SR02 : TD 8

TD Machine sur une séance de 2h

Objectifs

- **Utiliser les threads POSIX**

Exercice 1. (Threads et X11 et Rendez-vous)

Compiler, linker et exécuter le programme pth.c

```
/*-------------------------------------------------------*/
/* > gcc -o pth pth.c -L/usr/X11R6/lib/ -lpthread */
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#define nth 3 /* nbre de threads a lancer */
#define ifer(is,mess) if (is==-1) perror(mess)
pthread_t threads[nth];
/* routine exécutée dans les threads */
void *th_fonc (void *arg) {
   int is, numero, i;
   numero = (int)arg;
   for (i=1;i<=5;i++) {
      printf("ici thread %d, i=%d\n",numero,i);
      sleep(1);
   }
   return ((void *)(numero+100));
}
main() {
int is, i;
```

```
void *val=0;
/* créer les threads */
for(i=0; i<nth; i++) {
   printf("ici main, création thread %d\n",i);
   is = pthread_create( &threads[i], NULL, th_fonc, (void *)i );
   ifer (is,"err. création thread");
}
/* attendre fin des threads */
for(i=0; i<nth; i++) {
   is = pthread_join( threads[i], &val);
   ifer (is,"err. join thread");
   printf("ici main, fin thread %d\n",(int)val);
}
}
```

- Compiler, linker et exécuter le programme bar.c

On trouvera ici barx.o nécessaire pour ce link

barx.o

```
/* bar.c exemple X11 * appelle routines definies dans barx.c *
> gcc -o bar bar.c barx.o -L/usr/X11R6/lib/ -lX11 */
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
char buf[20];
/* ---------------------------------------------- */
void *b_fonc (void * arg) {
```

```
int is, numero, i,j, m1;
numero = (int)arg;
m1 = 20;
i = m1;
printf("numero= %d, i=%d \n",numero,i);
drawstr (30, 125, "_0_", 3);
drawrec (100,100,100+m1*10,30);
for (j=1;j<=m1;j++) {
   printf("num %d j=%d\n",numero,j);
   fillrec (100,102,100+j*10,26,"yellow");
}
flushdis ();
return ( (void *)(numero+100) );
}
int liretty (char *prompt, char *buffer) {
int i;
printf("\n%s",prompt);
i = scanf ("%s",buffer);
return strlen(buffer);
}
main () {
int nlu, is, i=0;
initrec();
/* creer rectangle rouge */
is = (int)b_fonc( (void *)i );
```

printf("is= %d\n",is);

nlu = liretty("sortir ?",buf);

printf("--fin--\n");

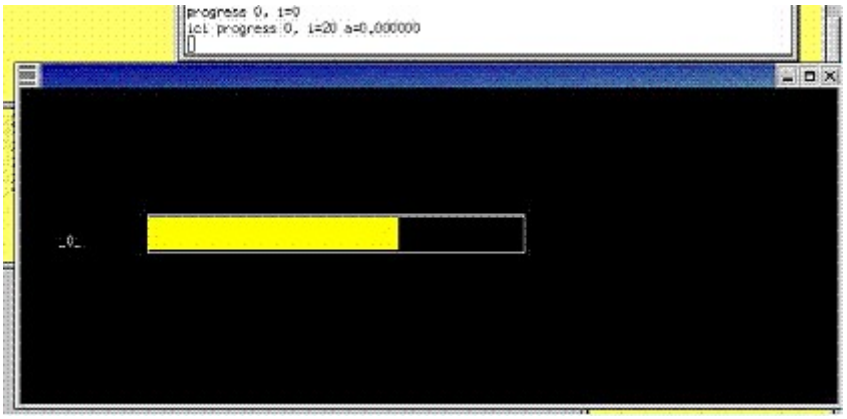
detruitrec();

/ detruire la fenetre rectangle */*

exit(EXIT_SUCCESS);

}

execution



Indice :

Ce programme bar.c utilise les routines suivantes :

/* -- */

Fonctions de dessin appelées : (ces fonctions ne sont PAS des fonctions de X11: elles sont définies dans barx.o pour simplifier la réalisation du TD)

initrec(); / creer fenetre rectangle rouge */*

detruitrec(); / detruire la fenetre rectangle */*

flushdis (); / flush display buffer */*

drawstr (x,y,string,long); / écrire string en x,y */*

drawrec (x,y,larg,hter); / dessine rect. en blanc */*

fillrec (x,y,larg,hter,col1); / remplir rect. avec col1 */*

exemples :

drawstr (30, 125, "_0_", 3);

*drawrec (100,100,100+m1*10,30);*

*fillrec (100,102,100+j*10,26,"yellow");*

/ ---------- */*

Fonction de mise en attente d'un thread Fonction :

thread_wait(numth,s,c); /* sec. et c/100 sec. */ appelée par le thread de numéro "numth", elle met ce thread en attente pour "s" secondes et "c" centièmes.

thread_wait doit être initialisée avant utilisation par: thread_wait(-1,0,0);

Prototype : void thread_wait(int th, unsigned long sec, long csec)

Exemple :

thread_wait(2,1,40); /* thread 2 attend 1s. 40/100 */

- Ecrire un programme thx.c répondant à l'algorithme ci-dessous :

Main :

- initialisations

- création de 3 threads exécutant la fonction th_fonc

- attente de la fin des threads

th_fonc :

si thread numéro i :

- écrire numéro dans fenêtre

- tracer un rectangle vide

- boucle de longueur mi :

printf message

k = compteur_global

th_wait()

faire progresser rectangle rempli

k++;

compteur_global = k;

à la fin de thread i :

- protéger par mutex : total = total + mi

- renvoyer mi comme valeur de retour de th_fonc

ATTENTION :

Dans cette version multi-threads, tout appel à une fonction de la librairie graphique barx.o (utilisant Xwindow) doit être protégé par un mutex. En effet, il s'agit de sections critiques. Utilisez par exemple mutexbarx pour protéger ces appels.

Indice :

thx.c utilisera les "include" suivants :

/* thx.c exemple threads+X11 *

appelle routines definies dans thw.c et barx.c *

> gcc -c barx.c *

> gcc -c thw.c *

> gcc -o thx thx.c thw.o barx.o -L/usr/X11R6/lib/ -lX11 -lpthread */

/* main lance des threads qui chacun: *

fait progresser un rectangle horizontal, */

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#include <signal.h>

#include <pthread.h>

thx.c utilisera les fonctions "pthread" (Posix threads) et déclarations suivantes :

/ thx.c exemple threads+X11 */*

*/*gcc -o thx thx.c thw.o barx.o -lX11 -lpthread */*

#include <pthread.h>

pthread_t threads[nth];

pthread_mutex_t mutex;

pthread_cond_t event;

is = pthread_mutex_init(&mutex, NULL);

is = pthread_cond_init (&event, NULL);

is = pthread_mutex_lock(&mutex);

is = pthread_mutex_unlock(&mutex);

is = pthread_cond_signal(&event);

is = pthread_cond_wait(&event,&mutex);

*is = pthread_create(&threads[i], NULL, th_fonc, (void *)i);*

is = pthread_join(threads[j], &val);

On trouvera ici thw.o et barx.o nécessaires pour "linker" thx.c.

barx.o et thw.o

L'exécutable "thx", crée une fenêtre ayant l'aspect suivant :

thx (exécutable à tester)

Fenêtre créée par thx

-Ecrire un programme thrdv.c répondant à l'algorithme ci-dessous :

Identique à thx.c ci-dessus auquel on ajoute:

Dans chaque fonction th_fonc:

- si on a déjà fait progresser le rectangle de couleur du thread de 10 fois, alors appeler une fonction rdv();

Comme sur la copie d'écran ci-dessous, on ajoutera sur le dessin une marque verticale à l'endroit du rendez-vous.

fonction rdv() :

- si le thread appelant n'est PAS le denier qui arrive alors, il s'endort,

- si c'est le dernier arrivé alors il provoque le réveil de tous les autres et il continue.

Il y a deux façons de coder la fonction rendez-vous :

1. par un pthread_cond_wait() et un pthread_cond_broadcast() (6 lignes de code)
2. par un pthread_cond_wait() et un pthread_cond_signal() (7 lignes de code)

Vous programmerez les deux : thrdva et thrdvb.

Indice :

La copie d'écran ci-dessous montre les deux threads 0 et 1 bloqués sur le rendez-vous dans l'attente de l'arrivée du dernier.

thrdv

On trouvera ici thrdv, un binaire qui montre le fonctionnement du rendez-vous.

thrdv