# SYSTEM D'EXPLOITATION

## Mini projet

**Réaliser par :**

NADRANI OUSSAMA 1311778906

TAZI Achraf : 1513755449

**Encadré par :**

**Mr MEKNASSI**

# • REMERCIEMENT

- Ce modeste travail que nous avons réalisé est le fruit des efforts fourni par Mr. MEKNASSI qui nous a poussés à donner le meilleur de nous-mêmes, et aussi grâce à sa pédagogie unique qui permet à l'étudiant de s'implique dans la construction de son cours, et nous permet ainsi d'élucidé toute ambiguïté concernant le module System d'exploitions qu'est le cœur de la formation d'un informaticien Nous tenons à vous remercier Mr. MEKNASSI pour vos efforts et vos précieux conseils qui nous permettent de développer nos compétences.

# • PLAN

# • ÉNONCÉ

- Ecrire un programme en C qui en utilisant les threads, gère le stock d'un hôpital qui est composé des produits suivants: pansements, ciseaux et antiseptiques. Dans ce programme vous créez trois threads dont chacun gère un de ces produits et dix threads pour la gestion des médecins qui veulent se procurer de ces produits médicamentaux.
- Les threads médecins prennent dans le stock et les threads du magasin vont réapprovisionner le stock dès qu'il devient trop bas pour satisfaire les clients (200 pansements, 60 ciseaux et 150 antiseptiques). Le nombre d'articles pris du stock sont des nombres aléatoires ainsi que l'ordre de passage des médecins.
- Le nombre d'articles pris du stock sont des nombres aléatoires qui ne dépassent pas une certaine valeur par demande à savoir : 85 pour les pansements, 05 pour les ciseaux et 15 pour l'antiseptique. L'ordre de passage des clients est aussi aléatoire.
- Au début du programme le stock contient les quantités suivantes: 1500 pansements, 250 ciseaux et 1100 antiseptiques.

# • PROGRAMME :

## • Structure stockant les informations des threads clients et du magasin .

**/* Structure stockant les informations des threads clients et du magasin. */**

**typedef struct{  int stock_pans;  int stock_cise; int stock_anti;**

**pthread_t thread_store_pans;**

**pthread_t thread_store_cise;**

**pthread_t thread_store_anti;**

**pthread_t thread_clients [NB_MEDECIN];**

**pthread_mutex_t mutex_stock_pans;**

**pthread_cond_t cond_stock_pans;**

**pthread_cond_t cond_clients_pans;**

**pthread_mutex_t mutex_stock_cise;**

**pthread_cond_t cond_stock_cise;**

**pthread_cond_t cond_clients_cise;**

**pthread_cond_t cond_clients_cise;**

**pthread_mutex_t mutex_stock_anti;**

**pthread_cond_t cond_stock_anti;**

**pthread_cond_t cond_clients_anti; }**

- **Fonction pour tirer un nombre au sort entre 0 et max .**

```
/* Fonction pour tirer un nombre au sort entre 0 et max. */
static int get_random (int max)
{
  double val;
  val = (double) max * rand ();
  val = val / (RAND_MAX + 1.0);
  return ((int) val);
}
```

- ## Fonction pour le thread du stock hopital .

```
/* Fonction pour le thread du magasin. */

static void * fn_store_pans (void * p_data)

{

  while (1)


     /* Debut de la zone protegee. */

     pthread_mutex_lock (& store.mutex_stock_pans);

     pthread_cond_wait (& store.cond_stock_pans, &
store.mutex_stock_pans);

       store.stock_pans = INITIAL_STOCK_pans;

 printf ("Remplissage du stock de %d pansements !\n",
store.stock_pans);

     pthread_cond_signal (& store.cond_clients_pans);

     pthread_mutex_unlock (& store.mutex_stock_pans);

     /* Fin de la zone protegee. */

}

return NULL;
```

- ## Fonction pour les threads des clients Medcin .

```
/* Fonction pour les threads des clients. */

static void * fn_clients (void * p_data)

{
```

```
   int nb = (int) p_data;
   while (1) {
       int produit= get_random(4);
       int val;
if(produit==1)
      { val = get_random (101);
      psleep (get_random (3));
 /* Debut de la zone protegee. */
      pthread_mutex_lock (& store.mutex_stock_pans);
      if (val > store.stock_pans)
 {
         pthread_cond_signal (& store.cond_stock_pans);
         pthread_cond_wait (& store.cond_clients_pans, &
store.mutex_stock_pans);
      }
store.stock_pans = store.stock_pans - val;
      printf ("Client %d prend %d du pansements, reste %d
en stock !\n " ,nb, val, store.stock_pans );
pthread_mutex_unlock (& store.mutex_stock_pans);
      /* Fin de la zone protegee. */}
if(produit==2)
      { val = get_random (11);
      psleep (get_random (3));
 /* Debut de la zone protegee. */
      pthread_mutex_lock (& store.mutex_stock_cise);
```

```
      if (val > store.stock_cise)
      {
        pthread_cond_signal (& store.cond_stock_cise);
        pthread_cond_wait (& store.cond_clients_cise, &
store.mutex_stock_cise);
      }
      store.stock_cise = store.stock_cise - val;
      printf ("Client %d prend %d du ciseaux, reste %d en
stock !\n",nb, val, store.stock_cise );
```

- **Fonction principale .**

```
int main (void)
{
int i = 0;
   int ret = 0;
   /* Creation des threads. */
   printf ("Creation du thread du pasement !\n");
   ret = pthread_create (
      & store.thread_store_pans, NULL,
      fn_store_pans, NULL );
   printf ("Creation du thread du ciseaux !\n");
   ret = pthread_create (& store.thread_store_cise, NULL,
fn_store_cise, NULL);
 printf ("Creation du thread du antiseptique !\n");
   ret = pthread_create (& store.thread_store_anti,
NULL,fn_store_anti, NULL);
```

```
 /* Creation des threads des clients si celui du magasinn a reussi. */
   if (! ret) {
      printf ("Creation des threads clients !\n");
      for (i = 0; i < NB_MEDECIN; i++) {
         ret = pthread_create ( & store.thread_clients [i], NULL,fn_clients, (void *) i);
if (ret)  {
           fprintf (stderr, "%s",strerror(ret));
        }
      }
   }
```

- **PROGRAMME GENERALE .**

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<unistd.h>
#include<signal.h>
#include<string.h>
#define psleep(sec) sleep ((sec))
#define INITIAL_STOCK_pans 1500
#define INITIAL_STOCK_cise 250
#define INITIAL_STOCK_anti 1100
#define NB_MEDECIN 10
/* Structure stockant les informations des threads clients et du magasin. */
typedef struct{
 int stock_pans;
 int stock_cise;
 int stock_anti;
 pthread_t thread_store_pans;
 pthread_t thread_store_cise;
 pthread_t thread_store_anti;
 pthread_t thread_clients [NB_MEDECIN];
 pthread_mutex_t mutex_stock_pans;
 pthread_cond_t cond_stock_pans;
 pthread_cond_t cond_clients_pans;
 pthread_cond_t mutex_stock_cise;
 pthread_cond_t cond_stock_cise;
 pthread_cond_t cond_clients_cise;
 pthread_mutex_t mutex_stock_anti;
```

```
static store_t store =
{
        .stock_pans = INITIAL_STOCK_pans,
        .stock_cise = INITIAL_STOCK_cise,
        .stock_anti = INITIAL_STOCK_anti,
        .mutex_stock_pans = PTHREAD_MUTEX_INITIALISER,
            .mutex_stock_cise =PTHREAD_MUTEX_INITIALISER,
                .mutex_stock_anti =PTHREAD_MUTEX_INITIALISER,
        .cond_clients_pans = PTHREAD_COND_INITIALISER,
                .cond_clients_cise = PTHREAD_COND_INITIALISER,
                        .cond_clients_anti = PTHREAD_COND_INITIALISER,
        .cond_clients_pans = PTHREAD_COND_INTIALISER,
                .cond_clients_cise = PTHREAD_CONS_INTIALISER,
                        .cond_clients_anti = PTHREAD_COND_INITIALISER,
};

        /* Fonction pour tirer un nombre au sort entre 0 et max. */
 static int get_random (int max) {
double val;
val = (double) max * rand ();
val = val / (RAND_MAX + 1.0);
return ((int) val);
}
```

```
        /* Fonction pour le thread du magasin. */
static void * fn_store_pans (void * p_data) { while (1)
/* Debut de la zone protegee. */
         pthread_mutex_lock (& store.mutex_stock_pans);
         pthread_cond_wait (& store.cond_stock_pans, & store.mutex_stock_pans);
         store.stock_pans = INITIAL_STOCK_pans;
        printf ("Remplissage du stock de %d pansements !\n",store.stock_pans);
        pthread_cond_signal (& store.cond_clients_pans);
        pthread_mutex_unlock (& store.mutex_stock_pans);
 /* Fin de la zone protegee. */
} return NULL;
static void * fn_store_cise (void * p_data) { while (1)
/* Debut de la zone protegee. */
         pthread_mutex_lock (& store.mutex_stock_cise);
         pthread_cond_wait (& store.cond_stock_cise, & store.mutex_stock_cise);
         store.stock_cise = INITIAL_STOCK_cise;
        printf ("Remplissage du stock de %d pansements !\n",store.stock_cise);
        pthread_cond_signal (& store.cond_clients_cise);
        pthread_mutex_unlock (& store.mutex_stock_cise);
 /* Fin de la zone protegee. */
} return NULL;
```

```
static void * fn_store_anti (void * p_data) { while (1)
/* Debut de la zone protegee. */
         pthread_mutex_lock (& store.mutex_stock_anti);
         pthread_cond_wait (& store.cond_stock_anti, & store.mutex_stock_anti);
         store.stock_anti = INITIAL_STOCK_anti;
        printf ("Remplissage du stock de %d pansements !\n",store.stock_anti);
        pthread_cond_signal (& store.cond_clients_anti);
        pthread_mutex_unlock (& store.mutex_stock_anti);
 /* Fin de la zone protegee. */
} return NULL;
```

```
        /* Fonction pour les threads des clients. */
static void * fn_clients (void * p_data) {
        int nb = (int) p_data;
        while (1) {
        int produit= get_random(4);
        int val;
        if(produit==1) {
        val = get_random (101);
         psleep (get_random (3));
        /* Debut de la zone protegee. */
        pthread_mutex_lock (& store.mutex_stock_pans);
         if (val > store.stock_pans) {
        pthread_cond_signal (& store.cond_stock_pans);
         pthread_cond_wait(& store.cond_clients_pans, & store.mutex_stock_pans);
 }
        store.stock_pans = store.stock_pans - val;
printf ("Client %d prend %d du pansements, reste %d en stock !
\n" ,nb,val,store.stock_pans );
 pthread_mutex_unlock (& store.mutex_stock_pans);
/* Fin de la zone protegee. */}
      if(produit==2) { val = get_random (11);
 psleep (get_random (3));
         /* Debut de la zone protegee. */
 pthread_mutex_lock (& store.mutex_stock_cise);
if (val > store.stock_cise) {
 pthread_cond_signal (& store.cond_stock_cise);
 pthread_cond_wait (& store.cond_clients_cise, & store.mutex_stock_cis
store.stock_cise = store.stock_cise - val;
 printf ("Client %d prend %d du ciseaux, reste %d en stock !\n",nb, va
store.stock_cise );}}}
```

```
   int main(void){
   int i = 0;
    int ret = 0;
         /* Creation des threads. */
 printf ("Creation du thread du pasement !\n");
 ret = pthread_create ( & store.thread_store_pans, NULL, fn_store_pans, NULL );
 printf ("Creation du thread du ciseaux !\n");
 ret = pthread_create (& store.thread_store_cise, NULL, fn_store_cise, NULL);
printf ("Creation du thread du antiseptique !\n");
 ret = pthread_create (& store.thread_store_anti, NULL,fn_store_anti, NULL);
 /* Creation des threads des clients si celui du magasinn a reussi. */
 if (! ret) {
 printf ("Creation des threads clients !\n");
 for (i = 0; i < NB_MEDECIN; i++) {
 ret = pthread_create (& store.thread_clients [i], NULL,fn_clients, (void *) i);
 if (ret)  { fprintf (stderr, "%s",strerror(ret)); }}}
else { fprintf (stderr, "%s",strerror(ret)); }
         /* Attente de la fin des threads. */
 i = 0;
 for (i = 0; i < NB_MEDECIN; i++) {
 pthread_join (store.thread_clients [i], NULL);
 }
 pthread_join (store.thread_store_pans, NULL);
 pthread_join (store.thread_store_cise, NULL);
 pthread_join (store.thread_store_anti, NULL);
 return EXIT_SUCCESS; }
```

- **EXÉCUTION :**