Documentation: Config Manager

The config manager is a program which can:

- load **configuration files**.
- create an **environment**.

The environment works in the same way that a shell environment.

It allows to **stock**, **removal, modify** or **access** to any variable in the whole program.

Loading configuration file:

```
env_t my_env = config_manager_create();

int ret = config_manager_read(&my_env, "./asset/conf/file.conf");
if (ret == EXIT_ERROR)
        return EXIT_ERROR;
config_manager_destroy(&my_env);
```

Prototype:

```
int config_manager_read(env_t *env, const char *filepath);
```

see more in config_manager.h

Reading configuration variables:

Content of the configuration file:

```
# in file.conf
WIDTH = 10
HEIGHT = 20
Double = 42.42
String = “Hello world”
```

Read variables content:

```
char *str = my_env_get_value(&my_env, "String"); // Return a copy (must be free)
int number = my_env_get_value_int(&my_env, "WIDTH", NULL);
double decimal = my_env_get_value_dec(&my_env, "Double", NULL);


//   For str variables
char *my_env_get_value(env_t *env, const char *label);
//   For Int varaibles
int my_env_get_value_int(env_t *env, const char *label, bool *error);
//   For double variables
double my_env_get_value_dec(env_t *env, const char *label, bool *error);
```

*(Information) : my_env_get_value_int and my_env_get_value_dec can get a pointer to a boolean at third parameter. Il the pointer isn’t null, this value will be updated with **True** (if an error occur) or **False** (if it’s ok)*

see more in my_env.h (in the libmy)

Other functions of the environment lib:

```
// Update value of a variable
int my_env_update(env_t *self, const char *label, const char *value);
// Remove a variable
int my_env_rm(env_t *self, const char *label);
// Insert a variable
int my_env_add(env_t *self, const char *label, const char *value);
```

```
int my_env_size(env_t *env);
void my_env_display(env_t *env);
bool my_env_exist(env_t *env, const char *label); // Check if a variable exist
```

Rules of the configuration files:

- The file must have the extension .conf
- Reserved characters => `' ' (space), '\t' (tab), '=' (equal), " (double quote)`
- Exception: if you surround a label width **double quotes** (“) you can use **space** and **tabulation** character in it. *(see my_str in the example)*
- *A variable label must be **unique**.*

Example of a (strange) error-free configuration file:

```
# A line commented
myvar  = value
my-var2=value
my_var3 =value # New comment

# The previous line is empty
       my_var4      =      value  # Tabulation are used

my_var5 = 01254*/-_-/;:...
my_str = “A value with spaces      and tabulation”
```

Comments are made with the # character.

/! info !/ If an error occur, an error message will be displayed:

```
# If the file doesn’t exist or can’t be readed:
config_manager : cannot open/read file "./asset/conf/not_exist.conf"

# If there is an error in the config file:
config_manager : syntax error in "./asset/config/window.conf" on line 5.

# If a variable already exist:
config_manager : label name conflict in "./asset/config/window.conf" with the
label: "A_String".
```