CSGE602055 Operating Systems
CSF2600505 Sistem Operasi
Minggu 07: Synchronization

Rahmat M. Samik-Ibrahim

Universitas Indonesia

http://rms46.vlsm.org/2/207.html

REV092 2-NOV-2017

OS172 | INT TU/TH 13:00-15:00 | EXT TH 19:00-21:50

Minggu 00	29 Aug - 05 Sep 2017	Intro & Review
Minggu 01	07 Sep - 12 Sep 2017	IPR, SED, AWK, REGEX, & Scripting
Minggu 02	14 Sep - 19 Sep 2017	Protection, Security, Privacy, & C-language
Minggu 03	26 Sep - 30 Sep 2017	BIOS, Loader, Systemd, & I/O
Minggu 04	03 Okt - 07 Okt 2017	Addressing, Shared Lib, Pointer & I/O Programming
Minggu 05	10 Okt - 14 Okt 2017	Virtual Memory
Ming. UTS	15 Okt - 24 Okt 2017	
Minggu 06	26 Okt - 31 Okt 2017	Concurency: Processes & Threads
Minggu 07	02 Nov - 07 Nov 2017	Synchronization
Minggu 08	09 Nov - 14 Nov 2017	Scheduling & Network Sockets Programming
Minggu 09	16 Nov - 21 Nov 2017	File System & Persistent Storage
Minggu 10	23 Nov - 28 Nov 2017	Special Topic: Blockchain
Cadangan	30 Nov - 09 Des 2017	
Ming. UAS	10 Des - 23 Des 2017	

Agenda I

Week 07: Synchronization

- Reference: (OSCE2e ch5) (UCB 7/8) (UDA P3L3/4) (OLD 04)
- The Critical Section Problem
- Race Condition
- Peterson's Solution
- Semaphores
- Classical Problems
  - Bounded-Buffer Problem
  - Readers and Writers Problem
  - Dining-Philosophers Problem
- Resource and Allocation Graph

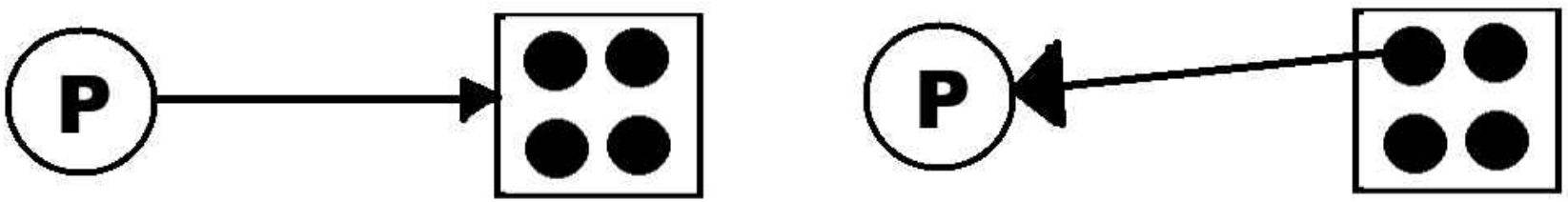

Figure: Request and Holding

Peterson's Solution

Process 0

```
   flag[0]=                    turn=

do {
    flag[0] = true
    turn = 1
    while (flag[1] && turn == 1)
      (do nothing);
    [CRITICAL SECTION];
    flag[0] = false
    [REMAINDER SECTION];
} while(true);
```

Process 1

```
   flag[1]=

do {
    flag[1] = true
    turn = 0
    while (flag[0] && turn == 0)
      (do nothing);
    [CRITICAL SECTION];
    flag[1] = false
    [REMAINDER SECTION];
} while(true);
```

Semaphore

- Dijkstra's Seinpalen (1963): Probeer (Try) en Verhoog (+1)
- Semaphore: Wait(S) and Signal(S)
- Linux System Calls: `sem_init()`, `sem_wait()`, and `sem_post()`

```
# Semaphore (Seinpalen)
# Wait (Probeer)

wait(S) {
   while (S <= 0)
      ; // busy wait
   S--;
}

# Signal (Verhoog)

signal(S) {
   S++;
}
```

Deadlock and Starvation

- Deadlock Characterization
  - Mutual exclusion
  - Hold and wait
  - No preemption
  - Circular wait
- Banker's Algorithm
- Deadlock Prevention
- Deadlock Avoidence
- How do Operating Systems handle Deadlocks?

IGNORE THE PROBLEM!
Pretending that deadlocks never occur
Just **RESET/REBOOT** it
This is how they **DO IT**!

99-myutils.h

```
/*
 * (c) 2011-2016 Rahmat M. Samik-Ibrahim -- This is free software
 */

#define MAX_THREAD 256
#define BUFFER_SIZE  5
#define TRUE         1
#define FALSE        0

typedef struct {
   int   buffer[BUFFER_SIZE];
   int   in;
   int   out;
   int   count;
} bbuf_t;

void daftar_trit   (void* trit);      // mempersiapkan "trit"
void jalankan_trit (void);            // menjalankan dan menunggu hasil dari
                                      // "daftar_trit"
void beberes_trit  (char* pesan);     // beberes menutup "jalankan_trit"

void rehat_acak    (long max_mdetik); // istirohat acak "0-max_mdetik" (ms)

void init_buffer   (void);            // init buffer
void enter_buffer  (int entry);       // enter  an integer item
int remove_buffer  (void);            // remove the item

void init_rw       (void);            // init readers writers
int  startRead     (void);            // start reading
int  endRead       (void);            // end reading
void startWrite    (void);            // start writing
void endWrite      (void);            // end writing
```

99-myutils.c

```
/*
 * (c) 2011-2016 Rahmat M. Samik-Ibrahim -- This is free software
 * Feel free to copy and/or modify and/or distribute it,
 * provided this notice, and the copyright notice, are preserved.
 * REV01 Wed Nov  2 11:49:55 WIB 2016
 * REV00 Xxx Sep 30 XX:XX:XX UTC 2015
 * START Xxx Mar 30 02:13:01 UTC 2011
 */

#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#include "99-myutils.h"
sem_t    mutex, db, empty, full, rmutex, wmutex;

/* TRIT ************************************************/
int       jumlah_trit = 0;

void*     trits  [MAX_THREAD];
pthread_t trit_id[MAX_THREAD];

void daftar_trit(void *trit) {
   if(jumlah_trit >= MAX_THREAD) {
      printf("\n ERROR MAX daftar_trit %d\n",jumlah_trit);
      exit(1);
   }
   trits[jumlah_trit++] = trit;
}
```

99-myutils.c (2)

```
void beberes_trit(char* pesan) {
   if (pesan != NULL)
      printf("%s\n",pesan);
   pthread_exit(NULL);
}

/* REHAT *********************************************/
int  pertamax    = TRUE;

void rehat_acak(long max_mdetik) {
   struct timespec tim;
   long            ndetik;

   if (pertamax) {
      pertamax = FALSE;
      srandom((unsigned int) time (NULL));
   }
   ndetik      = random() % max_mdetik;
   tim.tv_sec  = ndetik   / 1000L;
   tim.tv_nsec = ndetik   % 1000L * 1000000L;
   nanosleep(&tim,NULL);
}
```

99-myutils.c (3)

```
/* BOUNDED BUFFER ****************************************/
bbuf_t buf;
void init_buffer(void) {
   buf.in    = 0;
   buf.out   = 0;
   buf.count = 0;
   sem_init   (&mutex, 0, 1);
   sem_init   (&empty, 0, BUFFER_SIZE);
   sem_init   (&full,  0, 0);
}

void enter_buffer(int entry) {
   sem_wait(&empty);
   sem_wait(&mutex);
   buf.count++;
   buf.buffer[buf.in] = entry;
   buf.in = (buf.in+1) % BUFFER_SIZE;
   sem_post(&mutex);
   sem_post(&full);
}

int remove_buffer(void) {
   int item;
   sem_wait(&full);
   sem_wait(&mutex);
   buf.count--;
   item = buf.buffer[buf.out];
   buf.out = (buf.out+1) % BUFFER_SIZE;
   sem_post(&mutex);
   sem_post(&empty);
   return item;
}
```

99-myutils.c (4)

```
/* READERS WRITERS ************************************/
int readerCount;
void init_rw(void) {
   readerCount = 0;
   sem_init   (&mutex,  0, 1);
   sem_init   (&rmutex, 0, 1);
   sem_init   (&wmutex, 0, 1);
   sem_init   (&db,     0, 1);
}

int startRead(void) {
   sem_wait(&mutex);
   if (++readerCount == 1 )
      sem_wait(&db);
   sem_post(&mutex);
   return readerCount;
}

int endRead(void) {
   sem_wait(&mutex);
   if (--readerCount == 0 )
      sem_post(&db);
   sem_post(&mutex);
   return readerCount;
}

void startWrite(void) {
   sem_wait(&db);
}
void endWrite(void) {
   sem_post(&db);
}
```

00-thread

```
/* (c) 2015-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 * REV03 Wed Nov  1 15:17:08 WIB 2017
 * REV02 Tue Apr 18 15:28:19 WIB 2017
 * REV01 Wed Nov  2 11:49:30 WIB 2016
 * START Xxx Sep 30 XX:XX:XX UTC 2015
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>
#include "99-myutils.h"
#define  LOOP0    10
#define  LOOP1   500
#define  LOOP2  1000
#define  LOOP3 10000

volatile int loop = LOOP0;
int          share;
```

00-thread (2)

```
void* thread1 (void* a) {
   int ii, jj, kk;
   printf("I am a thread no 1\n");
   sleep(1);
   share = 1000;
   while (loop > 0) {
      for (ii=0;ii<LOOP1;ii++) {
         for (jj=0;jj<LOOP2;jj++) {
              ;
         }
      }
      share++;
   }
}

void* thread2 (void* a) {
   int ii, jj, kk;
   printf("I am a thread no 2\n");
   sleep(1);
   share = 2000;
   while (loop > 0) {
      for (ii=0;ii<LOOP1;ii++) {
         for (jj=0;jj<LOOP2;jj++) {
              ;
         }
      }
      share--;
   }
}
```

00-thread (3)

```
void* thread3 (void* a) {
   int ii, jj, kk;
   printf("I am a thread no 3\n");
   sleep(1);
   while (loop-- > 0) {
      for (ii=0;ii<LOOP3;ii++) {
         for (jj=0;jj<LOOP3;jj++) {
               ;
         }
      }
      printf("SHARE = %4.4d\n", share);
   }
}
void main(void) {
   daftar_trit   (thread1);
   daftar_trit   (thread2);
   daftar_trit   (thread3);
   jalankan_trit ();
   printf        ("I am MAIN\n");
   beberes_trit  ("Done...");
}
```

00-thread (4)

```
>>>>> $ 00-thread
I am a thread no 1
I am a thread no 2
I am a thread no 3
SHARE = 1994
SHARE = 1989
SHARE = 1985
SHARE = 1977
SHARE = 1966
SHARE = 1954
SHARE = 1944
SHARE = 1933
SHARE = 1923
SHARE = 1923
I am MAIN
Done...
>>>>> $ 00-thread
I am a thread no 2
I am a thread no 1
I am a thread no 3
SHARE = 0992
SHARE = 0985
SHARE = 0987
SHARE = 0994
SHARE = 0991
SHARE = 0982
SHARE = 0974
SHARE = 0967
SHARE = 0959
SHARE = 0959
I am MAIN
Done...
>>>>> $
```

01-thread

```
>>>>> $ cat 01-thread.c
/*
 * (c) 2015-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 * REV02 Wed Nov  1 16:48:40 WIB 2017
 * REV01 Wed Nov  2 11:49:39 WIB 2016
 * START Xxx Sep 30 XX:XX:XX UTC 2015
 */

#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include "99-myutils.h"

sem_t generik;
sem_t generik2;
```

01-thread (2)

```
void* thread1 (void* a) {
   sem_wait   (&generik);
   printf("THREAD1: I am second!\n");
   sem_post   (&generik2);
}

void* thread2 (void* a) {
   printf("THREAD2: I am first!\n");
   sem_post   (&generik);
}

void* thread3 (void* a) {
   sem_wait   (&generik2);
   printf("THREAD3: I am last!\n");
}

void main(void) {
   sem_init      (&generik,  0, 0);
   sem_init      (&generik2, 0, 0);
   daftar_trit   (thread1);
   daftar_trit   (thread2);
   daftar_trit   (thread3);
   jalankan_trit ();
   beberes_trit  ("Bye Bye Main...");
}
```

```
>>>>> $ 01-thread
THREAD2: I am first!
THREAD1: I am second!
THREAD3: I am last!
Bye Bye Main...
```

02-prodkon

```
>>>>> $ cat 02-prodkon.c
/*
 * (c) 2011-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 * REV02 Wed Nov  1 16:50:50 WIB 2017
 * REV01 Wed Nov  2 11:20:30 WIB 2016
 * REV00 Xxx Sep 30 XX:XX:XX UTC 2012
 * START Xxx Mar 30 02:13:01 UTC 2011
 */

#include <stdio.h>
#include <stdlib.h>
#include "99-myutils.h"

#define P_REHAT 2000
#define K_REHAT 2000
int produk = 0;

void* Produsen (void* a) {
   printf("Produsen siap...\n");
   while (TRUE) {
      printf("P: REHAT ******\n");
      rehat_acak(P_REHAT);
      printf("P: PRODUKSI %d\n", produk);
      enter_buffer  (produk++);
   }
}
```

02-prodkon (2)

```
void* Konsumen (void* a) {
   printf   ("                          Konsumen siap...\n");
   while (TRUE) {
      printf("                          K: REHAT ******\n");
      rehat_acak(K_REHAT);
      printf("                          K: KONSUMSI %d\n", remove_buffer());
   }
}
int main(int argc, char * argv[])
{
   init_buffer();
   daftar_trit(Produsen);
   daftar_trit(Konsumen);
   jalankan_trit();
   beberes_trit("Selese...");
}
##################
>>>>> $ ./02-prodkon
Produsen siap...
P: REHAT ******
                        Konsumen siap...
                        K: REHAT ******
P: PRODUKSI 0
P: REHAT ******
                        K: KONSUMSI 0
                        K: REHAT ******
P: PRODUKSI 1
P: REHAT ******
P: PRODUKSI 2
P: REHAT ******
                        K: KONSUMSI 1
                        K: REHAT ******
```

Rock Paper Scissors Lizard Spock

```
/*
 * (c) 2014-2016 Rahmat M. Samik-Ibrahim
 * -- This is free software
 * Feel free to copy and/or modify and/or
 * distribute it,  provided this notice, and
 * the copyright notice, are preserved.
 * REV01 Wed Nov  2 11:20:30 WIB 2016
 * REV00 Xxx Sep 30 XX:XX:XX UTC 2015
 * START Xxx Oct 19 XX:XX:XX UTC 2014
 */
```

RPSLS 1

```
// *Rock*Paper*Scissors*Lizard*Spock*
// Invented by Sam Kass and Karen Bryla
// Rock crushes Scissors
// Rock crushes Lizard
// Paper covers Rock
// Paper disproves Spock
// Scissors cut Paper
// Scissors decapitate Lizard
// Lizard eats Paper
// Lizard poisons Spock
// Spock vaporizes Rock
// Spock smashes Scissors
```

RPSLS 2

```
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include "99-myutils.h"
#define nPlayers 2
#define nWeapons 5
int     playerSEQ=1;
int     myWeapon[nPlayers+1];
sem_t   mutex, sync1, sync2;

// (0=Rock) (1=Paper) (2=Scissors) (3=Lizard) (4=Spock)
char* weaponName[nWeapons]= {
   "Rock", "Paper", "Scissors", "Lizard", "Spock"
};
```

RPSLS 3

```
// '-' = draw  'v' = win  'x' = lose
char weaponTable[nWeapons][nWeapons] = {
   {'-','x','v','v','x'},
   {'v','-','x','x','v'},
   {'x','v','-','v','x'},
   {'x','v','x','-','v'},
   {'v','x','v','x','-'}
};
void waitPlayers() {
   for (int ii=0; ii < nPlayers; ii++)
      sem_wait(&sync1);
}
void postPlayers() {
   for (int ii=0; ii < nPlayers; ii++)
      sem_post(&sync2);
}
```

RPSLS 4

```
void* playerThread (void* a) {
   int      playerID;
   sem_wait (&mutex);
   playerID=playerSEQ++;
   sem_post (&mutex);
   printf("Player[%d]: READY\n",playerID);
   sem_post (&sync1);
   sem_wait (&sync2);
   myWeapon[playerID] = rand() % nWeapons;
   printf("Player[%d]: %s\n",
      playerID, weaponName[myWeapon[playerID]]);
   sem_post (&sync1);
}
```

RPSLS 5

```
void* refereeThread (void* a) {
   waitPlayers();
   printf("Referee:   ALL READY!\n");
   postPlayers();
   waitPlayers();
   char result =
      weaponTable[myWeapon[1]][myWeapon[2]];
   if (result == '-')
      printf("Referee:   DRAW!\n");
   else if (result == 'v')
      printf("Referee:   Player[1] WINS!\n");
   else
      printf("Referee:   Player[2] WINS!\n");
}
```

RPSLS 6

```
void main() {
   // randomize with a time seed
   srand(time(NULL));
   sleep(1);
   // init semaphore mutex = 1 syncx = 0
   sem_init (&mutex, 0, 1);
   sem_init (&sync1, 0, 0);
   sem_init (&sync2, 0, 0);
   // register and execute threads
   daftar_trit (refereeThread);
   for (int ii=0; ii<nPlayers; ii++)
      daftar_trit (playerThread);
   jalankan_trit ();
   beberes_trit  ("Goodbye...");
}
```

tba

tba

The End

- This is the end of the presentation.