מערכות הפעלה - תרגיל בית 2
רבקה סורשר 567153394

1. הגורם לבעית אי העקביות הינה שכמה תהליכונים מעדכנים יחד את החשבון המשותף אזי כשתהליכון אחד קורא מה היתרה עד כה והוא עוד לא מספיק לכתוב לחשבון את היתרה הנוכחית ע"פ הפעולה שהוא עשה בינתיים תהליכון אחר יכול לעדכן מה היתרה ע"פ הפעולה שלו וכך התהליכון הנ"ל יעדכן את החשבון לפי המידע הישן והלא רלונטי.
בקצרה: נוצר מצב של תנאי מירוץ בין התהליכונים בעדכון החשבון המשותף.
ע"מ לתקן זאת נגדיר את השורות של הקריאה של הקובץ וכתיבה לקובץ בפונקציית update בתור קטע קריטי וננעל אותם בסמפורים כך:

```
sem_wait(&mutex); //Entered critical code;
fscanf (fp, "%s", buf);
fclose( fp );
//if ( amount != 0 )  printf("%s", "."); //uncomment to see the progress,not needed
if (!( fp = fopen( account,"w" )))  return( ERROR );
fprintf( fp,"%d", n = atoi(buf) + amount );
fclose( fp );
sem_post(&mutex); //Exited critical code
```

<u>הקוד במלואו אחרי התיקון של סעיף 1:</u>
השורות המודגשות הינם ההוספות עבור התיקון.

```
/*****************************************************************************************
File: agents.c
Compile:  gcc agents.c -o agents -pthread
Or use make, there is also clean target in Makefile
*****************************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdatomic.h> //!!! Put attension on it !!!!
// Usage: atomic_int i=0; After that it can be used i++; synchronized by paraller threads.
#include "agents.h"
/****************************************/
int init (int);
int check_point(int*);
void go_agent (char *);
int log_sum (char *);
int update(int, char *, char * );
int write_to_log(int, char *);
void fatal_error(char *,char *);
int gen_rand(int);
/*************************************************************************/
/*          Global Array - All Agents' names              */
char agents[ MAX_AGENTS ][ NAMES_LEN ] = AGENTS;
int p ; /* number of the started agentes */
pthread_t thread[MAX_AGENTS]/*agents*/, cThread/*control thtead*/;
```

```
sem_t mutex;
/*********************************************************************/
/* Reset all files,Dispatch all agents and control actions */
int main(int ac,char **av )
{
 sem_init(&mutex, 0, 1);
 int i = 0, num_agents = 1;/* default */
 if ( ac == 2 )  i = atoi( av[1] );  /* Check arguments */
 else if (ac != 1) fatal_error("USAGE:agents [number_of_agents]\n", NULL);
 /* initiate an account file with a start sum */
 num_agents = init( i );
 /* Main process distributes work to threads (agents).
    Agents perform independent operations.Every agent updates the account file
    after each operation and logs them in a personal log file */
 for(i=0, p=0 ; i < num_agents; i++, p++ ){
 if(pthread_create(&thread[p], NULL, (void *)go_agent, &agents[p]))
   { perror("ERROR creating thread."); --p; }
   }
 i = p;
 printf("Finished creating %d agent(s). \n",num_agents);
 sleep(1);
 /* The father waits for agents to return succesfully */
 while ( p >= 1 ) { sleep(2);} //!!!Have to transfer to check_point
/* check the consistency(log files vs.the account file. */
 check_point( &i );
 sem_destroy(&mutex);
 return 0;
}

/*********************************************************************/
/*  initiate the account file with the starting sum
*  return the number of agents to be created in the range [1, MAX_AGENTS] */
int init(int num_agents )   /* Initiate auxiliary files,returns number agents */
{
  FILE *fp;
  int i;
  //srand( time( NULL ) ); // seeds rand()
  if ( num_agents < 1 || num_agents > MAX_AGENTS ){
    printf("%d agent(s) defaults to 1\n",num_agents);
    num_agents = 1;
   }
  printf("Creating %d agent(s). \n",num_agents);
  // delete old files; exit if fails
  for ( i = 0; i < num_agents; i++)
      if ((unlink( agents[i] ) < 0) && (errno != ENOENT)  )
         fatal_error("Deleting agents log", "");
  if (( fp = fopen( ACCOUNT,"w")) == NULL )
       fatal_error("Creating ACCOUNT file", "");
  fprintf( fp,"%d\n",START_SUM );
  fclose( fp );
  return( num_agents );
}
/*********************************************************************/
int check_point(int * num_agents ) /* Are agents' actions consistent ? */
```

```
{
int i,n, control_sum , sum ;
  control_sum=0;
  if ( ( n = rand() % MAX_SLEEP  ) != 0 ) sleep( n * 3 );
  printf("\n\n ----- Controler ----- \n");
  if ((sum = update(0,ACCOUNT, NULL )) == ERROR) return(ERROR);//get the sum by updating 0
  for ( i = 0; i < *num_agents; ++i )
       control_sum = control_sum + log_sum( agents[i] );
  control_sum = START_SUM + control_sum;
  if  ( sum != control_sum )
  {
     fprintf( stderr,"\n Not consistent! Somebody steals?!\n");
     fprintf( stderr,"Controler: %d, Account: %d \n",control_sum,sum );
                fprintf( stderr," Terminating all agents!\n");
                return(1);
  }
  printf("\nOperations O.K. Controler: %d, Acount: %d\n", control_sum, sum );
exit(SUCCESS);
}
/*************************************************************************/
int log_sum(char *log_file )    /* Returns the sum of log_file */
{
  FILE  *fp;
  int sum = 0,tmp_sum;
  if ( ( fp = fopen( log_file ,"r+") ) == NULL )
  {
     perror( log_file );
     printf("\nCan't Control %s agent!\n",log_file );
     return( 0 );
  }
  while ( fscanf( fp,"%d",&tmp_sum ) > 0 ) sum = sum + tmp_sum;
  fclose( fp );
  printf(" Sum of %s = %d\n",log_file,sum );
  return( sum );
}
/*****************************************************************/
void fatal_error(char *err_msg,char *perror_arg)  /* Prints Error messages */
{
 fprintf(stderr,"\nERROR: %s. ", err_msg);
 if ( perror_arg != NULL ) perror( perror_arg);
 exit(ERROR);
}
/*************************************************************************/
void go_agent(char *agent_name)      /* sends agent to work */
{
 int n, i, amount;
   for( i = 0; i < OPERATIONS ; ++i )
             {
                amount = gen_rand( MAX_OPER );
                if ( update( amount,ACCOUNT, agent_name ) == ERROR )
                {
                            fatal_error( "Can't Update Account File", ACCOUNT); //It includes process termination
                }
     else {
```

```
                if ( write_to_log( amount,agent_name ) == ERROR ){ // rollback
                  if (update(-amount, ACCOUNT, NULL ) == ERROR)
                    fprintf(stderr, "ERROR: Rollback failed. Agent %s\n", agent_name);
                fatal_error( "Can't log operations", agent_name); //It includes process termination
                 }
               }
              printf(" Op of %s = %d\n",agent_name,amount );
              if ( ( n = rand() % MAX_SLEEP ) != 0 )  sleep( n );
            }
            p--;
   printf(" Agent %s : Succefull!\n",agent_name );
  return;
}
/*****************************************************************/
int gen_rand(int range)/*Generates random numbers between -rang and +range*/
{
  int n,sign;
  static int s=17;
  srand( time( NULL ) % s++);
  while( ( n = rand() ) == 0 );
  sign = (n/1111) % 2;
  n = ( (n/1117) % range ) + 1;
  if ( sign == 0 )  return( n );
  else return( 0 - n );
}
/*********************************************************************/
/* Updates account using amount. Returns new sum */
int update(int amount,char *account, char * name )
{
  FILE  *fp;
  int n;
  char buf[ BUF_LENGTH ];
  memset (buf, '\0',  BUF_LENGTH);
  if ( !(fp = fopen( account , "r"))) return( ERROR );
  sem_wait(&mutex); //Entered critical code;
  fscanf (fp, "%s", buf);
  fclose( fp );
  //if ( amount != 0 )  printf("%s", "."); //uncomment to see the progress,not needed
  if (!( fp = fopen( account,"w" )))  return( ERROR );
  fprintf( fp,"%d", n = atoi(buf) + amount );
  fclose( fp );
  sem_post(&mutex); //Exited critical code
  return( n + amount );              /* New sum in account */
}
/*********************************************************************/
int write_to_log(int amount,char *log_file )  /* Appends amount to log_file */
{
  FILE *fp;
  if ((fp = fopen( log_file,"a")) == NULL ) return( ERROR );
  fprintf( fp,"%d\n",amount );
  fclose( fp );
  return( SUCCESS );
}
/***************************EOF****************************/
```

הרצה חלקית של הקוד אחרי תיקון סעיף 1:

```
lubuntu@lubuntu:~$ gcc agents1.c -o agents1 -pthread
lubuntu@lubuntu:~$ ./agents1 8
Creating 8 agent(s).
Finished creating 8 agent(s).
 Op of A1 = 46
 Op of A8 = -42
 Op of A8 = -11
 Op of A7 = -3
 Op of A7 = -1
 Op of A7 = 50
 Op of A7 = -42
 Op of A7 = -16
 Op of A5 = -32
 Op of A5 = 30
 Op of A6 = -1
 Op of A2 = -30
 Op of A4 = -30
 Op of A3 = -36
 Op of A1 = -42
 Op of A1 = -38
 Op of A6 = -3
 Op of A6 = -49
 Op of A6 = -3
 Op of A6 = -3
```

```
 Op of A4 = -27
 Op of A3 = -10
 Agent A3 : Succefull!
 Op of A1 = -10
 Op of A1 = -35
 Op of A4 = 44
 Op of A4 = -2
 Agent A4 : Succefull!
 Op of A1 = 43
 Agent A1 : Succefull!


 ----- Controler -----
 Sum of A1 = -55
 Sum of A2 = -197
 Sum of A3 = -14
 Sum of A4 = -71
 Sum of A5 = -201
 Sum of A6 = -196
 Sum of A7 = -137
 Sum of A8 = -332

Operations O.K. Controler: -703, Acount: -703
lubuntu@lubuntu:~$
```

2. הבקר מבצע את הבדיקה רק אחרי שכל התהליכונים מסתיימים וזאת בגלל הלולאה שמחכה לסיום כל התהליכונים שנמצאת בקוד לפני הקריאה לcheck_point:

```
while ( p >= 1 ) { sleep(2);} //!!!Have to transfer to check_point
```

ולכן נעביר את הלולאה הזאת לפונקציה check_point מה שבעצם גורם לזה שהבדיקות יהיו רנדומליות.

הערה: לא יצרנו תהליכון חדש שיבדוק ברנדומליות אלא כיוון שהתהליכון הראשי שלנו כעת רק מחכה לסיום התהליכונים האחרים שיצרנו אז במקום שהוא לא יעשה דבר, הוא יהיה זה שמבצע את את הבדיקות ברנדומליות לצורך כך העברנו את הלולאה לתוך check_point וכך הוא בודק את זה רנדומלי (וזה בעצם חיסכון בתהליכון נוסף כי אנחנו משתמשים בתהליכון הראשי שממילא מחכה במקום לייצר חדש).

כמו כן בעקבות השינוי של ה"מבקר" נצטרך לנעול בסמפורים את עדכון החשבון המשותף ביחד עם הכתיבה ללוג הפרטי של כל אחד בפונקציה go_agent וכן את הבדיקה של הסכום בחשבון המשותף ביחד עם הסכום של כל הלוגים של כל אחד בפונקציה check_point, כל זאת ע"מ למנוע תנאי מירוץ בין התהליכונים למעדכנים את החשבון וכותבים ללוג לבין ה"מבקר" (שהוא בעצם בתהליכון הראשי) שבודק האם התרחשה מעילה בחשבון.

(נצטרך להרחיב את הקטעים הקריטים ולכן לא נשתמש בסעיף 1).

השינויים שבקוד מופיעים בסוף!

3. הוספתי את הקוד הזה:

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

והסרתי את הלולאה שחיכתה שP יהיה 0 כלומר שיגמרו כל התהליכונים באופן פרימיטיבי.

את שורות הקוד הנ"ל מיקמתי לפני sem_destroy שבסיום המain.

ראו למטה קוד סופי!

4. הגדרנו את המשתנה p כ atomic_int במקום int רגיל כך שהפעולות p++ וp- - שבעצם הינם אוסף של 3 פעולות יפעלו כיחידה אטומית אחת ולא יווצרו תנאי מירוץ בשינוי הערך p.

ראו למטה קוד סופי!

הקוד במלואו אחרי השינויים של סעיפים 2,3,4:

המודגש באדום הינו שורה שהורדתי מהקוד.

המודגש בצהוב הינם שורות שהוספתי לקוד בסעיף 2.

המודגש בתכלת הינם שורות שהוספתי לקוד בסעיף 3.

המודגש בירוק הינם שורות שהוספתי לקוד בסעיף 4.

```
/*****************************************************************************************
File: agents.c
Compile:  gcc agents.c -o agents -pthread
Or use make, there is also clean target in Makefile
*****************************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdatomic.h> //!!! Put attension on it !!!!
// Usage: atomic_int i=0; After that it can be used i++; synchronized by paraller threads.
#include "agents.h"
```

```
/****************************************/
int init (int);
int check_point(int*);
void go_agent (char *);
int log_sum (char *);
int update(int, char *, char * );
int write_to_log(int, char *);
void fatal_error(char *,char *);
int gen_rand(int);
/*********************************************************************/
/*          Global Array - All Agents' names              */
char agents[ MAX_AGENTS ][ NAMES_LEN ] = AGENTS;
atomic_int p ; /* number of the started agentes */ ; /* number of the started agentes */
pthread_t thread[MAX_AGENTS]/*agents*/, cThread/*control thtead*/;
sem_t mutex;
/*********************************************************************/
/* Reset all files,Dispatch all agents and control actions */
int main(int ac,char **av )
{
sem_init(&mutex, 0, 1);
int i = 0, num_agents = 1;/* default */
if ( ac == 2 )  i = atoi( av[1] );  /* Check arguments */
else if (ac != 1) fatal_error("USAGE:agents [number_of_agents]\n", NULL);
/* initiate an account file with a start sum */
num_agents = init( i );
/* Main process distributes work to threads (agents).
   Agents perform independent operations.Every agent updates the account file
   after each operation and logs them in a personal log file */
for(i=0, p=0 ; i < num_agents; i++, p++ ){
if(pthread_create(&thread[p], NULL, (void *)go_agent, &agents[p]))
  { perror("ERROR creating thread."); --p; }
  }
i = p;
printf("Finished creating %d agent(s). \n",num_agents);
sleep(1);
/* The father waits for agents to return succesfully */
//while ( p >= 1 ) { sleep(2);} //!!!Have to transfer to check_point
/* check the consistency(log files vs.the account file. */
check_point( &i );
for (int j = 0; j < i; j++) {
      pthread_join(thread[j], NULL);
}
sem_destroy(&mutex);
return 0;
}

/*********************************************************************/
/*  initiate the account file with the starting sum
*  return the number of agents to be created in the range [1, MAX_AGENTS] */
int init(int num_agents )   /* Initiate auxiliary files,returns number agents */
{
 FILE *fp;
 int i;
 //srand( time( NULL ) ); // seeds rand()
```

```
 if ( num_agents < 1 || num_agents > MAX_AGENTS ){
   printf("%d agent(s) defaults to 1\n",num_agents);
   num_agents = 1;
  }
 printf("Creating %d agent(s). \n",num_agents);
 // delete old files; exit if fails
 for ( i = 0; i < num_agents; i++)
     if ((unlink( agents[i] ) < 0) && (errno != ENOENT)  )
        fatal_error("Deleting agents log", "");
 if (( fp = fopen( ACCOUNT,"w")) == NULL )
      fatal_error("Creating ACCOUNT file", "");
 fprintf( fp,"%d\n",START_SUM );
 fclose( fp );
 return( num_agents );
}
/*********************************************************************/
int check_point(int * num_agents ) /* Are agents' actions consistent ? */
{
while ( p >= 1 ) {
int i,n, control_sum , sum ;
 control_sum=0;
 if ( ( n = rand() % MAX_SLEEP  ) != 0 ) sleep( n * 3 );
 printf("\n\n ----- Controler ----- \n");
 sem_wait(&mutex); //Entered critical code;
 if ((sum = update(0,ACCOUNT, NULL )) == ERROR) return(ERROR);//get the sum by updating 0
 for ( i = 0; i < *num_agents; ++i )
      control_sum = control_sum + log_sum( agents[i] );
 sem_post(&mutex); //Exited critical code
 control_sum = START_SUM + control_sum;
 if  ( sum != control_sum )
 {
    fprintf( stderr,"\n Not consistent! Somebody steals?!\n");
    fprintf( stderr,"Controler: %d, Account: %d \n",control_sum,sum );
                fprintf( stderr," Terminating all agents!\n");
                return(1);
 }
 printf("\nOperations O.K. Controler: %d, Acount: %d\n", control_sum, sum );
 }
exit(SUCCESS);
}
/**********************************************************************/
int log_sum(char *log_file )    /* Returns the sum of log_file */
{
 FILE  *fp;
 int sum = 0,tmp_sum;
 if ( ( fp = fopen( log_file ,"r+") ) == NULL )
 {
    perror( log_file );
    printf("\nCan't Control %s agent!\n",log_file );
    return( 0 );
 }
  while ( fscanf( fp,"%d",&tmp_sum ) > 0 ) sum = sum + tmp_sum;
 fclose( fp );
 printf(" Sum of %s = %d\n",log_file,sum );
```

```
 return( sum );
}
/*****************************************************************/
void fatal_error(char *err_msg,char *perror_arg)  /* Prints Error messages */
{
fprintf(stderr,"\nERROR: %s. ", err_msg);
if ( perror_arg != NULL ) perror( perror_arg);n
exit(ERROR);
}
/*******************************************************************/
void go_agent(char *agent_name)      /* sends agent to work */
{
int n, i, amount;
  for( i = 0; i < OPERATIONS ; ++i )
            {
              amount = gen_rand( MAX_OPER );
              sem_wait(&mutex); //Entered critical code;
              if ( update( amount,ACCOUNT, agent_name ) == ERROR )
              {
                          fatal_error( "Can't Update Account File", ACCOUNT); //It includes process termination
              }
    else {
                if ( write_to_log( amount,agent_name ) == ERROR ){ // rollback
                  if (update(-amount, ACCOUNT, NULL ) == ERROR)
                   fprintf(stderr, "ERROR: Rollback failed. Agent %s\n", agent_name);
                fatal_error( "Can't log operations", agent_name); //It includes process termination
                 }
               }
              sem_post(&mutex); //Exited critical code
              printf(" Op of %s = %d\n",agent_name,amount );
              if ( ( n = rand() % MAX_SLEEP ) != 0 )  sleep( n );
            }
            p--;
  printf(" Agent %s : Succefull!\n",agent_name );
 return;
}
/*****************************************************************/
int gen_rand(int range)/*Generates random numbers between -rang and +range*/
{
 int n,sign;
 static int s=17;
 srand( time( NULL ) % s++);
 while( ( n = rand() ) == 0 );
 sign = (n/1111) % 2;
 n = ( (n/1117) % range ) + 1;
 if ( sign == 0 )  return( n );
 else return( 0 - n );
}
/*******************************************************************/
/* Updates account using amount. Returns new sum */
int update(int amount,char *account, char * name )
{
 FILE  *fp;
 int n;
```

```
 char buf[ BUF_LENGTH ];
 memset (buf, '\0',  BUF_LENGTH);
 if ( !(fp = fopen( account , "r"))) return( ERROR );
 fscanf (fp, "%s", buf);
 fclose( fp );
 //if ( amount != 0 )  printf("%s", "."); //uncomment to see the progress,not needed
 if (!( fp = fopen( account,"w" )))  return( ERROR );
 fprintf( fp,"%d", n = atoi(buf) + amount );
 fclose( fp );
 return( n + amount );              /* New sum in account */
}
/*********************************************************************/
int write_to_log(int amount,char *log_file )  /* Appends amount to log_file */
{
 FILE *fp;
 if ((fp = fopen( log_file,"a")) == NULL ) return( ERROR );
 fprintf( fp,"%d\n",amount );
 fclose( fp );
 return( SUCCESS );
}
/***************************EOF****************************/
```

דוגמת הרצה חלקית של הקוד עם שינויי סעיפים 2,3, ו4:

```
lubuntu@lubuntu:~$ ./agents 4
Creating 4 agent(s).
Finished creating 4 agent(s).
 Op of A1 = -42
 Op of A1 = -48
 Op of A1 = -32
 Op of A1 = -38
 Op of A4 = -48
 Op of A4 = -4
 Op of A4 = -3
 Op of A4 = -49
 Op of A4 = -39
 Op of A3 = -39
 Op of A2 = -39
 Op of A2 = -48
 Op of A2 = 50
 Op of A2 = -48
 Op of A2 = 50
 Op of A2 = -38


 ----- Controler -----
 Sum of A1 = -160
 Sum of A2 = -73
 Sum of A3 = -39
```

```
 Op of A1 = -33
 Op of A3 = 42
 Op of A3 = -4
 Op of A3 = -35


 ----- Controler -----
 Sum of A1 = -229
 Sum of A2 = -328
 Sum of A3 = -150
 Sum of A4 = -204

Operations O.K. Controler: -411, Acount: -411


 ----- Controler -----
 Sum of A1 = -229
 Sum of A2 = -328
 Sum of A3 = -150
 Sum of A4 = -204

Operations O.K. Controler: -411, Acount: -411
 Op of A4 = -40
 Op of A4 = 1
```

```
 Op of A1 = 28
 Op of A3 = 35
 Op of A4 = -1
 Op of A4 = -4
 Op of A4 = -3
 Op of A4 = 19
 Op of A4 = 11
 Op of A3 = -32
 Op of A3 = -1
 Op of A3 = 25
 Op of A1 = 42
 Agent A4 : Succefull!
 Agent A1 : Succefull!
 Op of A3 = 21
 Agent A3 : Succefull!


 ----- Controler -----
 Sum of A1 = -159
 Sum of A2 = -328
 Sum of A3 = -102
 Sum of A4 = -221

Operations O.K. Controler: -310, Acount: -310
lubuntu@lubuntu:~$
```