Systèmes d'exploitation **GLO-2001**

H16

RAPPORT – Travail Pratique Nº 2
Travaux avancés avec les threads Linux.

fait par: Alexandre Picard-Lemieux nom 111 103 625 matricule

Gaël Dostie nom 111096568 matricule

Résultat: _____ / 100

(Note : ce rapport est écrit de façon à vous faciliter la vie. En cas d'omission ou de différence entre ce rapport vierge et l'énoncé du TP, l'énoncé a priorité).

1. Niveaux de priorités des threads dans Linux (/15 pts)

1.1 Programmation de threads avec niveaux de priorités (/**18 pts**)

a)
main(): Création du thread 0
main(): Création du thread 1
main(): Création du thread 2
setpriority():0
setpriority():0
errno: 0
main(): Création du thread 3
errno: 0
main(): Création du thread 4
setpriority():0
errno: 0
setpriority():0
errno: 0
setpriority():0
errno: 0

b)
main(): Création du thread 0
main(): Création du thread 1
setpriority():0
setpriority():0
errno: 0
main(): Création du thread 2
errno: 0
main(): Création du thread 3
setpriority():0
errno: 0

main(): Création du thread 4
setpriority():0
errno: 0
setpriority():0
errno: 0

c)
main(): Création du thread 0
main(): Création du thread 1
main(): Création du thread 2
setpriority():0
main(): Création du thread 3
errno: 0
setpriority():0
setpriority():0
main(): Création du thread 4
errno: 0
errno: 0
setpriority():0
errno: 0
setpriority():0
errno: 0

d)
main(): Création du thread 0
main(): Création du thread 1
main(): Création du thread 2
setpriority():-1
main(): Création du thread 3
setpriority():-1
setpriority():0
errno: 0
errno: 13
errno: 13
main(): Création du thread 4
setpriority():0
errno: 0
setpriority():0
errno: 0

e)

main(): Création du thread 0
main(): Création du thread 1
setpriority():0
setpriority():0
main(): Création du thread 2
errno: 0
errno: 0
main(): Création du thread 3

setpriority():0
errno: 0
main(): Création du thread 4
setpriority():0
errno: 0
setpriority():0
errno: 0

1.2 Observation du temps d'exécution des threads avec différents niveaux de priorités (/**12 pts**)

a)

Thread	Niveau de priorité PR	PID	% CPU utilisé
0	20	1527	19.9
1	20	1528	19.9
2	20	1529	19.9
3	20	1530	19.9
4	20	1531	19.9

b)

Thread	Niveau de priorité PR	PID	% CPU utilisé
0	20	1864	29.6
1	21	1865	23.6
2	22	1866	18.9
3	23	1867	15.3
4	24	1868	12.3

c)

Thread	Niveau de priorité PR	PID	% CPU utilisé	Pourcentage théorique CFS
0	20	1874	39.9	40.2
1	22	1875	25.9	25.7
2	24	1876	16.3	16.6
3	26	1877	10.6	10.7
4	28	1878	6.6	6.8

Détail des calculs pour CFS :

Priorité	Poids (w)	Fraction
120	1024	1024 / 2546 = 0.402
122	655	655 / 2546 = 0.257

124	423	423 / 2546 = 0.166
126	272	272 / 2546 = 0.107
128	172	171 / 2546 = 0.068
Total	2546	1

d)

Thread	Niveau de priorité PR	PID	% CPU utilisé
0	20	1880	24.6
1	20	1881	24.6
2	20	1882	24.6
3	22	1883	15.9
4	24	1884	10.3

e)

Thread	Niveau de priorité PR	PID	% CPU utilisé
0	16	1887	40.2
1	18	1888	25.6
2	20	1889	16.3
3	22	1890	10.3
4	24	1891	6.6

2. Variables de conditions et mutex pour l'implémentation d'un thread pool (25 pts)

```
#include "ThreadPool.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

/* Thunk : In computer programming, a thunk is a subroutine that is created, often automatically,
   to assist a call to another subroutine. Thunks are primarily used to represent an additional
   calculation that a subroutine needs to execute, or to call a routine that does not support the
   usual calling mechanism. http://en.wikipedia.org/wiki/Thunk */
typedef struct {
      ThreadPool *pThreadPool; // pointeur sur l'objet ThreadPool
      int ThreadNum; // Numéro du thread, de 0 à n
} threadArg;

void *Thunk(void *arg) {
      threadArg *pThreadArg = (threadArg *)arg;
      ThreadPool *pThreadPool;
      pThreadPool = static_cast<ThreadPool*>(pThreadArg->pThreadPool);
```

```
    pThreadPool->MyThreadRoutine(pThreadArg->ThreadNum);
}


/* void ThreadPool(unsigned int nThread)
 Ce constructeur doit initialiser le thread pool. En particulier, il doit initialiser les variables
 de conditions et mutex, et démarrer tous les threads dans ce pool, au nombre spécifié par nThread.
 IMPORTANT! Vous devez initialiser les variables de conditions et le mutex AVANT de créer les threads
 qui les utilisent. Sinon vous aurez des bugs difficiles à comprendre comme des threads qui ne débloque
 jamais de phtread_cond_wait(). */
ThreadPool::ThreadPool(unsigned int nThread) {
    // Cette fonction n'est pas complète! Il vous faut la terminer!
    // Initialisation des membres
    PoolDoitTerminer = false;
    nThreadActive = nThread;
    bufferValide = true;
    buffer = 0;

    // Initialisation du mutex et des variables de conditions.
    pthread_mutex_init(&mutex, NULL);
    pthread_cond_init(&CondThreadRienAFaire,0);
    pthread_cond_init(&CondProducteur,0);

    // Création des threads. Je vous le donne gratuit, car c'est un peu plus compliqué que vu en classe.
    pTableauThread      = new pthread_t[nThread];
    threadArg *pThreadArg = new threadArg[nThread];
    int i;
    for (i=0; i < nThread; i++) {
        pThreadArg[i].ThreadNum = i;
        pThreadArg[i].pThreadPool = this;
    printf("ThreadPool(): en train de creer thread %d\n",i);
        int status = pthread_create(&pTableauThread[i], NULL, Thunk, (void *)&pThreadArg[i]);
        if (status != 0) {
            printf("oops, pthread a retourne le code d'erreur %d\n",status);
            exit(-1);
    }
   }
}

/* Destructeur ThreadPool::~ThreadPool()
   Ce destructeur doit détruire les mutex et variables de conditions. */
ThreadPool::~ThreadPool() {
    // À compléter
    pthread_mutex_destroy(&mutex);
    pthread_cond_destroy(&CondProducteur);
    pthread_cond_destroy(&CondThreadRienAFaire);
    delete [] pTableauThread;
}

/* void ThreadPool::MyThreadRoutine(int myID)
   Cette méthode est celle qui tourne pour chacun des threads crées dans le constructeur, et qui est
   appelée par la fontion thunk. Cette méthode est donc effectivement le code du thread consommateur,
```

```
  qui ne doit quitter qu'après un appel à la méthode Quitter(). Si le buffer est vide, MyThreadRoutine
  doit s'arrêter (en utilisant une variable de condition). Le travail à accomplir est un sleep() d'une
  durée spécifiée dans le buffer.
  */
void ThreadPool::MyThreadRoutine(int myID) {
    // À compléter
    printf("Thread %d commence!\n", myID);

    while (!PoolDoitTerminer) {
        pthread_mutex_lock(&mutex);

        if (!bufferValide) {
            pthread_cond_wait(&CondThreadRienAFaire,&mutex);
        }

        if(!PoolDoitTerminer) {
            printf("Thread %d récupère l'item %d!\n",myID,buffer);

            int currentBuffer = buffer;
            bufferValide = false;

            pthread_cond_signal(&CondProducteur);
            pthread_cond_signal(&CondThreadRienAFaire);

            pthread_mutex_unlock(&mutex);

            printf("Thread %d va dormir %d sec.\n",myID,currentBuffer);

            sleep(currentBuffer);
        }
    }

    printf("############ Thread %d termine!################\n",myID);

}

/* void ThreadPool::Inserer(unsigned int newItem)
  Cette méthode est appelée par le thread producteur pour mettre une tâche à exécuter dans le buffer
  (soit le temps à dormir pour un thread). Si le buffer est marqué comme plein, il faudra dormir
  sur une variable de condition. */
void ThreadPool::Inserer(unsigned int newItem) {
    // À compléter
    pthread_mutex_lock(&mutex);

    if (bufferValide) {
        pthread_cond_wait(&CondProducteur,&mutex);
    }

    bufferValide = true;
    buffer = newItem;
    pthread_cond_signal(&CondThreadRienAFaire);
    pthread_mutex_unlock(&mutex);
```

```
}

/* void ThreadPool::Quitter()
   Cette fonction est appelée uniquement par le producteur, pour indiquer au thread pool qu'il n'y
   aura plus de nouveaux items qui seront produits. Il faudra alors que tous les threads terminent
   de manière gracieuse. Cette fonction doit bloquer jusqu'à ce que tous ces threads MyThreadRoutine
   terminent, incluant ceux qui étaient bloqués sur une variable de condition. */
void ThreadPool::Quitter() {
      // À compléter
      pthread_mutex_lock(&mutex);

      while (bufferValide) {
            pthread_cond_wait(&CondThreadRienAFaire,&mutex);
      }

      PoolDoitTerminer = true;
      pthread_cond_broadcast(&CondThreadRienAFaire);
      pthread_mutex_unlock(&mutex);


      for (int i = 0; i < nThreadActive; i++) {
            pthread_join(pTableauThread[i],NULL);
      }
}
```

Sortie d'écran

./threadpool 3 10
Programme de test avec 3 threads et 10 items.
ThreadPool(): en train de creer thread 0
ThreadPool(): en train de creer thread 1
Thread 0 commence!
Thread 0 récupère l'item 0!
Thread 0 va dormir 0 sec.
Thread 1 commence!
ThreadPool(): en train de creer thread 2
(0.527) main: Je produis item numero 0 avec valeur 1.
main: item inséré.
(0.527) main: Je produis item numero 1 avec valeur 2.
Thread 2 commence!
Thread 0 récupère l'item 1!
Thread 0 va dormir 1 sec.
Thread 1 récupère l'item 1!
Thread 1 va dormir 1 sec.
main: item inséré.
(0.527) main: Je produis item numero 2 avec valeur 3.
Thread 2 récupère l'item 2!
Thread 2 va dormir 2 sec.
main: item inséré.

(0.528) main: Je produis item numero 3 avec valeur 4.
Thread 0 récupère l'item 3!
Thread 0 va dormir 3 sec.
Thread 1 récupère l'item 4!
Thread 1 va dormir 4 sec.
main: item inséré.
(1.528) main: Je produis item numero 4 avec valeur 1.
main: item inséré.
(1.528) main: Je produis item numero 5 avec valeur 2.
Thread 2 récupère l'item 1!
Thread 2 va dormir 1 sec.
main: item inséré.
(2.528) main: Je produis item numero 6 avec valeur 3.
Thread 2 récupère l'item 2!
Thread 2 va dormir 2 sec.
main: item inséré.
(3.528) main: Je produis item numero 7 avec valeur 4.
Thread 0 récupère l'item 3!
Thread 0 va dormir 3 sec.
main: item inséré.
(4.528) main: Je produis item numero 8 avec valeur 1.
Thread 1 récupère l'item 4!
Thread 1 va dormir 4 sec.
main: item inséré.
(5.528) main: Je produis item numero 9 avec valeur 2.
Thread 2 récupère l'item 1!
Thread 2 va dormir 1 sec.
main: item inséré.
(5.528) main: Destruction du thread pool.
Thread 2 récupère l'item 2!
Thread 2 va dormir 2 sec.
############ Thread 0 termine!################
############ Thread 2 termine!################
############ Thread 1 termine!################
(9.528) main: FIN!

4. Implémentation partielle d'une libraire de thread utilisateur (/60 pts)

Le listing du code source

```
/*****************************************************************************
   Travail pratique No 2 : Thread utilisateurs

   Ce fichier est votre implémentation de la librarie des threads utilisateurs.
```

```
    Systemes d'explotation GLO-2001
    Universite Laval, Quebec, Qc, Canada.
    (c) 2016 Philippe Giguere
 ***************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <ucontext.h>
#include <time.h>
#include <signal.h>
#include <sys/time.h>
#include "ThreadUtilisateur.h"

/* Définitions privées, donc pas dans le .h, car l'utilisateur n'a pas besoin de
   savoir ces détails d'implémentation. OBLIGATOIRE. */
typedef enum {
    THREAD_EXECUTE=0,
    THREAD_PRET,
    THREAD_BLOQUE,
    THREAD_TERMINE
} EtatThread;

#define TAILLE_PILE 8192   // Taille de la pile utilisée pour les threads

/* Structure de données pour créer une liste chaînée simple sur les threads qui ont fait un join.
   Facultatif */
typedef struct WaitList {
    struct TCB *pThreadWaiting;
    struct WaitList *pNext;
} WaitList;

/* TCB : Thread Control Block. Cette structure de données est utilisée pour stocker l'information
   pour un thread. Elle permet aussi d'implémenter une liste doublement chaînée de TCB, ce qui
   facilite la gestion et permet de faire un ordonnanceur tourniquet sans grand effort.  */
typedef struct TCB {  // Important d'avoir le nom TCB ici, sinon le compilateur se plaint.
    tid             id;       // Numero du thread
    EtatThread          etat;     // Etat du thread
    ucontext_t      ctx;      // Endroit où stocker le contexte du thread
    time_t          WakeupTime; // Instant quand réveiller le thread, s'il dort, en epoch time.
    struct TCB      *pSuivant;   // Liste doublement chaînée, pour faire un buffer circulaire
    struct TCB      *pPrecedant; // Liste doublement chaînée, pour faire un buffer circulaire
    struct WaitList     *pWaitListJoinedThreads; // Liste chaînée simple des threads en attente.
} TCB;

// Pour que les variables soient absolument cachées à l'utilisateur, on va les déclarer static
static TCB *gpThreadCourant = NULL;  // Thread en cours d'execution
static TCB *gpNextToExecuteInCircularBuffer = NULL;
static int gNumberOfThreadInCircularBuffer = 0;
static int gNextThreadIDToAllocate = 0;
static WaitList *gpWaitTimerList = NULL;
static TCB *gThreadTable[MAX_THREADS]; // Utilisé par la fonction ThreadID()
```

```
static char gEtatTable[4] = {'E', 'P', 'B', 'T'};

/* Cette fonction ne fait rien d'autre que de spinner un tour et céder sa place. C'est l'équivalent
   pour un système de se tourner les pouces. */
void IdleThreadFunction(void *arg) {
      struct timespec SleepTime, TimeRemaining;
      SleepTime.tv_sec = 0;
      SleepTime.tv_nsec = 250000000;
      while (1) {
            printf("              #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######\n");
            /* On va dormir un peu, pour ne pas surcharger inutilement le processus/l'affichage. Dans un
               vrai système d'exploitation, s'il n'y a pas d'autres threads d'actifs, ce thread demanderait au
               CPU de faire une pause, car il n'y a rien à faire. */
          nanosleep(&SleepTime,&TimeRemaining); // nanosleep interfere moins avec les alarmes.
            ThreadCeder();
      }
}

void retirerDuBufferCirculaire(TCB *pThread) {
      // Mettre le suivant en tête
      pThread->pPrecedant->pSuivant = pThread->pSuivant;
      pThread->pSuivant->pPrecedant = pThread->pPrecedant;

      // Enlever du cercle le thread courant
      pThread->pSuivant = NULL;
      pThread->pPrecedant = NULL;

      gNumberOfThreadInCircularBuffer--;
}

void ajouterAuBufferCirculaire(TCB *pThread) {
      if (pThread->id == (tid)0) {
            pThread->pPrecedant = NULL;
            pThread->pSuivant = NULL;
      }
      else {
            pThread->pSuivant = gpThreadCourant->pSuivant;
            pThread->pSuivant->pPrecedant = pThread;
            pThread->pPrecedant = gpThreadCourant;
            pThread->pPrecedant->pSuivant = pThread;
      }
      gNumberOfThreadInCircularBuffer++;
      pThread->etat = THREAD_PRET;
      gpNextToExecuteInCircularBuffer = pThread;
}

/* ************************************************************************************************
                        T h r e a d I n i t
   ************************************************************************************************/
int ThreadInit(void){
      printf("\n  ********************************** ThreadInit()  ********************************** \n");
      // THREAD IDLE
```

```
    ThreadCreer(*IdleThreadFunction, 0);

    // THREAD MAIN
    TCB *tcb_main = (TCB *) malloc(sizeof(TCB));
    tcb_main->id = gNextThreadIDToAllocate;
    gNextThreadIDToAllocate++;
    gThreadTable[tcb_main->id] = tcb_main;
    tcb_main->pSuivant = gThreadTable[0];
    tcb_main->pSuivant->pPrecedant = tcb_main;
    tcb_main->pPrecedant = gThreadTable[0];
    tcb_main->pPrecedant->pSuivant = tcb_main;
    gpThreadCourant = tcb_main;
    return 1;
}


/* ******************************************************************************************
                    T h r e a d C r e e r
 ******************************************************************************************/
tid ThreadCreer(void (*pFuncThread)(void *), void *arg) {
    printf("\n                    ********************************                    ThreadCreer(%p,%p)
******************************** \n",pFuncThread,arg);
    // CREATION DU THREAD
    TCB *tcb_nouveau = (TCB *) malloc(sizeof(TCB));
    tcb_nouveau->id = gNextThreadIDToAllocate;
    gNextThreadIDToAllocate++;
    gThreadTable[tcb_nouveau->id] = tcb_nouveau;
    tcb_nouveau->pWaitListJoinedThreads = NULL;
    getcontext(&tcb_nouveau->ctx);

    char *pile = malloc(TAILLE_PILE);
    if (pile == NULL) {
        return -1;
    }
    tcb_nouveau->ctx.uc_stack.ss_sp = pile;
    tcb_nouveau->ctx.uc_stack.ss_size = TAILLE_PILE;
    makecontext(&tcb_nouveau->ctx, (void *) pFuncThread, 1, arg);

    ajouterAuBufferCirculaire(tcb_nouveau);

    gpNextToExecuteInCircularBuffer = tcb_nouveau;
    return tcb_nouveau->id;
}

/* ******************************************************************************************
                    T h r e a d C e d e r
 ******************************************************************************************/
void ThreadCeder(void) {
    printf("\n    ********************************  ThreadCeder()    ********************************
\n");
    printf("----- Etat de l'ordonnanceur avec %d threads -----\n", gNumberOfThreadInCircularBuffer);
    struct TCB *i_tcb = gpNextToExecuteInCircularBuffer;
```

```
    struct WaitList *i_waitList;

    do {
        char debut[40] = "\t";
        char special[40] = "";
        if (i_tcb == gpNextToExecuteInCircularBuffer) strncpy(debut, "prochain->", sizeof(debut));
        if (i_tcb->id == (tid)0) strncpy(special, "*Special Idle Thread*", sizeof(special));
        printf("| %s\tThreadID:%d\t État:%c %s\t WaitList", debut, i_tcb->id, gEtatTable[i_tcb->etat], special);
        struct WaitList *i_threadWaitList;
        for (i_threadWaitList = i_tcb->pWaitListJoinedThreads; i_threadWaitList != NULL; i_threadWaitList =
i_threadWaitList->pNext) {
            printf("-->(%d)", i_threadWaitList->pThreadWaiting->id);
        }
        printf("\n");
        i_tcb = i_tcb->pSuivant;
    } while(i_tcb != gpNextToExecuteInCircularBuffer);

    printf("----- Liste des threads qui dorment, epoch time=%d -----\n", (int)time(NULL));

    for (i_waitList = gpWaitTimerList; i_waitList != NULL; i_waitList = i_waitList->pNext) {
        printf("| \t\tThreadID:%d\t État:%c\t WakeTime=%d\t WaitList", i_waitList->pThreadWaiting->id,
gEtatTable[i_waitList->pThreadWaiting->etat], (int)i_waitList->pThreadWaiting->WakeupTime);
        struct WaitList *i_threadWaitList;
        for (i_threadWaitList = i_waitList->pThreadWaiting->pWaitListJoinedThreads; i_threadWaitList != NULL;
i_threadWaitList = i_threadWaitList->pNext) {
            printf("-->(%d)", i_threadWaitList->pThreadWaiting->id);
        }
        printf("\n");
    }
    printf("-------------------------------------------------\n");

    WaitList *precedant = NULL;
    WaitList *courant = gpWaitTimerList;
    while(courant) {
        if (courant->pThreadWaiting->WakeupTime < time(NULL)) {
            // Retirer de cette liste
            if(precedant) {
                precedant->pNext = courant->pNext;
            } else {
                gpWaitTimerList = courant->pNext;
            }
            ajouterAuBufferCirculaire(courant->pThreadWaiting);
        }
        precedant = courant;
        courant = courant->pNext;
    }
    // Faire du garbage collection
    while(gpNextToExecuteInCircularBuffer->etat == THREAD_TERMINE) {
        TCB * pThread = gpNextToExecuteInCircularBuffer;
        printf("ThreadCeder: Garbage collection sur le thread %d\n", pThread->id);
        // On passe au prochain thread à exécuter
        gpNextToExecuteInCircularBuffer = pThread->pSuivant;
```

```
        retirerDuBufferCirculaire(pThread);
        // Désallouer sa pile
        free(pThread->ctx.uc_stack.ss_sp);
        free(pThread->pWaitListJoinedThreads);
        // Retirer du ThreadTable
        gThreadTable[pThread->id] = NULL;
        // Desallouer le TCB
        free(pThread);
    }

    if (gNumberOfThreadInCircularBuffer == 1 && gpWaitTimerList == NULL) {
        ajouterAuBufferCirculaire(gThreadTable[1]);
    }

    TCB *oldCourrant = gpThreadCourant;
    if(oldCourrant->etat == THREAD_EXECUTE) {
        oldCourrant->etat = THREAD_PRET;
    }
    while (gpNextToExecuteInCircularBuffer->etat != THREAD_PRET) {
        gpNextToExecuteInCircularBuffer = gpNextToExecuteInCircularBuffer->pSuivant;
    }
    gpThreadCourant = gpNextToExecuteInCircularBuffer;
    gpThreadCourant->etat = THREAD_EXECUTE;
    gpNextToExecuteInCircularBuffer = gpThreadCourant->pSuivant;
    swapcontext(&oldCourrant->ctx, &gpThreadCourant->ctx);
}


/* ************************************************************************************************
                        T h r e a d J o i n d r e
  ************************************************************************************************/
int ThreadJoindre(tid ThreadID){
    printf("\n                                ********************************                ThreadJoindre(%d)
******************************** \n",ThreadID);
    // Obtenir le thread par son tid
    TCB *threadAJoindre = gThreadTable[ThreadID];
    if (threadAJoindre == NULL || threadAJoindre->etat == THREAD_TERMINE) {
        return -1;
    }
    // Met le thread comme bloquer
    gpThreadCourant->etat = THREAD_BLOQUE;
    retirerDuBufferCirculaire(gpThreadCourant);

    // Obtient l'espace nécessaire pour waitList
    WaitList *waitList = (struct WaitList *) malloc(sizeof(struct WaitList));
    waitList->pThreadWaiting = gpThreadCourant;
    waitList->pNext = threadAJoindre->pWaitListJoinedThreads;
    threadAJoindre->pWaitListJoinedThreads = waitList;

    // On continue dans l'ordonnanceur
    ThreadCeder();
```

```
    // On retourne 1, car tout a fonctionné
    return 1;
}

/* ************************************************************************************************
                        T h r e a d Q u i t t e r
  ************************************************************************************************/
void ThreadQuitter(void){
    printf("                                ********************************                ThreadQuitter(%d)
********************************* \n",gpThreadCourant->id);
    // Mettre le thread comme étant terminer
    gpThreadCourant->etat = THREAD_TERMINE;

    // Réveille les threads en attente
    WaitList *waitList = gpThreadCourant->pWaitListJoinedThreads;
    for (waitList; waitList != NULL; waitList = waitList->pNext) {
        printf("ThreadQuitter: je réveille le thread %d\n", waitList->pThreadWaiting->id);
        ajouterAuBufferCirculaire(waitList->pThreadWaiting);
    }
    free(waitList);
    gpThreadCourant->pWaitListJoinedThreads = NULL;

    ThreadCeder();
    printf(" ThreadQuitter:Je ne devrais jamais m'exectuer! Si je m'exécute, vous avez un bug!\n");
    return;
}

/* ************************************************************************************************
                        T h r e a d I d
  ************************************************************************************************/
tid ThreadId(void) {
    // Libre à vous de la modifier. Mais c'est ce que j'ai fait dans mon code, en toute simplicité.
    return gpThreadCourant->id;
}

/* ************************************************************************************************
                        T h r e a d D o r m i r
  ************************************************************************************************/
void ThreadDormir(int secondes) {
    printf("\n  ******************************** ThreadDormir(%d)  ********************************
\n",secondes);
    // Mettre le thread comme bloquer
    gpThreadCourant->etat = THREAD_BLOQUE;
    // Ajouter le nombre de seconde à dormir
    gpThreadCourant->WakeupTime = time(NULL) + secondes;
    // Retirer du buffer circulaire
    retirerDuBufferCirculaire(gpThreadCourant);
    // Allouer l'espace pour la nouvelle waitList.
    WaitList *waitList = (struct WaitList *)malloc(sizeof(struct WaitList));
    // Mettre le thread courant à dormir
    waitList->pThreadWaiting = gpThreadCourant;
    // Mettre le suivant comme étant le premier de l'autre liste.
```

```
    waitList->pNext = gpWaitTimerList;
    // Ajouter ensuite l'élément dans la liste du  WaitTimerList.
    gpWaitTimerList = waitList;
    // Céder la place à un autre thread.
    ThreadCeder();
    return;
}
```

Indiquez si certaines fonctions n'ont pas été implémentés ou sont susceptibles de planter
Aucune méthode n'a été implémentée . La méthode ThreadCeder est la méthode la plus susceptibles de planter.

Sortie d'écran de l'exécution du code `TestThread.c`.

```
 ********************************* ThreadInit()  *********************************

  ********************************* ThreadCreer(0x400d4d,(nil)) *********************************
(0.296) Main: Le thread ID du main est 1.

                          *********************************          ThreadCreer(0x4009dc,0x7ffc8fbdbc90)
*********************************
(0.296) Main: Le thread avec ID 2 a été créé.

                          *********************************          ThreadCreer(0x4009dc,0x7ffc8fbdbca8)
*********************************
(0.296) Main: Le thread avec ID 3 a été créé.

                          *********************************          ThreadCreer(0x4009dc,0x7ffc8fbdbcc0)
*********************************
(0.296) Main: Le thread avec ID 4 a été créé.
(0.296) Main: Je joins le thread ID 2

  ********************************* ThreadJoindre(2)  *********************************

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:4  État:P   WaitList
|          ThreadID:3  État:P   WaitList
|          ThreadID:2  État:P   WaitList-->(1)
|          ThreadID:0  État:P *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114500 -----
-------------------------------------------------
(0.296) Thread4: Je tourne avec une variable sur la pile à 0x0x25b7368.

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:3  État:P   WaitList
|          ThreadID:2  État:P   WaitList-->(1)
|          ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:4  État:E   WaitList
----- Liste des threads qui dorment, epoch time=1457114500 -----
-------------------------------------------------
```

(0.355) Thread3: Je tourne avec une variable sur la pile à 0x0x25b4f78.

********** ThreadCeder() **********
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:2 État:P WaitList-->(1)
| ThreadID:0 État:P *Special Idle Thread* WaitList
| ThreadID:4 État:P WaitList
| ThreadID:3 État:E WaitList
----- Liste des threads qui dorment, epoch time=1457114500 -----
---
(0.397) Thread2: Je tourne avec une variable sur la pile à 0x0x25b2b88.

********** ThreadCeder() **********
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:0 État:P *Special Idle Thread* WaitList
| ThreadID:4 État:P WaitList
| ThreadID:3 État:P WaitList
| ThreadID:2 État:E WaitList-->(1)
----- Liste des threads qui dorment, epoch time=1457114500 -----
---
######### Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

********** ThreadCeder() **********
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:4 État:P WaitList
| ThreadID:3 État:P WaitList
| ThreadID:2 État:P WaitList-->(1)
| ThreadID:0 État:E *Special Idle Thread* WaitList
----- Liste des threads qui dorment, epoch time=1457114500 -----
---
(0.678) Thread4: Je tourne avec une variable sur la pile à 0x0x25b7368.

********** ThreadCeder() **********
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:3 État:P WaitList
| ThreadID:2 État:P WaitList-->(1)
| ThreadID:0 État:P *Special Idle Thread* WaitList
| ThreadID:4 État:E WaitList
----- Liste des threads qui dorment, epoch time=1457114500 -----
---
(0.731) Thread3: Je tourne avec une variable sur la pile à 0x0x25b4f78.

********** ThreadCeder() **********
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:2 État:P WaitList-->(1)
| ThreadID:0 État:P *Special Idle Thread* WaitList
| ThreadID:4 État:P WaitList
| ThreadID:3 État:E WaitList
----- Liste des threads qui dorment, epoch time=1457114500 -----
---
(0.769) Thread2: Je tourne avec une variable sur la pile à 0x0x25b2b88.

```
 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:4  État:P   WaitList
|          ThreadID:3  État:P   WaitList
|          ThreadID:2  État:E   WaitList-->(1)
----- Liste des threads qui dorment, epoch time=1457114500 -----
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 3 threads -----
| prochain->ThreadID:4  État:P   WaitList
|          ThreadID:3  État:P   WaitList
|          ThreadID:2  État:P   WaitList-->(1)
|          ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114501 -----
-------------------------------------------------
(1.051) Thread4: Je tourne avec une variable sur la pile à 0x0x25b7368.
(1.113) Thread4: Je vais dormir pendant 3 secondes!

 ********************************* ThreadDormir(3)  *********************************

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 2 threads -----
| prochain->ThreadID:3  État:P   WaitList
|          ThreadID:2  État:P   WaitList-->(1)
|          ThreadID:0  État:P *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114501 -----
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
(1.113) Thread3: Je tourne avec une variable sur la pile à 0x0x25b4f78.
(1.149) Thread3: Je vais dormir pendant 2 secondes!

 ********************************* ThreadDormir(2)  *********************************

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:2  État:P   WaitList-->(1)
|          ThreadID:0  État:P *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114501 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
(1.149) Thread2: Je tourne avec une variable sur la pile à 0x0x25b2b88.

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:2  État:E   WaitList-->(1)
----- Liste des threads qui dorment, epoch time=1457114501 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList
```

```
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:2  État:P   WaitList-->(1)
|          ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114501 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
(1.973) Thread2: Je QUITTE!
  ********************************* ThreadQuitter(2)  *********************************
ThreadQuitter: je réveille le thread 1

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 2 threads -----
| prochain->ThreadID:1  État:P   WaitList
|          ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:2  État:T   WaitList
----- Liste des threads qui dorment, epoch time=1457114501 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
(1.973) Main: Le thread ID 2 a terminé!
(1.973) Main: Je joins le thread ID 3

  ********************************* ThreadJoindre(3)  *********************************

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:2  État:T   WaitList
----- Liste des threads qui dorment, epoch time=1457114501 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:2  État:T   WaitList
|          ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114502 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
ThreadCeder: Garbage collection sur le thread 2
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
```

```
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114502 -----
|           ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|           ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114502 -----
|           ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|           ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114502 -----
|           ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|           ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114503 -----
|           ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|           ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114503 -----
|           ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|           ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114503 -----
|           ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|           ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######
```

```
  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114503 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114504 -----
|          ThreadID:3  État:B   WakeTime=1457114503      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
(4.226) Thread3: Je tourne avec une variable sur la pile à 0x0x25b4f78.
(4.291) Thread3: Je vais dormir pendant 2 secondes!

  ******************************** ThreadDormir(2)  ********************************

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114504 -----
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114504 -----
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114504 -----
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
```

```
----- Liste des threads qui dorment, epoch time=1457114505 -----
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114504      WaitList
-------------------------------------------------
(5.042) Thread4: Je tourne avec une variable sur la pile à 0x0x25b7368.
(5.107) Thread4: Je vais dormir pendant 3 secondes!

  ******************************** ThreadDormir(3)  ********************************

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114505 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114505 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114505 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114505 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114506 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
```

```
          #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114506 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
          #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114506 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
          #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114506 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
          #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114507 -----
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
|          ThreadID:3  État:B   WakeTime=1457114506      WaitList-->(1)
-------------------------------------------------
(7.111) Thread3: Je tourne avec une variable sur la pile à 0x0x25b4f78.
(7.177) Thread3: Je vais dormir pendant 2 secondes!

  ******************************** ThreadDormir(2)  ********************************

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114507 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
-------------------------------------------------
          #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
```

```
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114507 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114507 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114507 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114508 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114508 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ******************************** ThreadCeder()  ********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114508 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######
```

```
  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114508 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114509 -----
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
|          ThreadID:4  État:B   WakeTime=1457114508      WaitList
-------------------------------------------------
(9.179) Thread4: Je tourne avec une variable sur la pile à 0x0x25b7368.
(9.240) Thread4: Je vais dormir pendant 3 secondes!

  ********************************* ThreadDormir(3)  *********************************

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114509 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114509 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114509 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
-------------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114509 -----
```

```
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
-----------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114510 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
|          ThreadID:3  État:B   WakeTime=1457114509      WaitList-->(1)
-----------------------------------------------
(10.242) Thread3: Je tourne avec une variable sur la pile à 0x0x25b4f78.

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:3  État:E   WaitList-->(1)
----- Liste des threads qui dorment, epoch time=1457114510 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
-----------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:3  État:P    WaitList-->(1)
|          ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114511 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
-----------------------------------------------
(11.158) Thread3: Je tourne avec une variable sur la pile à 0x0x25b4f78.

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:3  État:E   WaitList-->(1)
----- Liste des threads qui dorment, epoch time=1457114511 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
-----------------------------------------------
            #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:3  État:P    WaitList-->(1)
|          ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114512 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
-----------------------------------------------
(12.024) Thread3: Je QUITTE!
  ********************************* ThreadQuitter(3)  *********************************
ThreadQuitter: je réveille le thread 1
```

```
 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 2 threads -----
| prochain->ThreadID:1  État:P   WaitList
|          ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:3  État:T   WaitList
----- Liste des threads qui dorment, epoch time=1457114512 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList
-------------------------------------------------
(12.024) Main: Le thread ID 3 a terminé!
(12.024) Main: Je joins le thread ID 4

 ********************************* ThreadJoindre(4)  *********************************

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
|          ThreadID:3  État:T   WaitList
----- Liste des threads qui dorment, epoch time=1457114512 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList-->(1)
-------------------------------------------------
           #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 1 threads -----
| prochain->ThreadID:3  État:T   WaitList
|          ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114512 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList-->(1)
-------------------------------------------------
ThreadCeder: Garbage collection sur le thread 3
           #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114512 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList-->(1)
-------------------------------------------------
           #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114512 -----
|          ThreadID:4  État:B   WakeTime=1457114512      WaitList-->(1)
-------------------------------------------------
           #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

 ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 0 threads -----
| prochain->ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114513 -----
```

```
|           ThreadID:4  État:B   WakeTime=1457114512      WaitList-->(1)
-------------------------------------------------
(13.025) Main: Le thread ID 4 a terminé!
(13.025) Main: Tous les threads ont terminé!

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 2 threads -----
| prochain->ThreadID:4  État:P    WaitList-->(1)
|           ThreadID:0  État:P *Special Idle Thread*      WaitList
|           ThreadID:1  État:E   WaitList
----- Liste des threads qui dorment, epoch time=1457114513 -----
-------------------------------------------------
(13.025) Thread4: Je tourne avec une variable sur la pile à 0x0x25b7368.

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 2 threads -----
| prochain->ThreadID:0  État:P *Special Idle Thread*      WaitList
|           ThreadID:1  État:P    WaitList
|           ThreadID:4  État:E   WaitList-->(1)
----- Liste des threads qui dorment, epoch time=1457114513 -----
-------------------------------------------------
             #########  Idle Thread 0 s'exécute et va prendre une pose de 250 ms... #######

  ********************************* ThreadCeder()  *********************************
----- Etat de l'ordonnanceur avec 2 threads -----
| prochain->ThreadID:1  État:P    WaitList
|           ThreadID:4  État:P    WaitList-->(1)
|           ThreadID:0  État:E *Special Idle Thread*      WaitList
----- Liste des threads qui dorment, epoch time=1457114513 -----
-------------------------------------------------
(13.928) Main: je termine.
```