07 SYNC - Synchronization Vaulsworg Aneka Soal Ujian Sistem Operasi Rahmat M. Samik-Ibrahim et.al. Rahmat M. Samik-Ibrahim et.al.

(c) 2016 - 2018 — Rev: 16 - 10-Apr-2018. Silakan mengubah, memperbanyak, serta mendistribusikan dokumen ini selama tidak menghapus ketentuan ini. URL: http://rms46.vlsm.org/2/202.pdf

1. **2016-1**

Circle or cross: "T" if True – "F" if False.

- T / FA semaphore is a data structure.
- T / FSemaphores can not be used for avoiding dead locks
- T / FA monitor is a programming language construct
- T / FMonitors encapsulate shared data structures.
- T / FBoth semaphores and monitors are distributed as function calls.
- T / FMonitors use condition variables, while semaphores do not.

2. **2016-2**

```
001 /*
                                                   030 void* thread3 (void* a) {
                                                          printf("T3X\n");
002 * (c) 2015-2016 Rahmat M. Samik-Ibrahim
                                                   031
002 * -- This is free software
                                                   032
                                                          sem_post
                                                                      (&sem[6]);
003 * Feel free to copy and/or modify and/
                                                   033
                                                          sem_post
                                                                      (&sem[2]);
004 * or distribute it, provided this notice,
                                                   034 }
004 * and the copyright notice, are preserved.
                                                   035
005 * REV04 Tue Dec 13 15:19:04 WIB 2016
                                                   036 void* thread4 (void* a) {
006 * START Wed Sep 30 00:00:00 UTC 2015
                                                                      (&sem[4]);
                                                   037
                                                          sem_wait
007
    */
                                                   038
                                                          printf("T44\n");
800
                                                   039
                                                          sem_wait
                                                                      (&sem[5]);
009 #include <stdio.h>
                                                   040
                                                          printf("T45\n");
010 #include <stdlib.h>
                                                   041
                                                          sem_wait
                                                                      (&sem[6]);
011 #include <semaphore.h>
                                                   042
                                                          printf("T46\n");
012 #include "99-myutils.h"
                                                   043 }
013 #define nSem 7
                                                   044
014
                                                   045 void main(void) {
015 sem_t sem[nSem];
                                                   046
                                                          printf("MAIN\n");
016
                                                   047
                                                          for (int ii=1;ii<nSem;ii++)</pre>
017 void* thread1 (void* a) {
                                                              sem_init(&sem[ii], 0, 0);
                                                   048
018
                   (&sem[1]);
                                                   049
                                                          daftar_trit
                                                                         (thread1);
       sem_wait
019
       printf("T1X\n");
                                                   050
                                                          daftar_trit
                                                                         (thread2);
020
       sem_post
                   (&sem[4]);
                                                   051
                                                          daftar_trit
                                                                         (thread3);
021 }
                                                   052
                                                          daftar_trit
                                                                         (thread4);
022
                                                   053
                                                          jalankan_trit ();
023 void* thread2 (void* a) {
                                                          beberes_trit ("TREXIT");
                                                   054
024
       sem_wait
                   (&sem[2]);
                                                   055 }
025
       printf("T2X\n");
026
       sem_post
                   (&sem[5]);
027
       sem_post
                   (&sem[1]);
028 }
```

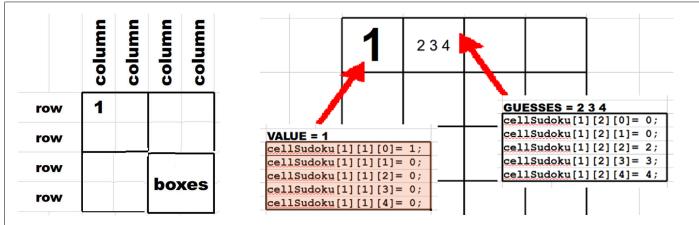
Write down the program output:

3. **2017-1**

```
Program Code of Synchronization (using 99-myutils.h and 99-myutils.c from the lab assignment)
001 /* (c) 2011-2017 Rahmat M. Samik-Ibrahim
                                                          039 void* Kiri (void* a) {
002 * This is free software. Feel free to copy and/or
                                                                 while (TRUE) {
                                                          040
003 * modify and/or distribute it, provided this
                                                                    cetak("Kiri-+-+-");
                                                          041
004 * notice, and the copyright notice, are preserved.
                                                                    sem_post (&syncKiriMod);
                                                          042
005 * REV01 Wed May 17 17:02:37 WIB 2017
                                                          043
                                                                    sem_wait (&syncModKiri);
006 * START Wed May 3 12:58:28 WIB 2017
                                                          044
007 *
                                                          045 }
008 * sem_init(), sem_wait(), sem_post(): semaphore
                                                          046 void* Moderator (void* a) {
009 * sleep(X): sleep X seconds
                                                          047
                                                                 int ii;
010 * daftar_trit(T): register thread T
                                                          048
                                                                 while (TRUE) {
011 * jalankan_trit(): start all registered threads.
                                                          049
                                                                    for (ii=0; ii<jmlKIRI; ii++)</pre>
012 * beberes_trit(): exit all threads above. */
                                                          050
                                                                       sem_wait (&syncKiriMod);
013 #define jmlKIRI
                        3
                                                          051
                                                                    for (ii=0; ii<jmlKANAN; ii++)</pre>
014 #define jmlKANAN
                        2
                                                          052
                                                                       sem_post (&syncModKanan);
015 #define SLEEP
                        1
                                                          053
                                                                    for (ii=0; ii<jmlKANAN; ii++)</pre>
016 #include <stdio.h>
                                                          054
                                                                       sem_wait (&syncKananMod);
017 #include <stdlib.h>
                                                          055
                                                                    for (ii=0; ii<jmlKIRI; ii++)</pre>
018 #include <semaphore.h>
                                                          056
                                                                       sem_post (&syncModKiri);
019 #include <unistd.h>
                                                          057
020 #include "99-myutils.h"
                                                          058 }
021 sem_t mutexID, syncModKiri, syncModKanan;
                                                          059 int main(int argc, char * argv[]) {
022 sem_t syncKiriMod, syncKananMod;
                                                          060
                                                                 int ii;
023 int
            sequence = 0;
                                                                 sem_init (&syncModKiri, 0, 0);
                                                          061
024
                                                                 sem_init (&syncModKanan, 0, 0);
                                                          062
                                                                 sem_init (&syncKiriMod, 0, 0);
025 void cetak(char* posisi) {
                                                          063
026
    sem_wait (&mutexID);
                                                          064
                                                                 sem_init (&syncKananMod, 0, 0);
027
       printf("%s (%d)\n", posisi, sequence++);
                                                          065
                                                                 sem_init (&mutexID,
028
       fflush(NULL);
                                                          066
029
       sem_post (&mutexID);
                                                          067
                                                                 for (ii = 0 ; ii < jmlKANAN; ii++)</pre>
030
       sleep(SLEEP);
                                                          068
                                                                    daftar_trit(Kanan);
031 }
                                                          069
                                                                 for (ii = 0 ; ii < jmlKIRI; ii++)</pre>
032 void* Kanan (void* a) {
                                                          070
                                                                    daftar_trit(Kiri);
033
                                                          071
       while (TRUE) {
                                                                 daftar_trit(Moderator);
034
          sem_wait (&syncModKanan);
                                                          072
035
          cetak("-+-+-Kanan");
                                                          073
                                                                 jalankan_trit();
036
          sem_post (&syncKananMod);
                                                          074
                                                                 beberes_trit("Selese...");
037
       }
                                                          075 }
038 }
```

Write down the next 5 lines of the program output:																				
K	i	r 	i	<u>-</u>	+	<u>-</u>	+	<u>-</u>	+	-		(0)	 	 	 	 	 	

4. **2017-2**



In this mini-Sudoku 4x4 — each **column**, **row**, and 2x2 sub-grid **box** — should contain the digits of: 1, 2, 3, or 4. This C program "07-mini-sudoku-4x4.c" is using a 3 dimensional array called "cellSudoku[][][]". If "cellSudoku[row][column][0] == 0" (or: no value), "cellSudoku[row][column][1]" to "[4]" will contain of all values that are possible (or guesses).

- (a) How many Semaphores were created in that program?
- (b) Specify what the names of those Semaphores are!
- (c) How many threads were created in that program?
- (d) Specify what the (unique) names of those threads are!
- (e) How many critical zone(s) are there in that program?
- (f) Specify the line numbers of those critical zone(s)!
- (g) Name the function that receives the input file "07-data.txt" in that program above!

Program Code 07-mini-sudoku-4x4.c (using 99-myutils.h and 99-myutils.c from the DEMO set.)

```
001 /*
                                                      052 // Filling the CELLs
002 * (c) 2017 Rahmat M. Samik-Ibrahim
                                                      053 void
003 * http://rahmatm.samik-ibrahim.vlsm.org/
                                                      054 fillCell(int rowCell,int colCell,int valCell)
004 * This is free software.
005 * REV04 Tue Dec 12 20:35:44 WIB 2017
                                                      056
                                                             sem_wait (&mutexing);
006 * START Mon Dec 4 18:52:57 WIB 2017
                                                      057
                                                             // Filling "valCell" into
007 */
                                                      058
                                                             // cellSudoku[rowCell, colCell];
800
                                                      059
                                                             cellSudoku[rowCell][colCell][0] = valCell;
009 #include <stdio.h>
                                                      060
                                                             // This is Cell is "taken".
010 #include <stdlib.h>
                                                      061
                                                             // Eliminate all guesses!
011 #include <unistd.h>
                                                      062
                                                             for (int ii=1; ii<SSIZE+1; ii++) {</pre>
012 #include "99-myutils.h"
                                                      063
                                                                cellSudoku[rowCell][colCell][ii] = 0;
                                                      064
013 #define WaitSudoku 3
014 #define SSIZE
                                                      065
                                                             // Deleting "valCell"
                                                             // from all "columns guess"
015 #define TOTALSIZE SSIZE * SSIZE
                                                      066
016
                                                             for (int ii=1; ii<SSIZE+1; ii++) {</pre>
                                                      067
017 int
           globalExit=FALSE;
                                                      068
                                                                cellSudoku[rowCell][ii][valCell] = 0;
018 sem_t mutexing;
                                                      069
                                                      070
                                                             // Delete "valCell" from all "rows guess".
019 sem_t syncing1;
020 sem_t syncing2;
                                                      071
                                                             for (int ii=1: ii<SSIZE+1: ii++) {</pre>
021
                                                      072
                                                                cellSudoku[ii][colCell][valCell] = 0;
022 // cellSudoku[row][column][0]
                                                      073
                                      = value
                                                             // Delete "valCell" from all "boxes guess"
023 // cellSudoku[row][column][1-4] = guesses
                                                      074
024 // \text{ if (value != 0) all guesses = 0}
                                                      075
                                                             rowCell = 1 + 2*((rowCell - 1)/2);
025 //
                              (no more guesses)
                                                      076
                                                             colCell = 1 + 2*((colCell - 1)/2);
026 int cellSudoku[][SSIZE+1][SSIZE+1]={
                                                      077
                                                             for (int ii=rowCell; ii<rowCell+2; ii++) {</pre>
027
       \{\}, \{\{\}, \{0,1,2,3,4\}, \{0,1,2,3,4\},
                                                      078
                                                                for (int jj=colCell; jj<colCell+2; jj++){</pre>
                \{0,1,2,3,4\}, \{0,1,2,3,4\}\},
028
                                                      079
                                                                   cellSudoku[ii][jj][valCell] = 0;
029
          \{\{\}, \{0,1,2,3,4\}, \{0,1,2,3,4\},
                                                      080
030
                                                      081
                \{0,1,2,3,4\}, \{0,1,2,3,4\}\},\
                                                             }
031
          \{\{\}, \{0,1,2,3,4\}, \{0,1,2,3,4\},
                                                     082
                                                             sem_post (&mutexing);
                \{0,1,2,3,4\}, \{0,1,2,3,4\}\},\
032
                                                      083 }
033
                                                      084
          \{\{\}, \{0,1,2,3,4\}, \{0,1,2,3,4\},
034
                \{0,1,2,3,4\}, \{0,1,2,3,4\}\}
                                                      085 // From Standard Input into Cell using
035 };
                                                      086 // fillCell -- SCAN INPUT: scanf()
036
                                                      087 // is the oposite of printf()
037 // Print Cells
                                                      088 void inputCell(void) {
038 void printCells(char* state) {
                                                      089
                                                             for (int ii=0; ii < TOTALSIZE; ii++) {</pre>
039
       printf ("\nSudoku Cells: %s\n", state);
                                                      090
                                                                int tmpCell=0;
040
                                                      091
                                                                scanf("%d", &tmpCell);
                int jj=1; jj<SSIZE+1; jj++) {</pre>
041
          for (int kk=1; kk<SSIZE+1; kk++) {</pre>
                                                      092
                                                                int rowCell = ii/4 + 1:
042
              int cell=cellSudoku[jj][kk][0];
                                                     093
                                                                int colCell = ii\%4 + 1;
043
              if (cell == 0 || cell == 5)
                                                      094
                                                                if (tmpCell != 0) {
044
                              printf ("[]");
                                                      095
                                                                       fillCell(rowCell,colCell,tmpCell);
045
                       printf ("[%d]", cell);
                                                      096
                                                                }
             else
              if (kk == SSIZE) printf ("\n");
                                                      097
046
                                                             }
047
          }
                                                      098 }
048
       }
       fflush(NULL);
049
050 }
```

```
Program Code 07-mini-sudoku-4x4.c (using 99-myutils.h and 99-myutils.c from the DEMO set.)
100 // CellWatcher
                                                     141 // Display Sudoku
101 int cwID = 0;
                                                     142 void* displaySudoku (void* a) {
102 void* cellWatcher (void* a) {
                                                     143
                                                             printCells("INITIAL");
      sem_wait (&syncing1);
                                                     144
                                                             for(int jj=0;jj<TOTALSIZE;jj++)</pre>
104
      sem_wait (&mutexing);
                                                     145
                                                                sem_post(&syncing1);
105
      int rowCell = cwID/4 + 1;
                                                             for(int jj=0;jj<TOTALSIZE;jj++)</pre>
                                                     146
106
      int colCell = cwID%4 + 1;
                                                     147
                                                                sem_wait(&syncing2);
107
      cwID++;
                                                     148
                                                             printCells("RESULT");
108
      sem_post (&mutexing);
                                                     149 }
109
      int localExit=FALSE;
                                                     150
      while (!localExit && !globalExit) {
110
                                                     151 // This is MAIN
        int tmpCell=0, nZero=0;
                                                     152 void main(void) {
111
112
        for (int ii=1; ii<SSIZE+1; ii++) {</pre>
                                                     153
                                                             printf
                                                                      ("MAIN: START\n");
113
          if(cellSudoku[rowCell][colCell][ii]==0)
                                                     154
                                                             sem_init (&mutexing, 0, 1);
114
                                                     155
                                                             sem_init (&syncing1, 0, 0);
            nZero++;
115
          else
                                                     156
                                                             sem_init (&syncing2, 0, 0);
116
            tmpCell=ii;
                                                     157
                                                             inputCell();
        }
                                                             for (int ii=0; ii<TOTALSIZE; ii++) {</pre>
117
                                                     158
        if (nZero==3)
                                                     159
                                                                daftar_trit(cellWatcher);
118
119
          fillCell(rowCell, colCell, tmpCell);
                                                     160
                                                             }
120
        localExit =
                                                     161
                                                             daftar_trit
                                                                            (displaySudoku);
121
          cellSudoku[rowCell][colCell][0]!=0;
                                                     162
                                                             daftar_trit
                                                                            (managerSudoku);
122
      }
                                                     163
                                                             jalankan_trit ();
                                                     164
123
      fflush(NULL);
                                                             beberes_trit ("\nTRIT: EXIT");
124
                                                     165 }
      sem_post (&syncing2);
125 }
126
127 // Timeout after "WaitSudoku"
128 void* managerSudoku (void* a) {
129
      sleep(WaitSudoku);
130
      for (int ii=0; ii<TOTALSIZE; ii++) {</pre>
131
        int rowCell = ii/4 + 1;
132
        int colCell = ii\%4 + 1;
133
        if(cellSudoku[rowCell][colCell][0]==0){
           cellSudoku[rowCell][colCell][0] = 5;
134
135
        }
136
        sem_post (&syncing2);
137
      }
      globalExit = TRUE;
138
139 }
    This following is the output of executing:
                                                          Bonus Question:
                                                          What is inside file 07-data.txt?
     ./07-mini-sudoku-4x4 < 07-data.txt
    MAIN: START
    Sudoku Cells: INITIAL
     [][][3]
     [][1][4][]
     [ ][2][3][ ]
     [1][][][]
    Sudoku Cells: RESULT
     [2] [4] [1] [3]
     [3] [1] [4] [2]
     [4][2][3][1]
     [1] [3] [2] [4]
    TRIT: EXIT
```