נועה מרציאנו 1556260990

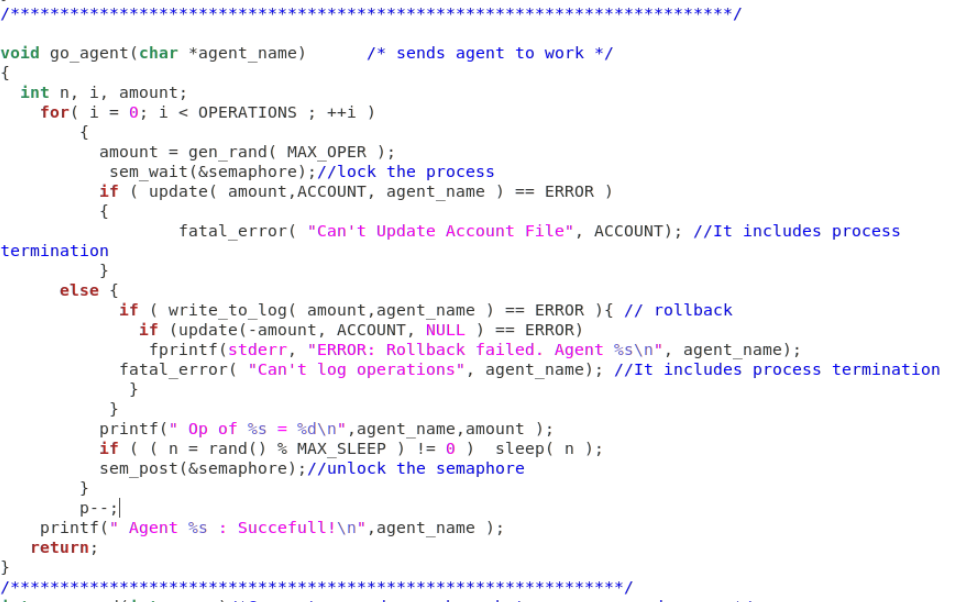
**תרגיל בית 2 – מערכות הפעלה**

Controler ושל Account**שאלה** **1**: הבעיה היא שכאש יוצרים יותר מסוכן אחד הכמות של

אינם שווים. הבעיה קורית מכיון שהקובץ הוא גלובלי וכל הסוכנים יכולים לגשת אליו בו זמנית ולשנות אותו וזה יוצר נתונים לא עקביים.

להלן הפונקציה שבה עשיתי שינויים על מנת שהקוד יעבוד כראוי.

ולאחר האתחול שלו.semaphoreכמובן לאחר שהגדרתי משתנה



**שאלה 2:**

עושה את הבדיקה שלו בסוף אחרי שכל הסוכנים גומרים את פעילותם.Controlerה-

הנה הפונקציות שבהם עשיתי שינויים:

char agents[ MAX\_AGENTS ][ NAMES\_LEN ] = AGENTS;

atomic\_int p ; /\* number of the started agentes \*/

pthread\_t thread[MAX\_AGENTS]/\*agents\*/, cThread/\*control thtead\*/;

sem\_t semaphore;//define a semaphore

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Reset all files,Dispatch all agents and control actions \*/

int main(int ac,char \*\*av )

{

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 \*/

sem\_init(&semaphore,0,1);//init the semaphore

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);

pthread\_t controllerT;

pthread\_create(&controllerT, NULL, (void \*)check\_point, &i);

pthread\_join(controllerT,0);

sem\_destroy(&semaphore);

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 ;

while(p)

{

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] );

sem\_post(&semaphore);

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);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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 );

sem\_post(&semaphore);//unlock the semaphore

}

p--;

printf(" Agent %s : Succefull!\n",agent\_name );

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 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(&semaphore);

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 \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/