```c
/*
 * ######
 * ## sema1.c
 * ## ~~~~~~~
 * ## 04.02.1995: Creation TL
 * ## 07.05.2018:
 * ######
 */

#include <errno.h>
#include <stdio.h>
#include <stdlib.h>

#include <signal.h>

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <unistd.h>


/* Declaration des Types et des Variables */

  typedef void (*sighandler_t) (int);

  extern int errno;

  key_t      key = 3;
  int        semid, nsems, semnum;

  struct sembuf sops;

  union  semun {
      int                 val;
      struct semid_ds    *buf;
      unsigned short int *array;
      struct seminfo     *__buf;
  } arg;

/* Routine de Traitement de l'interruption SIGINT */
/* -> on arrete le programme */

void my_stop_int()
{
  printf( "\nsema1> my_stop_int : Arret du programme sans destuction du semaphore\n" );
  exit( 0 );
}

/* Routine de Traitement de l'interruption SIGUSR1 */
/* -> on remet a zero la valeur du semaphore */

void my_stop_usr1()
{
  int kr;
  semnum  = 0;
  arg.val = 0;

  kr = semctl( semid, semnum, SETVAL, arg );
  if ( kr == 1 ) {
    perror( "\nsema1> my_stop_usr1 : Pb. initialisation SEMAPHORE\n" );
  } else {
    printf( "\nsema1> my_stop_usr1 : Remise a Zero  SEMAPHORE( 3, 0 )\n" );
  }
}

/* Routine de Traitement de l'interruption SIGUSR2 */
/* -> on supprime le semaphore et arrete le programme */

void my_stop_usr2()
{
  int kr;
  semnum = 0;

  kr = semctl(semid, semnum, IPC_RMID, NULL);
```

```c
  if ( kr == -1 ) {
    perror( "\nsema1> my_stop_usr2 : Pb. destruction semaphore\n" ); exit( 2 );
  } else {
    printf( "\nsema1> my_stop_usr2 : Destruction SEMAPHORE(3)\n" );
  }
  exit( 0 );
}

/* Programme principal */

int main()
{
  int    kr, val;

  char   SS[ 1024 ];
  char * Ptr_SS = &SS[ 0 ];

/* Prise en compte des routines de traitement des interruptions SIGINT, SIGUSR1 et SIGUSR2 */

  /* tester sans SIGINT puis avec...*/

  /*
  signal( SIGINT , (sighandler_t) my_stop_int );
  */

  signal( SIGUSR1, (sighandler_t) my_stop_usr1 );
  signal( SIGUSR2, (sighandler_t) my_stop_usr2 );

/* Creation/attachement du groupe 3 constitue de 1 semaphore (numero zero) */

  printf( " sema1> Ouverture Ensemble de Semaphores ( key = 3 ) avec 1 seul semaphore\n");
  printf( " sema1> \n");

  key   = 3;
  nsems = 1;

  semid = semget( key, nsems, 0640 | IPC_CREAT );
  if ( semid == -1 ) {
    perror( " sema1> Creation SEMAPHORE impossible" ); exit( 1 );
  }

/* Affichage de la valeur associiee au semaphore zero */

  semnum = 0;

  val = semctl( semid, semnum, GETVAL, NULL );
  if ( val == -1 ) {
    perror( " sema1> Pb. recuperation valeur semaphore zero" );
  }

  printf( " sema1> Valeur du semaphore zero = %d\n", val );
  printf( " sema1> \n");

/* Message d'accueil  */

  printf( " sema1> Pour deverouiller la ressource, lancer sema2\n");
  printf( " sema1> Pour remettre a Zero le SEMAPHORE(3), faire kill -%d %d\n"
          , SIGUSR1, getpid() );
  printf( " sema1> Pour arreter le programme, faire ^C ou bien kill -%d %d\n"
          , SIGUSR2, getpid() );

/* precaution deverouillage automatique semop */

  sops.sem_flg = SEM_UNDO;

/* boucle */

  for (;;) {
    printf( " sema1> Entrez un commentaire ----> " ); scanf( "%s", Ptr_SS );

    sops.sem_num =  0;
    sops.sem_op  = -1; /* < 0 :  demande d'un droit */

    kr = semop( semid, &sops, 1 );
    if ( kr == -1 ) {
```

```c
      if ( errno == 4 ) {
        printf( " sema1> Deblocage par signal\n" );
      } else {
        printf( " sema1> Erreur decrementation semaphore\n" );
      }
    }

    printf( " sema1> Liberation\n" );
    printf( " sema1> Echo du commentaire   - - > %s\n", Ptr_SS );

    semnum = 0;

    val = semctl( semid, semnum, GETVAL, NULL );
    if ( val == -1 ) {
      perror( " sema1> Pb. recuperation valeur semaphore" );
    }

    printf( " sema1> Valeur du semaphore = %d\n", val );
  }
}
/* --- fin  sema1.c  --- */
```