ft_printf

Generated by Doxygen 1.8.16

1 ft_printf

This project is part of the official curriculum at `School 42`.

1.1 Overview

- `Official instructions`
- The goal of this project is to practice the following concepts
  - variadic functions
  - dispatch tables – loose coupling and performance
  - void pointers – generic programming
  - optimization patterns – buffering
- The project (with the exception of tests) is consistent with the `Norme`, the code standard accepted at *School 42*. In particular, this means that
  - no comments inline or inside functions
  - `for` loops and `switch` statements are forbidden
  - each function must be maximum 25 lines
- As per instructions, the project is realised using `libft` compiled during a previous `project`.

1.2 Checklist

Standard

- [x] `csp%` conversions
- [x] `diouxX` conversions with `hh`, `h`, `l`, `ll` flags
- [x] `f` conversion with flags `L`, `l`
- [x] `#0-+` flag management (when applicable)
- [x] minimum field-width
- [x] precision
- [x] `*` flag management

Extra

- [x] `b` conversion to print in binary
- [x] variants of `printf`
  - `ft_dprintf`
  - `ft_vprintf`
  - `ft_vdprintf`
- [x] colors
  - `ft_printf("{red} Color-print.{eoc} Normal print.")`
  - red, blue, yellow, green, cyan, magenta

1.3 Tests

- Behaviour is tested with `Unity`
  - `make test-behavior`
- Performance of `ft_printf` can be benchmarked against that of `printf`
  - `make test-speed`

1.4 Acknowledgements

Many of the included tests are borrowed from `pft` by `gavinfielder` and other contributors and `Moulitest` by `yyang42` and other contributors. My thanks go to them.
If you have any questions, please contact me on Github.

2 Module Index

2.1 Modules

Here is a list of all modules:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Module Documentation

4.1 ft_printf() family

Replicate behaviour of `printf(3)` functions.

Functions

- int ft_printf (const char *format,...)

  Replicates behaviour of `printf(3)`.
- int ft_dprintf (int fd, const char *format,...)

  Replicates behaviour of `dprintf(3)`.
- int ft_vprintf (const char *format, va_list ap)

  Replicates behaviour of `vprintf(3)`.
- int ft_vdprintf (int fd, const char *format, va_list ap)

  Replicates behaviour of `vdprintf(3)`.

4.1.1 Detailed Description

Replicate behaviour of `printf(3)` functions.
Support standard field values and combination thereof (where applicable):

- flags: `#`, `0`, ` `, `-`, `+`
- width and precision, including `*` options
- length: `hh`, `h`, `l`, `ll`, `L`
- type: `c`, `s`, `p`, `%`, `d`, `i`, `o`, `u`, `x`, `X`, `f`

Custom specifications:

- `b` type to print in binary format (supports same parameters as `i` type)
- color support
  - `cyan`
  - `red`
  - `green`
  - `yellow`
  - `blue`

- magenta
- ft_printf("{red}Color-print. {eoc}Normal print.");
  gives
  Color-print. Normal print.

See also

https://en.wikipedia.org/wiki/Printf_format_string

https://linux.die.net/man/3/printf

4.1.2 Function Documentation

4.1.2.1 ft_dprintf()

```
int ft_dprintf (
        int fd,
        const char * format,
        ... )
```

Replicates behaviour of dprintf(3).

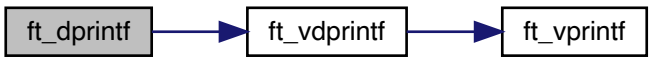
Parameters

in	*fd*	File descriptor where to print output.
in	*format*	Format string that specifies how subsequent arguments are converted for output
in	...	Variadic arguments

Returns

Number of characters printed or −1 if an error occurs. Additionaly, in case of an error, errno is set to ENOMEM (memory allocation error), EINVAL (invalid format placeholder specification), ENOTSUP (type field value not supported).

References ft_vdprintf().
Here is the call graph for this function:



4.1.2.2 ft_printf()

```
int ft_printf (
        const char * format,
        ... )
```

Replicates behaviour of printf(3).

Parameters

in	*format*	Format string that specifies how subsequent arguments are converted for output
in	...	Variadic arguments

Returns

Number of characters printed or `-1` if an error occurs. Additionaly, in case of an error, `errno` is set to `ENOMEM` (memory allocation error), `EINVAL` (invalid format placeholder specification), `ENOTSUP` (type field value not supported).

References ft_vprintf().
Here is the call graph for this function:

4.1.2.3 ft_vdprintf()

```
int ft_vdprintf (
        int fd,
        const char * format,
        va_list ap )
```

Replicates behaviour of `vdprintf(3)`.

Parameters

`in`	*fd*	File descriptor where to print output
`in`	*format*	Format string that specifies how subsequent arguments are converted for output
`in`	*ap*	A variable used by `stdarg(3)` to step through a list of variadic arguments.

Returns

Number of characters printed or `-1` if an error occurs. Additionaly, in case of an error, `errno` is set to `ENOMEM` (memory allocation error), `EINVAL` (invalid format placeholder specification), `ENOTSUP` (type field value not supported).

References ft_vprintf().
Here is the call graph for this function:

4.1.2.4 ft_vprintf()

```
int ft_vprintf (
        const char * format,
        va_list ap )
```

Replicates behaviour of `vprintf(3)`.

Parameters

in	*format*	Format string that specifies how subsequent arguments are converted for output
in	*ap*	A variable used by `stdarg(3)` to step through a list of variadic arguments.

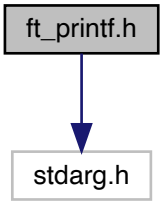
Returns

Number of characters printed or `-1` if an error occurs. Additionaly, in case of an error, `errno` is set to `ENOMEM` (memory allocation error), `EINVAL` (invalid format placeholder specification), `ENOTSUP` (type field value not supported).

5 File Documentation

5.1 ft_printf.h File Reference

`#include <stdarg.h>`
Include dependency graph for ft_printf.h:



Functions

- int ft_printf (const char *format,...)

  Replicates behaviour of `printf(3)`.

- int ft_dprintf (int fd, const char *format,...)

  Replicates behaviour of `dprintf(3)`.

- int ft_vprintf (const char *format, va_list ap)

  Replicates behaviour of `vprintf(3)`.

- int ft_vdprintf (int fd, const char *format, va_list ap)

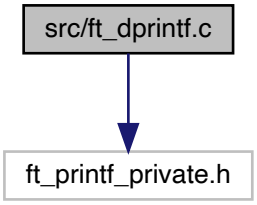
  Replicates behaviour of `vdprintf(3)`.

5.2 README.md File Reference

5.3 src/ft_dprintf.c File Reference

`#include "ft_printf_private.h"`

Include dependency graph for ft_dprintf.c:



Functions

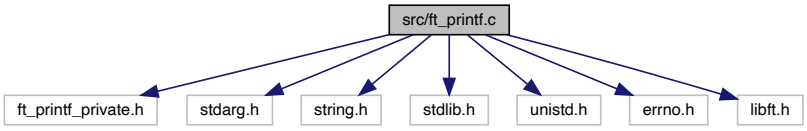
- int ft_dprintf (int fd, const char ∗format,...)

  Replicates behaviour of `dprintf(3)`.

5.4 src/ft_printf.c File Reference

```
#include "ft_printf_private.h"
```

Include dependency graph for ft_printf.c:



Functions

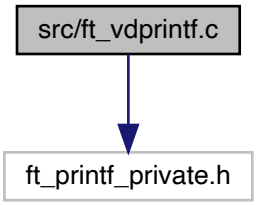
- int ft_printf (const char ∗format,...)

  Replicates behaviour of `printf(3)`.

5.5 src/ft_vdprintf.c File Reference

```
#include "ft_printf_private.h"
```

Include dependency graph for ft_vdprintf.c:



Functions

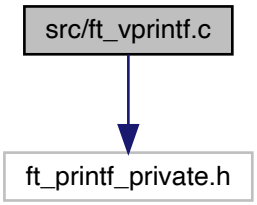
- int ft_vdprintf (int fd, const char *format, va_list ap)

  Replicates behaviour of `vdprintf(3)`.

5.6 src/ft_vprintf.c File Reference

```
#include "ft_printf_private.h"
```

Include dependency graph for ft_vprintf.c:



Functions

- int ft_vprintf (const char *format, va_list ap)

  Replicates behaviour of `vprintf(3)`.

Index