TP n° 5 – éléments de réponse

```h
#ifndef PARAMETERS_H
#define PARAMETERS_H

#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

#define DEBUG_  1
#define ON              1
#define OFF             0
#define PARAM_PATH      "/saperlipopette"
#define MUTEX_NAME      "/saperlipopette-sem"
#define FOREVER         for (;;)
#define MAX_LEN         256

/**
 * The structure that holds parameters for the syrup fountain.
 * It is stored in the associated shared memory.
 */
typedef struct parameters_t {
    pid_t pid_gui;
    int ending_flag;
    char key[MAX_LEN];   /*!< The key to activate the simulation */
    char tap[MAX_LEN];   /*!< The open/close tap */
    char drink[MAX_LEN]; /*!< The drink (water, etc.) */
} parameters;

void initialize_parameters(parameters *params);
void analyze_parameters_to_read_key(parameters *params);
void close_and_destroy_parameters(parameters *params, const char *path);

#endif
```

```c
void close_parameters(parameters *param)
{
    munmap(param, sizeof(parameters));
}

parameters *open_parameters(const char *path)
{
    int fd;             /* file descriptor associated to the parameters shm */
```

```
    parameters *params; /* pointer to the "parameters" structure */
    size_t size;        /* mailbox size */

    fd = shm_open(path, O_RDWR, S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);
    if (fd == -1) {
        log_error("shm_open() problem!");
        return NULL;
    }

    size = sizeof(parameters);
    params = (parameters *) mmap(NULL, size, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
    (off_t) 0);
    if (params == MAP_FAILED) {
        log_error("mmap() problem!");
        return NULL;
    }

    return params;
}

void modify_parameters(parameters *params)
{
    semaphore *mutex;

    log_info("Monitor> Trying to access Dispenser (PID: %d)...", params->pid_gui);
    mutex = open_semaphore(MUTEX_NAME, 0);
    if (mutex == NULL) {
        log_error("Monitor> open_semaphore() problem!");
        exit(EXIT_FAILURE);
    }
    log_info("Monitor> Mutex is going to be preempted.");
    log_info("Monitor> Trying to setup bad key...");
    P(mutex);
    strcpy(params->key, "Manon");
    strcpy(params->tap, "closed");
    strcpy(params->drink, "water");
    V(mutex);
    log_info("Monitor> Mutex has been released.");
    sleep(10);

    log_info("Monitor> Mutex is going to be preempted.");
    log_info("Monitor> Trying to setup good key...");
    P(mutex);
    /* Test "Manon des sources" */
    strcpy(params->key, "Znaba qrf fbheprf");
    strcpy(params->tap, "closed");
    strcpy(params->drink, "water");
```

```
    V(mutex);
    log_info("Monitor> Mutex has been released.");
    sleep(10);

    log_info("Monitor> Mutex is going to be preempted.");
    log_info("Monitor> Trying to open tap...");
    P(mutex);
    strcpy(params->tap, "opened");
    strcpy(params->drink, "water");
    V(mutex);
    log_info("Monitor> Mutex has been released.");
    sleep(10);

    log_info("Monitor> Mutex is going to be preempted.");
    log_info("Monitor> Trying to choose \"mint\" syrup...");
    P(mutex);
    strcpy(params->drink, "mint");
    V(mutex);
    log_info("Monitor> Mutex has been released.");
    sleep(10);

    log_info("Monitor> Mutex is going to be preempted.");
    log_info("Monitor> Trying to choose \"grenadine\" syrup...");
    P(mutex);
    strcpy(params->drink, "grenadine");
    V(mutex);
    log_info("Monitor> Mutex has been released.");
    sleep(40);

    log_info("It's time to exit....");
    log_info("Monitor> Mutex is going to be preempted.");
    log_info("Monitor> Trying to close Dispenser (PID: %d)...", params->pid_gui);
    P(mutex);
    strcpy(params->tap, "closed");
    sleep(1);
    params->ending_flag = ON;
    V(mutex);
    sleep(1);
    log_info("Bye...");
}
```

```
int main(int argc, char **argv)
{
    parameters *params;

    log_info("Monitor> ON");
```

```
    params = open_parameters(PARAM_PATH);

    if (params != NULL) {
        modify_parameters(params);
        close_parameters(params);
    }

    log_info("Monitor> OFF");

    exit(EXIT_SUCCESS);
}
```