
Hexadecimal:    0x1.8p+0 0X1.8P+0
```

References

- C17 standard (ISO/IEC 9899:2018):
    - 7.21.1 Introduction (p: 217-218)

- 7.21.2 Streams (p: 217–219)
- 7.21.2 Files (p: 219–221)

- C11 standard (ISO/IEC 9899:2011):
  - 7.21.1 Introduction (p: 296–298)
  - 7.21.2 Streams (p: 298–299)
  - 7.21.2 Files (p: 300–302)
- C99 standard (ISO/IEC 9899:1999):
  - 7.19.1 Introduction (p: 262–264)
  - 7.19.2 Streams (p: 264–265)
  - 7.19.2 Files (p: 266–268)
- C89/C90 standard (ISO/IEC 9899:1990):
  - 4.9.1 Introduction
  - 4.9.2 Streams
  - 4.9.3 Files

See also

FILE	object type, capable of holding all information needed to control a C I/O stream (typedef)
C++ documentation for **stdin**, **stdout**, **stderr**	