



المعهد الوطني للبريد والمواصلات
المعهد الوطني للبريد والمواصلات
Institut National des Postes et Télécommunications

SUD341 : systèmes temps réel et embarqués

TP3 le temps réel sous LINUX: L'API POSIX



Réalisé par :

EL MEKKAOUI Fayssal
HAJJAJI Ayyoub
ELAMMARI Souhail

Professeur :

Mr EN-NOUAARY Abdeslam

SUD (Cloud & IoT) 2^{ème} année

Dimanche 20 décembre 2020

I. Introduction :

Dans ce TP nous allons essayer d'apprendre le développement en temps réel sous les systèmes d'exploitation de type UNIX/LINUX à travers l'interface de programmation standard POSIX (Portable Operating System Interface for UNIX) définie depuis 1988 par IEEE sur proposition de Monsieur Richard Mathew Stallman. La norme présente principalement une bibliothèque riche pthreads qui facilite la programmation concurrentielle/multitâches ainsi que les aspects s'y relatant tels que la communication et la synchronisation.

II, Exemple 1 :

Le programme ci-dessous est une illustration de la gestion des threads POSIX. Il permet de créer et lancer deux threads concurrents. Le premier thread permet d'afficher un message de bienvenue et de lire un entier au clavier puis de le retourner au thread principal pour affichage sur la sortie standard.

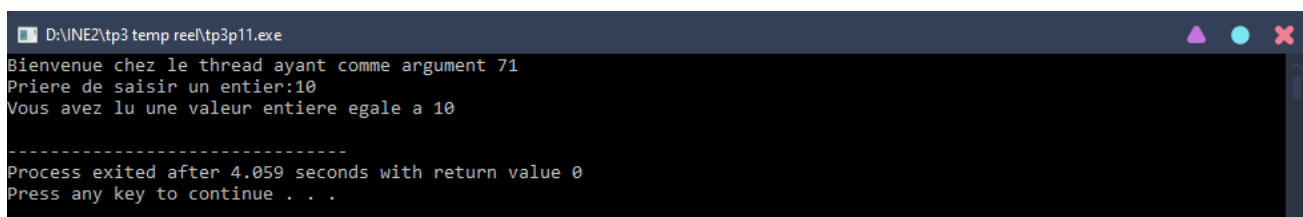
```

tp3p11.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include <pthread.h>
5
6  //La fonction du thread
7  void* lire_entier(void *arg) {
8      int un_entier;
9      int val_arg = (int) arg;
10     printf("Bienvenue chez le thread ayant comme argument %d\n", val_arg);
11     printf("Priere de saisir un entier:");
12     scanf("%d", &un_entier);
13     pthread_exit((void *) un_entier);
14 } //fin lire_entier
15
16 int main(void) {
17     int i, val_retour;
18     pthread_t thread1;
19     srand(time(NULL));
20     i = 1 + rand() % 100;
21     int succes = pthread_create(&thread1, NULL, lire_entier, (void *) i);
22     if (succes != 0) {
23         fprintf(stderr, "Erreur de creation du thread ...");
24         exit(0);
25     } //fin if (succes != 0)
26
27     //Attente du thread de lecture
28     pthread_join(thread1, (void *)&val_retour);
29     printf("Vous avez lu une valeur entiere egale a %d\n", val_retour);
30     return 0;
31 } //fin main()

```

Le thread principal initialise un nouveau thread nommé « thread1 », La fonction « srand » permet d'initialiser le générateur de nombres aléatoires (la fonction rand) fournit par la librairie C standard. Après avoir générer un nombre aléatoirement.

Execution du code :



```

D:\INE2\tp3 temp reel\tp3p11.exe
Bienvenue chez le thread ayant comme argument 71
Priere de saisir un entier:10
Vous avez lu une valeur entiere egale a 10

-----
Process exited after 4.059 seconds with return value 0
Press any key to continue . . .

```

Maintenant on ajoutez d'autres threads dans le programme pour afficher l'entier lu aussi bien en décimal qu'en hexadécimal et qu'en octal. Le code devient de la forme suivante :

```
[*] tp3p11.c
12  scanf("%d", &un_entier);
13  pthread_exit((void *) un_entier);
14  } //fin lire_entier
15
16  // fonction d'affichage du nombre en octal
17  void* afficher_octal(void *arg) {
18      printf("Vous avez lu une valeur octal egale a : %o\n", (int) arg);
19  }
20
21  // fonction d'affichage du nombre en hexadecimal
22  void* afficher_hex(void *arg) {
23      printf("Vous avez lu une valeur hexadecimal egale a : %x\n", (int) arg);
24  }
25
26  int main(void) {
27      clock_t time_req;
28      time_req = clock();
29      pthread_t thread1;
30      int i, val_retour;
31      srand(time(NULL));
32      i = 1 + rand() % 100;
33
34      int succes = pthread_create(&thread1, NULL, lire_entier, (void *) i);
35      if (succes != 0) {
36          fprintf(stderr, "Erreur de creation du thread ...");
37          exit(0);
38      } //fin if (succes != 0)
39
40      pthread_join(thread1, (void *)&val_retour);
41      printf("Temps pour la lecture : %d\n", clock() - time_req);
42      time_req = clock();
43      printf("Vous avez lu une valeur entiere egale a : %d\n", val_retour);
44      pthread_t thread2, thread3;
45      pthread_create(&thread2, NULL, afficher_octal, (void *) val_retour);
46      pthread_create(&thread3, NULL, afficher_hex, (void *) val_retour);
47      pthread_join(thread2, NULL);
48      pthread_join(thread3, NULL);
49      printf("Temps pour l'affichage' : %d\n", clock() - time_req);
50      time_req = clock();
51      return 0;
52  } //fin main()
```

Sel: 0 Lines: 52 Length: 1567 Insert Done parsing in 0.047 seconds

Exécution :

```
D:\INE2\tp3 temp reel\tp3p11.exe
Bienvenue chez le thread ayant comme argument 10
Priere de saisir un entier:30
Vous avez lu une valeur entiere egale a : 30
Vous avez lu une valeur octal egale a : 36
Vous avez lu une valeur hexadecimal egale a : 1e
-----
Process exited after 4.517 seconds with return value 0
Press any key to continue . . .
```

Nous allons modifier le programme pour qu'il affiche le temps pris par chaque opération, la lecture, l'affichage et aussi son temps de réponse depuis son lancement jusqu'à terminaison.

```
D:\INE2\tp3 temp reel\tp3p11.exe
Bienvenue chez le thread ayant comme argument 66
Priere de saisir un entier:15
Temps pour la lecture : 2554
Vous avez lu une valeur entiere egale a : 15
Vous avez lu une valeur octal egale a : 17
Vous avez lu une valeur hexadecimal egale a : f
Temps pour l'affichage' : 0
-----
Process exited after 2.643 seconds with return value 0
Press any key to continue . . .
```

III. Exemple 2 :

Dans cet exemple nous allons exécuter un programme qui permet d'illustrer la synchronisation entre les threads POSIX en utilisant des mutexes et des variables de conditions. Le but du programme est de gérer des transactions bancaires sur un compte qui est alimenté régulièrement pour accommoder les dépenses personnelles. Ensuite nous allons essayer d'enlever quelques lignes et comparer les résultats.

- Exécution sans modification :

```

D:\INE2\tp3 temp reel\tp3p21.exe
Creation du thread de la banque!
Creation des threads clients !
Client 4 prend 0 dirhams, reste 200 en solde!
Client 3 prend 0 dirhams, reste 200 en solde!
Client 2 prend 0 dirhams, reste 200 en solde!
Client 1 prend 0 dirhams, reste 200 en solde!
Client 0 prend 0 dirhams, reste 200 en solde!
Client 0 prend 11 dirhams, reste 189 en solde!
Client 1 prend 11 dirhams, reste 178 en solde!
Client 2 prend 11 dirhams, reste 167 en solde!
Client 3 prend 11 dirhams, reste 156 en solde!
Client 4 prend 11 dirhams, reste 145 en solde!
Client 0 prend 35 dirhams, reste 110 en solde!
Client 4 prend 35 dirhams, reste 75 en solde!
Client 3 prend 35 dirhams, reste 40 en solde!
Client 2 prend 35 dirhams, reste 5 en solde!
      Alimentation du compte bancaire avec 200 dirhams!
Client 1 prend 35 dirhams, reste 165 en solde!
Client 1 prend 21 dirhams, reste 144 en solde!
Client 2 prend 21 dirhams, reste 123 en solde!
Client 3 prend 21 dirhams, reste 102 en solde!
Client 4 prend 21 dirhams, reste 81 en solde!
Client 0 prend 21 dirhams, reste 60 en solde!
Client 0 prend 49 dirhams, reste 11 en solde!
      Alimentation du compte bancaire avec 200 dirhams!
Client 4 prend 49 dirhams, reste 151 en solde!
Client 0 prend 10 dirhams, reste 141 en solde!
Client 4 prend 10 dirhams, reste 131 en solde!
Client 0 prend 42 dirhams, reste 89 en solde!
Client 4 prend 42 dirhams, reste 47 en solde!
Client 0 prend 18 dirhams, reste 29 en solde!
Client 4 prend 18 dirhams, reste 11 en solde!
Client 4 prend 5 dirhams, reste 6 en solde!
Client 0 prend 5 dirhams, reste 1 en solde!
      Alimentation du compte bancaire avec 200 dirhams!
Client 0 prend 8 dirhams, reste 192 en solde!
Client 3 prend 49 dirhams, reste 143 en solde!
Client 0 prend 59 dirhams, reste 84 en solde!
Client 0 prend 7 dirhams, reste 77 en solde!
Client 0 prend 0 dirhams, reste 77 en solde!

```

le solde initial du compte bancaire est 200dh. Ensuite, chaque client prend une somme d'argent qui est générée aléatoirement entre 1 et 50 dirhams. Si la valeur du solde bancaire est insuffisante pour débiter la valeur générée, on le crédite d'un montant de 200 dirhams.

- On Enlève les variables de conditions :

```

D:\VINE2\tp3 temp reel\tp3p21.exe
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Client 3 prend 11 dirhams, reste 189 en solde!
Client 1 prend 11 dirhams, reste 178 en solde!
Client 0 prend 11 dirhams, reste 167 en solde!
Client 2 prend 11 dirhams, reste 156 en solde!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Client 4 prend 11 dirhams, reste 189 en solde!
Alimentation du compte bancaire avec 200 dirhams!

```

Lorsque on enlève variables de condition, la probabilité pour que le verrou « mutex_balance » soit occupé par thread client est très petit, alors le solde va être presque toujours égal à 200dhs.

- On enlève les mutexes :

```

D:\VINE2\tp3 temp reel\tp3p21.exe
Alimentation du compte bancaire avec 200 dirhams!
Client 4 prend 49 dirhams, reste 151 en solde!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Client 2 prend 49 dirhams, reste 102 en solde!
Client 1 prend 49 dirhams, reste 151 en solde!
Client 0 prend 49 dirhams, reste 102 en solde!
Client 3 prend 49 dirhams, reste 151 en solde!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!
Alimentation du compte bancaire avec 200 dirhams!

```

Lorsqu'on enlève les mutexes, le solde de compte bancaire après chaque transaction peut être erroné

- Le taux d'utilisation

```

D:\VINE2\tp3 temp reel\tp3p21.exe
                                temps prise par transaction : 1
Client 0 prend 35 dirhams, reste 2 en solde!
                                temps prise par transaction : 1989
                                Alimentation du compte bancaire avec 200 dirhams!
Client 2 prend 35 dirhams, reste 165 en solde!
                                temps prise par transaction : 6
Client 0 prend 15 dirhams, reste 150 en solde!
                                temps prise par transaction : 1
Client 1 prend 35 dirhams, reste 115 en solde!
                                temps prise par transaction : 2
Client 2 prend 15 dirhams, reste 100 en solde!
                                temps prise par transaction : 1
Client 3 prend 35 dirhams, reste 65 en solde!
                                temps prise par transaction : 1
Client 4 prend 35 dirhams, reste 30 en solde!
                                temps prise par transaction : 1
Client 1 prend 15 dirhams, reste 15 en solde!
                                temps prise par transaction : 1
Client 3 prend 15 dirhams, reste 0 en solde!
                                temps prise par transaction : 1
                                Alimentation du compte bancaire avec 200 dirhams!
Client 4 prend 15 dirhams, reste 185 en solde!
                                temps prise par transaction : 3
Client 0 prend 4 dirhams, reste 181 en solde!
                                temps prise par transaction : 976
Client 0 prend 7 dirhams, reste 174 en solde!
                                temps prise par transaction : 0
Client 2 prend 4 dirhams, reste 170 en solde!
                                temps prise par transaction : 1
Client 2 prend 7 dirhams, reste 163 en solde!

```

- On vérifie si la limite de 1000 dirhams par jour est toujours respectée :

Maintenant, nous allons modifier notre programme pour qu'on peut mettre une limite a notre bancaire pour fixer le plafond de ces dépenses à 1000 dirhams. Après on exécute notre programme

```

D:\VINE2\tp3 temp reel\tp3p21.exe
                                temps prise par transaction : 1
Client 4 prend 0 dirhams, reste 136 en solde!
                                temps prise par transaction : 4
                                impossible de prend 0 dirhams, la limite de 1000dh par jour!
                                impossible de prend 3 dirhams, la limite de 1000dh par jour!
                                impossible de prend 2 dirhams, la limite de 1000dh par jour!
                                impossible de prend 1 dirhams, la limite de 1000dh par jour!
                                impossible de prend 4 dirhams, la limite de 1000dh par jour!
                                impossible de prend 3 dirhams, la limite de 1000dh par jour!
                                impossible de prend 0 dirhams, la limite de 1000dh par jour!
                                impossible de prend 2 dirhams, la limite de 1000dh par jour!
                                impossible de prend 1 dirhams, la limite de 1000dh par jour!
                                impossible de prend 4 dirhams, la limite de 1000dh par jour!
                                impossible de prend 3 dirhams, la limite de 1000dh par jour!
                                impossible de prend 0 dirhams, la limite de 1000dh par jour!
                                impossible de prend 2 dirhams, la limite de 1000dh par jour!
                                impossible de prend 1 dirhams, la limite de 1000dh par jour!
                                impossible de prend 4 dirhams, la limite de 1000dh par jour!
                                impossible de prend 3 dirhams, la limite de 1000dh par jour!
                                impossible de prend 0 dirhams, la limite de 1000dh par jour!
                                impossible de prend 2 dirhams, la limite de 1000dh par jour!
                                impossible de prend 1 dirhams, la limite de 1000dh par jour!
                                impossible de prend 4 dirhams, la limite de 1000dh par jour!
Client 3 prend 2 dirhams, reste 134 en solde!
                                temps prise par transaction : 52058
                                impossible de prend 0 dirhams, la limite de 1000dh par jour!
                                impossible de prend 2 dirhams, la limite de 1000dh par jour!
                                impossible de prend 1 dirhams, la limite de 1000dh par jour!
                                impossible de prend 4 dirhams, la limite de 1000dh par jour!

```

IV. Exemple 3 :

Dans cette eemple nous allons essayer d'exécuter un programme qui illustre l'utilisation des files de messages pour la communication entre les threads/processus. Il s'agit de développer une petite application à deux threads communicants, un émetteur et un récepteur, pour calculer le temps que prend les messages qui transitent entre les deux parties

D'abord nous devons créer les files de l'émetteur et le récepteur par la commande `gcc -o -lrt`

```
user@instant-contiki: ~/Desktop
user@instant-contiki:~$ cd
user@instant-contiki:~$ cd Desktop/
user@instant-contiki:~/Desktop$ gcc reciever.c -o reciever -lrt
user@instant-contiki:~/Desktop$ gcc sender.c -o sender -lrt
user@instant-contiki:~/Desktop$ ./sender /msg
```

Puis on ouvre de terminal et on tape dans le premier la comande `./sender /msg` (nom de la file message)
Et `./reciver /msg` dans l'autre.

```
user@instant-contiki: ~/Desktop
user@instant-contiki:~$ cd Desktop/
user@instant-contiki:~/Desktop$ ./reciever /msg

user@instant-contiki: ~/Desktop
user@instant-contiki:~$ cd
user@instant-contiki:~$ cd Desktop/
user@instant-contiki:~/Desktop$ gcc reciever.c -o reciever -lrt
user@instant-contiki:~/Desktop$ gcc sender.c -o sender -lrt
user@instant-contiki:~/Desktop$ ./sender /msg
```

Finalement on observe les résultats :

```
user@instant-contiki: ~/Desktop
user@instant-contiki:~$ cd Desktop/
user@instant-contiki:~/Desktop$ ./reciever /msg
min = 1    max =5924 moy= 24.7
min = 2    max =3132 moy= 19.3
min = 1    max =1036 moy=  7.3
min = 1    max =996  moy=  6.8
min = 1    max =977  moy=  6.7
min = 1    max =1383 moy=  7.4
min = 1    max =900  moy=  6.8
min = 1    max =993  moy=  6.9
min = 1    max =1424 moy=  6.9
min = 1    max =924  moy=  6.9
min = 1    max =953  moy=  6.8
min = 1    max =870  moy=  6.8
min = 1    max =350  moy=  6.7
min = 1    max =900  moy=  6.7
min = 1    max =896  moy=  6.7
min = 6    max =901  moy=  6.9

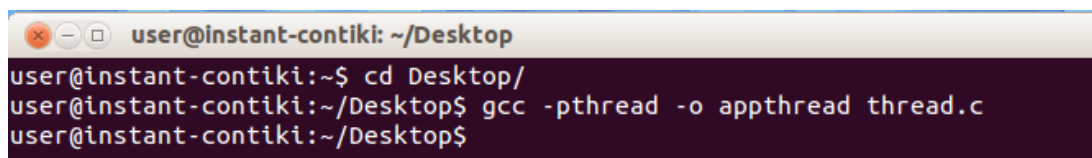
```


V. Étude de cas :

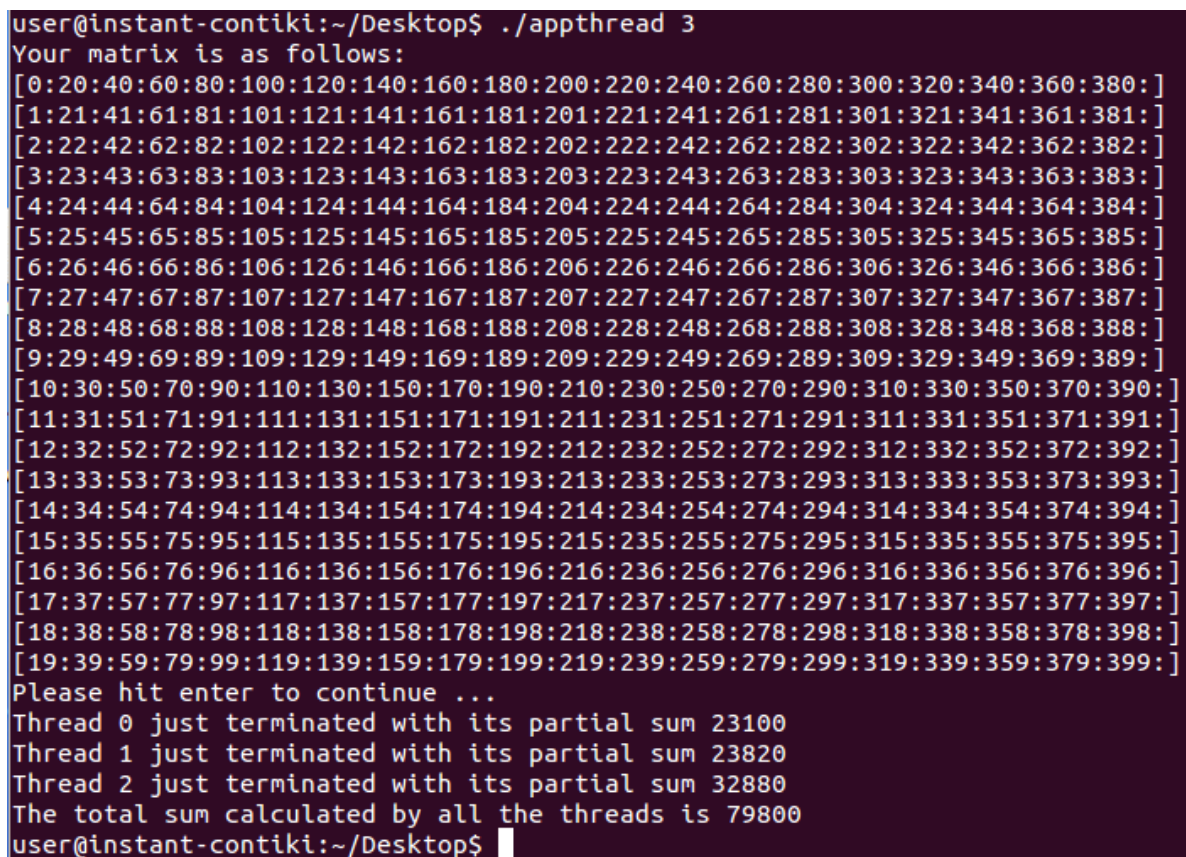
Dans cette partie nous allons développer une application permettant de traiter des données massives (BigData) à travers une grande matrice carrée ($n \times n$) dont les valeurs représentent des températures collectées en temps réel en interrogeant régulièrement, pendant une période donnée, des capteurs déployés dans une zone géographique sensible.

D'abord nous allons essayer d'exécuter un code qui donne une solution partielle :

On le compile avec la commande `gcc -pthread -o` ,



```
user@instant-contiki: ~/Desktop
user@instant-contiki:~$ cd Desktop/
user@instant-contiki:~/Desktop$ gcc -pthread -o appthread thread.c
user@instant-contiki:~/Desktop$
```



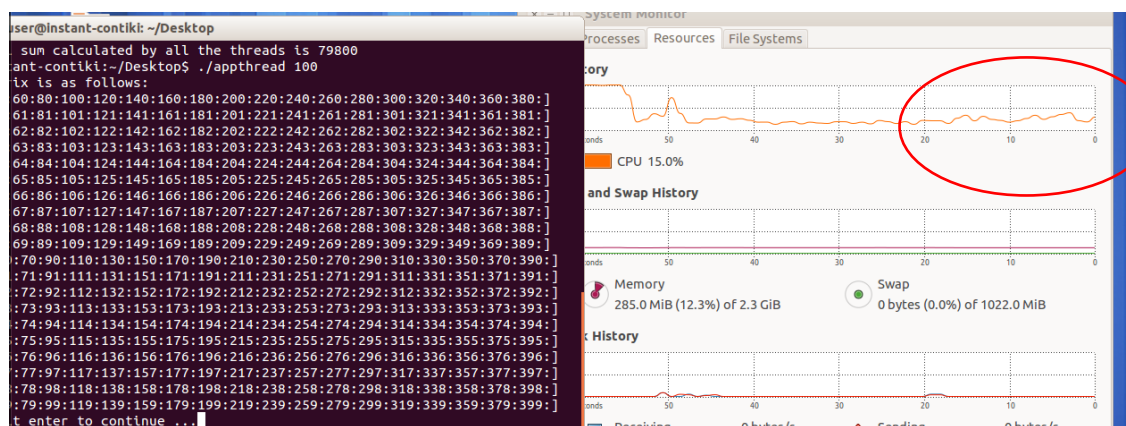
```
user@instant-contiki:~/Desktop$ ./appthread 3
Your matrix is as follows:
[0:20:40:60:80:100:120:140:160:180:200:220:240:260:280:300:320:340:360:380:]
[1:21:41:61:81:101:121:141:161:181:201:221:241:261:281:301:321:341:361:381:]
[2:22:42:62:82:102:122:142:162:182:202:222:242:262:282:302:322:342:362:382:]
[3:23:43:63:83:103:123:143:163:183:203:223:243:263:283:303:323:343:363:383:]
[4:24:44:64:84:104:124:144:164:184:204:224:244:264:284:304:324:344:364:384:]
[5:25:45:65:85:105:125:145:165:185:205:225:245:265:285:305:325:345:365:385:]
[6:26:46:66:86:106:126:146:166:186:206:226:246:266:286:306:326:346:366:386:]
[7:27:47:67:87:107:127:147:167:187:207:227:247:267:287:307:327:347:367:387:]
[8:28:48:68:88:108:128:148:168:188:208:228:248:268:288:308:328:348:368:388:]
[9:29:49:69:89:109:129:149:169:189:209:229:249:269:289:309:329:349:369:389:]
[10:30:50:70:90:110:130:150:170:190:210:230:250:270:290:310:330:350:370:390:]
[11:31:51:71:91:111:131:151:171:191:211:231:251:271:291:311:331:351:371:391:]
[12:32:52:72:92:112:132:152:172:192:212:232:252:272:292:312:332:352:372:392:]
[13:33:53:73:93:113:133:153:173:193:213:233:253:273:293:313:333:353:373:393:]
[14:34:54:74:94:114:134:154:174:194:214:234:254:274:294:314:334:354:374:394:]
[15:35:55:75:95:115:135:155:175:195:215:235:255:275:295:315:335:355:375:395:]
[16:36:56:76:96:116:136:156:176:196:216:236:256:276:296:316:336:356:376:396:]
[17:37:57:77:97:117:137:157:177:197:217:237:257:277:297:317:337:357:377:397:]
[18:38:58:78:98:118:138:158:178:198:218:238:258:278:298:318:338:358:378:398:]
[19:39:59:79:99:119:139:159:179:199:219:239:259:279:299:319:339:359:379:399:]
Please hit enter to continue ...
Thread 0 just terminated with its partial sum 23100
Thread 1 just terminated with its partial sum 23820
Thread 2 just terminated with its partial sum 32880
The total sum calculated by all the threads is 79800
user@instant-contiki:~/Desktop$
```


On constate que même si on augmente le nombre the threads le programme s'exécute presque immédiatement, mais lorsqu'on dépasse 256 threads le code ne s'exécute plus

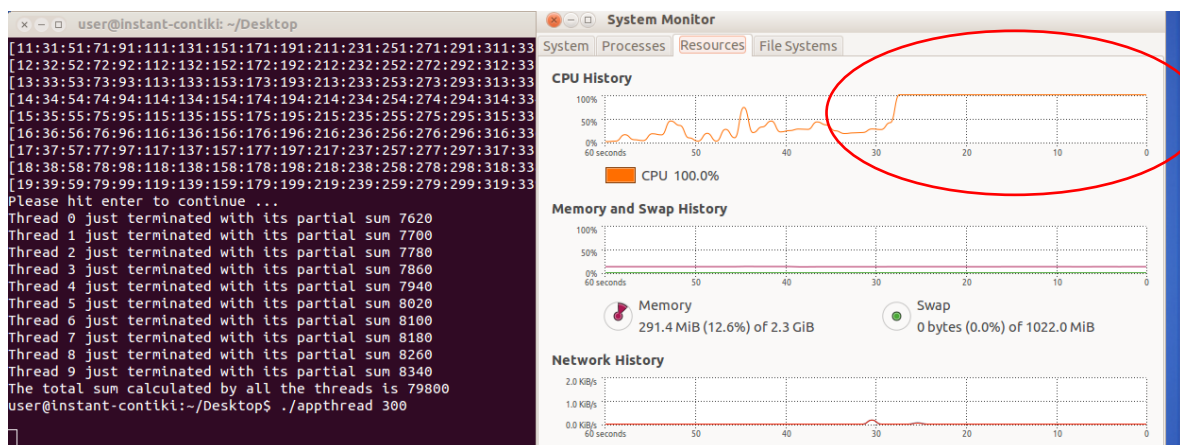
```
Please hit enter to continue ...
Thread 0 just terminated with its partial sum 7620
Thread 1 just terminated with its partial sum 7700
Thread 2 just terminated with its partial sum 7780
Thread 3 just terminated with its partial sum 7860
Thread 4 just terminated with its partial sum 7940
Thread 5 just terminated with its partial sum 8020
Thread 6 just terminated with its partial sum 8100
Thread 7 just terminated with its partial sum 8180
Thread 8 just terminated with its partial sum 8260
Thread 9 just terminated with its partial sum 8340
The total sum calculated by all the threads is 79800
user@instant-contiki:~/Desktop$
```

CPU Usage :

Si le nombre choisit est inférieur a 256 l'usage du CPU reste intermédiaire.



Lorsqu'on dépasse 256 threads l'usage du CPU est 100%.



VI. Conclusion :

À travers ce TP nous étions capable de voir et réaliser des programmes parallèles à l'aide de l'API Posix qui facilite la programmation concurrentielle/multitâches.

Ce TP nous a beaucoup aide à comprendre les différents mécanismes de Posix et ses commandes sur Linux.

Ainsi que sa riche bibliothèque PTHREADS qui nous fournit un ensemble de fonctions intéressantes pour la programmation temps réel.

Lien Github : https://github.com/ayy-oub/POSIX_TP