```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<syr1_file.h>

/*
 * SYNOPSIS :
 *    int syr1_fopen_read(char* name, SYR1_FILE* file) {
 * DESCRIPTION :
 *    Ce sous-programme gère l'ouverture d'un fichier logique en mode lecture.
 * PARAMETRES :
 *    name : chaîne de caratère contenant le nom externe du fichier à ouvrir
 *    file : pointeur sur un Bloc Control Fichier (File Control Bloc)
 * RESULTAT :
 *    0 : ouverture réussie
 *   -1 : autre erreur
 */
int syr1_fopen_read(char* name, SYR1_FILE* file) {
    // Recherche du fichier
    file_descriptor* desc = malloc(sizeof(file_descriptor));
    if (desc == NULL || search_entry(name, desc) < 0)
        return -1;
    // Si trouvé, allocation du BCF
    if (file == NULL)
        return -1;
    file->descriptor = *desc;
    unsigned char* buf = malloc(IO_BLOCK_SIZE);
    file->buffer = buf;
    strcpy(file->mode, "r");
    file->current_block = 0;
    file->file_offset = 0;
    file->block_offset = 0;
    // Préchargement du premier bloc de données
    if (read_block(desc->alloc[0], file->buffer) < 0)
        return -1;
    return 0;
}

/*
 * SYNOPSIS :
 *    int syr1_fread(SYR1_FILE* file, int item_size, int nbitem, char* buf)
 * DESCRIPTION :
 *    Ce sous-programme lit nbitem articles de taille item_size dans le fichier
 *    logique passé en paramètre.
 * PARAMETRES :
 *    file      : pointeur sur un Bloc Control Fichier (File Control Bloc)
 *    item_size : taille d'un article
 *    nb_item   : nombre d'article à lire
 * RESULTAT :
 *    le nombre d'articles effectivement lus dans le fichier, sinon un code
 *    d'erreur (cf syr1_getc())
 *   -1 : le BCF est NULL, ou le mode d'ouverture est incorrect
 *   -2 : erreur d'entrée-sorties sur le périphérique de stockage
 *   -3 : fin de fichier
 */
int syr1_fread(SYR1_FILE* file, int item_size, int nbitem, char* buf) {
    int count = 0;
    while (count < nbitem*item_size) {
        int res = syr1_getc(file);
        if (res<0)
            return res;
        else
```



```
            buf[count] = (unsigned char) res;
        count++;
    }
    return count/item_size;
}

/*
 * SYNOPSIS :
 *    int syr1_getc(SYR1_FILE* file)
 * DESCRIPTION :
 *    Ce sous-programme lit un caractère à partir du fichier passé en paramètre.
 * PARAMETRES :
 *    file : pointeur sur un descripteur de fichier logique (File Control Bloc)
 * RESULTAT :
 *    valeur (convertie en int) du caractère lu dans le fichier, sinon
 *   -1 : le BCF est NULL, ou le mode d'ouverture est incorrect
 *   -2 : erreur d'entrée-sortie sur le périphérique de stockage
 *   -3 : fin de fichier
 */
int syr1_getc(SYR1_FILE* file) {
    if (file == NULL || file->mode[0] != 'r')
        return -1;
    if (file->file_offset >= file->descriptor.size)
        return -3;
    // Si on est en fin de bloc de données, on charge le suivant
    if (file->block_offset >= IO_BLOCK_SIZE) {
        file->current_block++;
        if (read_block(file->descriptor.alloc[file->current_block], \
                file->buffer) < 0)
            return -2;
        file->block_offset = 0;
    }
    // On récupère le caractère voulu et on avance les compteurs
    int ch = file->buffer[file->block_offset];
    if (ch == EOF)
        return -3;
    file->block_offset++;
    file->file_offset++;
    return ch;
}

/*
 * SYNOPSIS :
 *    int syr1_fclose_read(SYR1_FILE* file) {
 * DESCRIPTION :
 *    Ce sous-programme gère la fermeture d'un fichier logique.
 * PARAMETRES :
 *    file : pointeur sur un Bloc de Contrôle Fichier (BCF)
 * RESULTAT :
 *    0 : la fermeture a réussi
 *   -1 : problème pendant la libération du descripteur de fichier logique
 *        (ou file vaut NULL)
 */
int syr1_fclose_read(SYR1_FILE* file) {
    if (file == NULL || free_logical_file(file) < 0)
        return -1;
    return 0;
}
```

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<syr1_file.h>

/*
 * SYNOPSIS :
 *    int syr1_fopen_write(char *name, SYR1_FILE *file) {
 * DESCRIPTION :
 *    Ce sous-programme gère l'ouverture d'un fichier logique en mode écriture.
 * PARAMETRES :
 *    name : chaîne de caratère contenant le nom externe du fichier à ouvrir
 *    file : pointeur sur un Bloc Control Fichier (File Control Bloc)
 * RESULTAT :
 *    0 : ouverture réussie
 *   -1 : autre erreur
 */
int syr1_fopen_write(char* name, SYR1_FILE* file) {
    // Recherche du fichier
    file_descriptor* desc = malloc(sizeof(file_descriptor));
    if (desc == NULL)
        return -1;
    int err = search_entry(name, desc);
    if (err == -2)
        return -1;
    if (err == -1) { // Non trouvé, création du fichier
        if (create_entry(name, desc) < 0)
            return -1;
        int new_ua = get_allocation_unit();
        if (new_ua < 0)
            return -1;
        desc->alloc[0] = new_ua;
    }
    // Si trouvé, allocation du BCF
    if (file == NULL)
        return -1;
    file->descriptor = *desc;
    unsigned char* buf = malloc(IO_BLOCK_SIZE);
    file->buffer = buf;
    strcpy(file->mode, "w");
    file->current_block = 0;
    file->file_offset = 0;
    file->block_offset = 0;
    // Préchargement du premier bloc de données
    if (read_block(desc->alloc[0], file->buffer) < 0)
        return -1;
    return 0;
}

/*
 * SYNOPSIS :
 *    int syr1_fwrite(SYR1_FILE* file, int item_size, int nbitem, char* buffer)
 * DESCRIPTION :
 *    Ce sous-programme écrit nbitem articles de taille item_size dans le
 *    fichier paramètre à partir du tampon mémoire.
 * PARAMETRES :
 *    file      : pointeur sur un descripteur de fichier
 *    item_size : taille d'un article
 *    nb_item   : nombre d'article à lire
 * RESULTAT :
 *    le nombre d'articles effectivement écrits dans le fichier, sinon un code
 *    d'erreur (cf syr1_putc())
```



```
 */
int syr1_fwrite(SYR1_FILE* file, int item_size, int nbitem, char* buffer) {
    int count = 0;
    while (count < nbitem*item_size) {
        int res = syr1_putc(buffer[count], file);
        if (res<0)
            return res;
        count++;
    }
    return count;
}

/*
 * SYNOPSYS :
 *   int syr1_putc(unsigned char c, SYR1_FILE *file)
 * DESCRIPTION :
 *   Ce sous-programme écrit un caractère dans le fichier passé en paramètre.
 * PARAMETRES :
 *   file : pointeur sur un Bloc Control Fichier (File Control Bloc)
 *      c : caractère à écrire
 * RESULTAT :
 *     0 : écriture réussie
 *    -1 : le descripteur de fichier logique est NULL, ou le mode d'ouverture
 *         du fichier passée en paramètre est incorrect
 *    -2 : erreur d'entrée-sorties sur le périphérique de stockage
 *    -3 : taille maximum de fichier atteinte
 *    -4 : plus de blocs disques libres
 */
int syr1_putc(unsigned char c, SYR1_FILE* file)  {
    if (file == NULL || file->mode[0] != 'w')
        return -1;
    // Si on est en fin de bloc de données
    if (file->block_offset >= IO_BLOCK_SIZE) {
        // On écrit le bloc courant
        write_block(file->descriptor.alloc[file->current_block], file->buffer);
        file->current_block++;
        if (file->current_block >= MAX_BLOCK_PER_FILE)
            return -3;
        // On récupère un nouveau bloc, ou le bloc suivant
        if (file->file_offset >= file->descriptor.size) {
            int new_au = get_allocation_unit();
            if (new_au == -1) // disque plein
                return -4;
            if (new_au == -2) // erreur E/S
                return -2;
            file->descriptor.alloc[file->current_block] = new_au;
        }
        if (read_block(file->descriptor.alloc[file->current_block], \
                file->buffer) < 0)
            return -2;
        file->block_offset = 0;
    }
    // On écrit le caractère voulu et on avance les compteurs
    file->buffer[file->block_offset] = c;
    file->block_offset++;
    file->file_offset++;
    if (file->file_offset > file->descriptor.size)
        file->descriptor.size = file->file_offset;
    return 0;
}

/*
```

```
 * SYNOPSYS :
 *    int syr1_fclose_write(SYR1_FILE* file) {
 * DESCRIPTION :
 *   Ce sous-programme gère la fermeture d'un fichier logique.
 * PARAMETRES :
 *   file : pointeur sur un Bloc de Contrôle Fichier (BCF)
 * RESULTAT :
 *    0 : la fermeture a réussi
 *   -1 : problème pendant la libération du descripteur de fichier logique
 *        (ou le fichier logiques file vaut NULL)
 *   -2 : erreur d'entrée-sorties sur le périphérique de stockage
 */
int syr1_fclose_write(SYR1_FILE* file) {
    if (file == NULL)
        return -1;
    // On écrit le buffer en cours
    if (write_block(file->descriptor.alloc[file->current_block], file->buffer) <
 0)
        return -2;
    if (update_entry(&file->descriptor) < 0)
        return -2;
    if (free_logical_file(file) < 0)
        return -1;
    return 0;
}
```