

# C Coding Guidelines

## Variables

Local: *{name\_of\_var}*

Global: *g\_{detailed\_name\_of\_var}*

```
int i;
extern int g_alien_cnt;
```

## Functions

### Names

*{libLabel}\_{object}\_{action\_or\_verb}* or *{libLabel}\_{action\_or\_verb}\_{object}*

```
sfSyl_welcome_txt_print ();
sfSyl_print_welcome_txt ();
```

Rationale: first form is easier for completion.

### Definitions

```
[attributes ]{type}
{func_name}( {args} )
{
    /*...*/
}
```

```
noreturn void
usage( int status )
{
    /*...*/
    exit( status );
}
```

Rationale: easier to `grep` ("^func\_name").

## Pointers declaration

```
{type} *{var}
int *var1, *var2;    /* 2 pointers */
```

Rationale: avoid confusion like in `int *var1, var2; /* 1 pointer, 1 int ! */`.

## Typedefs

```
[libLabel_] {name}_t
superint_t i;
sfs_superint_t j;
```

## Structures

```
{name_of_struct}_s  
{name_of_struct_var}  
datstruct_s this_is_a_struct;
```

## Enums

```
{name_of_enum}_e  
{ENUM_CONST}  
{name_of_enum_var}  
enum mood_e { TAKE_IT, GIVE_IT, KEEP_IT } my_mood;
```

## Gotos

*[GT\_]{ThisPart}* or *[GT\_]{this\_part}*

```
EmergencyClosure:  
GT_EmergencyClosure:  
GT_emergency_closure:
```

*Goto* statements must always be on the very first level of indentation: they must be immediately noticeable.

## Define

```
[TYPE_]{NAME_OF_DEF}  
#define ALIENS_ON_PLANET_CNT 1234
```

## Macros

```
[M_]{OBJECT}_{VERB} or [M_]{VERB}_{OBJECT}  
#define ALIENS_ON_PLANET_LOCATE ()  
#define M_ALIENS_ON_PLANET_LOCATE ()  
#define LOCATE_ALIENS_ON_PLANET ()
```

## Header guards

```
{NAME_OF_HEADER}_H  
#ifndef MY_COOL_LIB_H  
#define MY_COOL_LIB_H  
/*...*/  
#endif
```

Rationale: `_` and `__`-starting header guards are used by standard library headers.

## Parentheses / braces

We have different policies for functions and statements for the sake of distinction.

## Functions calls

```
{func_name}( {args} );  
printf( "spaces btwn args and parenthesis : %d", true_dat );
```

## Statements

```
{statement} ({condition}) {  
    /*...*/  
}  
  
if (true_dat == 1) {  
    /*...*/  
} else {  
    /*...*/  
}
```

## Code example

```
#ifndef THAT_GUARD_THOUGH_H  
#define THAT_GUARD_THOUGH_H  
  
#include "myheader.h"  
  
#include <header1.h>  
#include <header2.h>  
  
#define STR_SIZE_OF_PLANET  "BIG"  
  
noreturn void  
f_datFunc( void )  
{  
    unsigned int aliens_cnt = 100;  
    int happn = 0;  
  
    printf( "This planet is %s.\n", STR_SIZE_OF_PLANET );  
    if (aliens_cnt > 50) {  
        puts( "it's happening" );  
        happn = 1;  
    } else if (aliens_cnt > 0) {  
        puts( "we still have time" );  
        happn = 0;  
    } else { puts( "ERROR" ); goto GT_Habbenning; }  
  
    switch (happn) {  
        case 0:  
            return( EXIT_SUCCESS );  
        default:  
            GT_Habbenning:  
                return( EXIT_FAILURE );  
    }  
}
```

```
#endif      /* undef THAT_GUARD_THOUGH_H */
```

## General advices

- *snake\_case*: easier to type, harder to read
  - Though: some of the best ever written softwares were made in *snake\_case*
- *camelCase*: harder to type, easier to read
  - Microsoft uses it, so...
- Dividing the code in functions increase its comprehension and readability.
- Code must not be generic, but very specific to what exactly you're doing.
- Code for debug purpose must be removed from the final form of the code.
- Always use header guards in header files.
- Put braces even on one-line statements.

## Formatting your code using *sindent*

*sindent*, my own taste of GNU *indent*, format your code according to the Linux kernel coding style (`-linux`) plus a few option it's missing (`-ps1 -prs -npcs`), which allows for easier **grep**-ing of function definitions and function calls.

## References/resources

- *Linux Kernel Coding style*: <https://www.kernel.org/doc/html/v4.10/process/coding-style.html>
  - *Notes on Programming in C*, Rob Pike: <https://www.lysator.liu.se/c/pikestyle.html>
  - *C Header File Guidelines*, David Kieras, University of Michigan: <http://umich.edu/~eecs381/handouts/CHeaderFileGuidelines.pdf>
  - *JPL Coding Standard C*, Jet Propulsion Laboratory, NASA: [https://lars-lab.jpl.nasa.gov/JPL\\_Coding\\_Standard\\_C.pdf](https://lars-lab.jpl.nasa.gov/JPL_Coding_Standard_C.pdf)
- 

## Project Hierarchy Standard

### Tree

```
[PROJECT DIRECTORY]/
|-- bin
|   |-- data -> ../data
|   |-- project
|   |-- project.exe
|   +-- project.log
|-- data
|   |-- images
|   +-- ...
|-- etc
|   +-- project.conf
|-- lib32
|   |-- libcsfml-audio.dll
|   +-- ...
|-- lib64
|   |-- libcsfml-audio.so.1.6
```

```

|   +-- ...
|-- man
|   +-- project.6
|-- readme.d
|   |-- AUTHORS.txt
|   |-- LICENSE.SFML.txt
|   |-- LICENSE.txt
|   |-- changelog
|   +-- copyright
|-- src
|   |-- font
|   |   +-- usedGPLFont.zip
|   |-- inc
|   |   +-- SFML
|   |       |-- Audio
|   |           |-- AudioResource.hpp
|   |           |-- Types.h
|   |           +-- ...
|   |       |-- Graphics
|   |           +-- ...
|   |       +-- ...
|   |-- Makefile
|   |-- project.c
|   |-- project.h
|   |-- utils.c
|   +-- ...
|-- wip
|   |-- DevLog
|   |   |-- Screenshot - 12142013 - 02:44:22 PM.png
|   |   +-- ...
|   |-- datMusicParts
|   |   +-- ...
|   +-- ...
|-- NOTES
|-- README
+-- TODO

```

## Directories

| [Name]     | [Content]  |
|------------|--|
| ./         | Regular README files and possibly other (few) things                             |
| ./bin      | Binaries ; where the program is built  |
| ./data     | Project data (images, sounds, fonts, etc... )                                    |
| ./etc      | Configuration files  |
| ./lib32    | 32-bit libraries (*.lib, *.so, *.a, *.dll)                                       |
| ./lib64    | 64-bit libraries (*.lib, *.so, *.a, *.dll)                                       |
| ./man      | Linux manual pages   |
| ./readme.d | Remaining licensing information and other informative text files (not mandatory) |
| ./src      | Source files   |
| ./src/inc  | Included external headers  |
| ./wip      | <b>“Work In Progress”</b> material   |

## Releasing

When releasing the project to a wider audience, it's necessary to remove useless files and directory such as:

- `./wip`