# Algorithme principale:

## Constante :

MaxPansem =2500
MaxCiseaux= 250
MaxAntisep= 1500

MinPansem =150
MinCiseaux =70
MinAntisep =180

## Variable :

pthread_mutex_init(&lock,NULL)

pthread_t thPansem, thCiseaux, thAntisep

pthread_t th[20]

tirePansem, tireCiseaux, tireAntisep , Med : entire

srand(time(NULL))

nbPansem : entier

nbCiseaux : entier

nbAntisep : entier

tirePansem, tireCiseaux, tireAntisep : entier

pthread_mutex_t lock

## Debut

pthread_cond_t condPansem□PTHREAD_COND_INITIALIZER

pthread_cond_t condCiseau□PTHREAD_COND_INITIALIZER

pthread_cond_t condAntisep□PTHREAD_COND_INITIALIZER

nbPansem □ MaxPansem

nbCiseaux □ MaxCiseaux

nbAntisep □ MaxAntisep

Ecrire("Cliquer sur ENTRER pour commencer le programme :")

Lire()

Ecrire ("Creation de thread de Panesment")

pthread_create(&thPansem,NULL,GesPansem,NULL)

Ecrire ("Creation de thread de Ciseaux")

pthread_create(&thCiseaux,NULL,GesCiseaux,NULL)

Ecrire ("Creation de thread de Antiseptiques")

pthread_create(&thAntisep,NULL,GesAnti,NULL)

## Tant que(1)

Med=rand()%20+1

pthread_create(&th[Med],NULL,client,&Med)

pthread_create(&thPansem,NULL,GesPansem,NULL)

pthread_create(&thCiseaux,NULL,GesCiseaux,NULL)

pthread_create(&thAntisep,NULL,GesAnti,NULL)

pthread_join(th[Med],NULL)

pthread_join(thPansem,NULL)

pthread_join(thCiseaux,NULL)

pthread_join(thAntisep,NULL)

**FIN TANTQUE**

pthread_cond_signal(&condStart)

sleep(1)

pthread_cond_destroy (&condPansem)

pthread_cond_destroy (&condCiseau)

pthread_cond_destroy (&condAntisep)

pthread_mutex_lock(&lock)

pthread_mutex_destroy(&lock)

*FIN*

## Fonction qui gere les clients :

Foction client(c :entier) : retour vide

- Variable :

  pthread_mutex_lock(&lock)
  pthread_cond_wait (&condStart,&lock)

- Debut :

  srand(time(NULL));
  tirePansem rand()%80+1

  tireCiseaux rand()%15+1

tireAntisep rand()%25+1

Ecrire("Medcine prend: " Pansements, Ciseaux "et " Antiseptiques. )

Ecrire ( "Le reste: " nbPansem , " Pansements ", nbCiseaux "Ciseaux et " nbAntisep " Antiseptiques")

pthread_cond_signal (&condPansem)

pthread_cond_signal (&condCiseau)

pthread_cond_signal (&condAntisep)

pthread_mutex_unlock(&lock)

*FIN*

## Fonction qui gere le stock de pansements:

### Fonction GesPansem() : retour vide

- Variable:

pthread_mutex_lock(&lock)

- Debut:

pthread_cond_wait (&condPansem,&lock)

nbPansem  nbPansem - tirePansem

**si** (nbPansem < MinPansem)

Ecrire("Le stocke de Pansements est faible (nbPansem)Remplissage")

nbPansem MaxPansem

**Fin si**

pthread_mutex_unlock(&lock)

*FIN*

## Fonction qui gere le stock de ciseaux:

Fonction GesCiseaux() :reteur vide

- Variable:

pthread_mutex_lock(&lock)

- Debut:

pthread_cond_wait (&condCiseau,&lock)

nbCiseaux  nbCiseaux – tireCiseaux

**si** (nbCiseaux < MinCiseaux)

("Le stocke de Ciseaux est faible" (nbCiseaux), "Remplissage ... ")

nbCiseaux MaxCiseaux

**Fin si**

pthread_mutex_unlock(&lock)

*FIN*

## Fonction qui gere le stock de antiseptiques :

Fonction GesAnti() :retour vide

 Variable:

pthread_mutex_lock(&lock);

 Debut :

pthread_cond_wait (&condAntisep,&lock)

nbAntisep  nbAntisep - tireAntisep

**si**(nbAntisep < MinAntisep)

Ecrire("Le stocke de Antiseptiques est faible (%d), Remplissage ...\n");

nbAntisepnbAntisep -MaxAntisep

**Fin si**

pthread_mutex_unlock(&lock)

*FIN*

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<time.h>
#include<unistd.h>
// define les canstantes :
#define MaxPansem 2500
#define MaxCiseaux 250
#define MaxAntisep 1500

#define MinPansem 150
#define MinCiseaux 70
#define MinAntisep 180
//declaration des variables globals
int
int
int
int
pthread_mutex_t
pthread_cond_t
pthread_cond_t
pthread_cond_t
pthread_cond_t

void*          void*      //getion des medcines
 pthread_mutex_lock
 pthread_cond_wait
 int *
         int*     //casting un void * à un entier
 srand time NULL
                rand()   80  1 //contité de pensement
                rand()   15  1 //contité de Ciseaux
                rand()   25  1  //contité de Antisep
```

```
printf "Medcine numéro %d prend:\n\t\t[%d Pansements] \t[%d Ciseaux] \t
  [%d Antiseptiques]\n",
 printf "=====>Le reste: [%d Pansements] \t[%d Ciseaux]\t[%d Antiseptiques]
  \n\n"
   printf("_____\n"
//le thread envoi un signal a la fonction GesPansem pour lancer le traitement
pthread_cond_signal
  //le thread envoi un signal a la fonction GesCiseaux pour lancer le traitement
 pthread_cond_signal
  //le thread envoi un signal a la fonction GesCiseaux pour lancer le traitement
 pthread_cond_signal
  //le thread envoi un signal a la fonction GesAnti pour lancer le traitement
 pthread_mutex_unlock
 void*
 pthread_mutex_lock
  //reçoi le signal de thread de medcine
 pthread_cond_wait
                                  //traitement
 if                               //condiction pour remplire le stock
   printf "Le stocke de Pansements est faible (%d),Remplissage\n"
   printf "_____\n"


 pthread_mutex_unlock

void*
 pthread_mutex_lock
 //reçoi le signal de thread de medcine
 pthread_cond_wait
                                //traitement
 if                               //condiction pour remplire le stock
   printf "Le stocke de Ciseaux est faible (%d),Remplissage\n"
   printf("_____\n"
```

```
 pthread_mutex_unlock

void*
 pthread_mutex_lock
 //reçoi le signal de thread de medcine
 pthread_cond_wait
                                    //traitement
 if                                   //condiction pour remplire le stock
   printf "Le stocke de Antiseptiques est faible (%d),Remplissage\n"

printf "_____________________________________________________\n"


 pthread_mutex_unlock

//____________________Main Code_____________________
int         int         char const
 pthread_mutex_lock
 pthread_mutex_init          NULL
 //declaration de threades de gestion de stock
 pthread_t
  //declaration de threades de gestion de medcines
 pthread_t     20
 //variables alleatoires des contité
 int
 int
 int
 srand time NULL
 printf "-------------------------------------------------\n"
 printf "Realisé par :\n"
 printf "\tILIASS DAHMAN         group 1.\n"
 printf "\tMOHAMED EL KHALFAOUI    group 2.\n\n"
 printf "-------------------------------------------------\n"
```

```
printf "Cliquer sur ENTRER pour commencer le programme :"

while (1)
                  20  1     //choix de medcine
 //Creation de thread de medcine alleatoire
         pthread_create              NULL
 if            -1
   printf

 //Creation de thread de Panesment
        pthread_create                                  NULL
if             -1
 printf                                                  \n

 //Creation de thread de Ciseaux
        pthread_create              NULL              NULL
if             -1
 printf                                              \n

//Creation de thread de Antiseptiques
        pthread_create              NULL          NULL
if             -1
 printf "Erreur de la creation de thread de Antiseptiques\n"

pthread_cond_signal
         1

        pthread_join              NULL
if             0
   printf "erreur de join thread de Pansements\n"

        pthread_join              NULL
if             0
```

```
    printf "erreur de join thread de Ciseaux\n"

        pthread_join                NULL
if              0
    printf "erreur de join thread de Antiseptiques\n"

         pthread_join               NULL
 if              0
     printf "erreur de join thread de medcine \n"

 pthread_cond_destroy
 pthread_cond_destroy
 pthread_cond_destroy
 pthread_mutex_lock
 pthread_mutex_destroy
return 0
```