UTS 2016/2017

1. **Introduction & OS Structure** (bobot 15)
   a. Sistem Komputer yang kita kenal saat ini menggunakan konsep Stored Program dari Von Neumann. Jelaskan mengenai konsep tersebut
   b. Satu aspek penting dari Sistem Operasi untuk meningkatkan utilisasi CPU adalah Multiprogramming. Apa yang anda ketahui tentang multiprogramming? Dan apa perbedaannya dengan Multitasking?
   c. Apa fungsi dari System Call?

a) Konsep "Stored Program" yang diusulkan oleh John von Neumann merupakan dasar dari arsitektur komputer modern. Konsep ini mengarah pada pemisahan yang jelas antara data dan instruksi program, yang semuanya disimpan dalam memori komputer. Dengan penyimpanan program dalam memori, instruksi program dapat dimodifikasi selama eksekusi, memungkinkan untuk percabangan dan perulangan dalam program. Eksekusi instruksi dilakukan secara berurutan oleh unit pemroses sentral (CPU) dalam siklus fetch-execute. Prinsip ini memberikan fleksibilitas yang besar dalam pemrograman dan penggunaan komputer, serta menjadi dasar bagi pengembangan komputer modern yang mampu menjalankan berbagai jenis program dengan efisien.

b) Multiprogramming adalah kondisi dimana kita memaksimalkan kinerja CPU dengan menjalankan beberapa program sekaligus di dalam memori komputer (disebut proses) dalam satu waktu dengan mengorganisasi program-program tersebut sehingga selalu pasti ada program yang dijalankan oleh CPU. Dengan kata lain, beberapa program ada dalam keadaan aktif secara bersamaan di dalam sistem, tanpa harus menunggu satu sama lain untuk selesai sebelum program berikutnya dapat dimulai. Pada multiprogramming, program-program dijalankan dalam satu waktu dengan cara mengorganisasi program-program tersebut. Sedangkan pada multitasking, program-program dijalankan dalam satu waktu dengan cara switching program dengan frekuensi tinggi sehingga memberi user response time yang cepat.

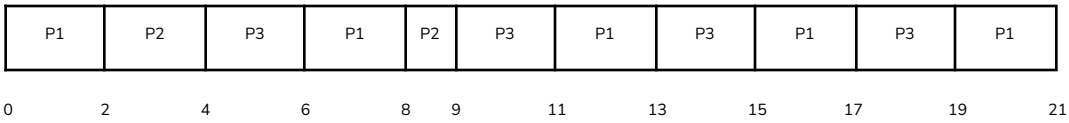
c) System call memberikan sebuah interface yang menghubungkan suatu program ke service yang ada di sistem operasi.

2. **Schedulling** (bobot 20)
   a. Penjadwalan dengan Round Robin untuk 3 proses P1, P2 dan P3 yang masing masing membutuhkan waktu eksekusi 10, 3, dan 8 satuan waktu. Quantum waktu yang digunakan adalah 2 satuan waktu.
      i. Ilustrasikan penjadwalannya dengan diagram waktu (Gantt Chart)
      ii. Berapa Average Waiting Time-nya?
   b. Untuk penjadwalan realtime, jika ada 2 proses P1 (dengan p1=50, t1=25) dan P2 (dengan p2=75, t2=30)
      i. Apakah bisa digunakan Rate-Monotonic Scheduling?
      ii. Ilustrasikan penjadwalan kedua proses dengan Earliest Deadline First (EDF)

a) Quantum = 2

P1 = 10, P2 = 3, P3 = 8

i) Gantt Chart



ii) Average waiting time?

P1 = 4 + 3 + 2 + 2 = 11

P2 = 2 + 4 = 6

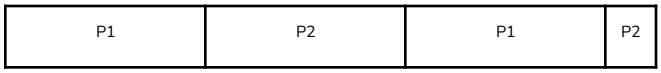
P3 = 4 + 3 + 2 + 2 = 11

Avg = $\frac{11 + 6 + 11}{2}$ = 9.33

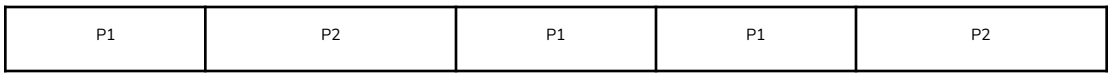
b) P1 = 50, t1 = 25 → deadline 50 - 100 - 150 - dst

P2 = 75, t2 = 30 → deadline 75 - 150 - 225 - 300 - dst

i) Rate monotonic scheduling, P2 di periode kedua nyampe di 80 pdhl dl di 75 jd gbs



ii) Earliest deadline first, di semua periode P1 P2 semuanya sebelum deadline

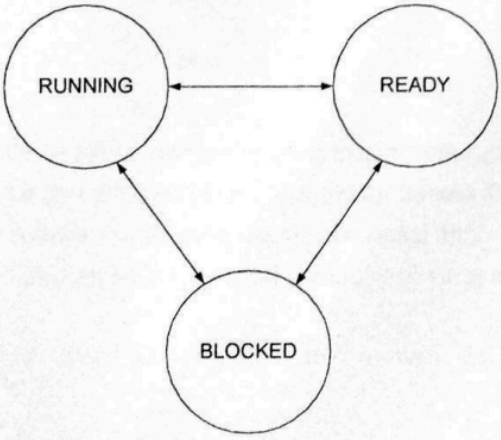


3. **Proses/IPC** (bobot 15)
   Bandingkanlah kedua model interprocess communication berikut: message passing dan shared memory.

Shared memory : Producer process menghasilkan informasi yang kemudian dikonsumsi oleh consumer process. Kedua proses harus saling tersinkronisasi sehingga tidak ada consumer yang mencoba untuk mengonsumsi item yang belum diproduksi.

Message passing : Untuk berkomunikasi, kedua proses harus melakukan communication link dan exchange pesan dengan operasi send dan receive. Ada 2 jenis communication. Direct berarti harus eksplisit sebut nama satu sama lain, biasanya bi-directional. Indirect berarti punya mailbox dengan id unik dan diimplementasikan pada semacam fungsi. Link komunikasi dibuat jika beberapa proses sharing mailbox yang sama dan dapat diasosiasikan dengan banyak proses, bisa bi-directional ataupun uni-directional.

4. **Thread** (bobot 15)



Jelaskan kondisi apa saja yang mengakibatkan sebuah thread berpindah dari satu state ke state lain pada diagram di atas. Jelaskan pula, jika ada, transisi antar state yang tidak mungkin terjadi

Running -> ready : interrupt

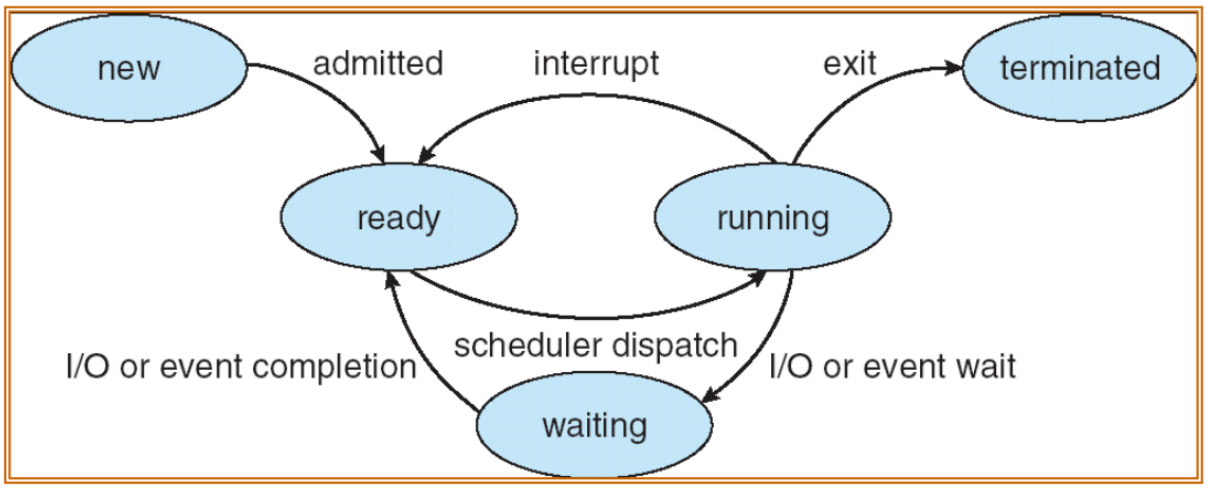
Ready -> running : scheduler dispatch

Ready -> waiting : gabisa

Waiting -> ready : I/O or event completing

Running -> waiting : I/O or event wait

Waiting -> running -> gabisa



5. **Deadlock** (bobot 20)
   a. Jelaskan 4 syarat terjadinya Deadlock
   b. Apa perbedaan Deadlock Prevention dan Deadlock Avoidance?
   c. Keadaan dari sebuah sistem adalah sebagai berikut:

	Allocation	*Max*	*Available*
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

Dengan menggunakan algoritma Bankers, tunjukkan apakah sistem berada dalam keadaan (state) Safe atau tidak?

a) Syarat deadlock

Mutual Exclusion (Mutex)	Tidak boleh ada proses lain yang mengeksekusi hal yang sama
Hold and wait	Suatu proses memegang paling sedikit 1 resource sambil nunggu resource lain yang sedang dipegang suatu proses
No preemption	Resource yang ingin digunakan harus nunggu suatu proses selesai, tidak boleh diserobot oleh proses lain
Circular wait	Intinya Pn-1 nunggu resource Pn, P0 nunggu P1, P1 nunggu P2, trus P2 nunggu P0 (muter trs)

b) Prevention itu memastikan suatu sistem tidak akan memenuhi at least satu syarat deadlock. Kalau Avoidance memastikan suatu sistem tidak akan mencapai unsafe state sehingga tidak akan mengalami deadlock.

c) Need :

P0 : 0 0 0 0

P1 : 0 7 5 0

P2 : 1 0 0 2

P3 : 0 0 2 0

P4 : 0 6 4 2

P0 -> 1 5 3 2

P1 -> 2 5 3 2

P2 -> 3 8 8 6

P3 -> 4 5 1 8

P4 -> 4 5 3 2 so safe state

6. **Sinkronisasi** (bobot 15)

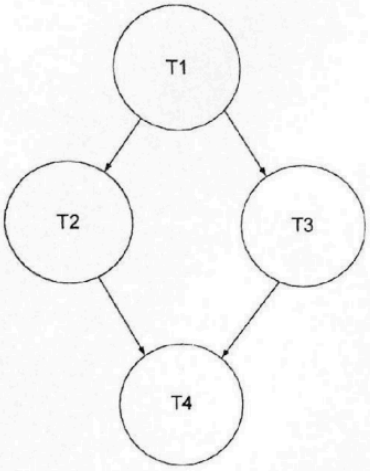


Diagram di atas menggambarkan hubungan antar thread, dimana panah dari sebuah thread (Tx) ke thread lain (Ty) menunjukkan bahwa Tx harus selesai dijalankan sebelum Ty mulai dijalankan. Tuliskan kode yang menjamin relasi antar thread di atas dengan menggunakan semaphore. Tuliskan pula nilai awal semaphore yang anda gunakan.

```
// definisi semaphore dan inisialisasi
...

void T1(void)     void T2(void)     void T3(void)     void T4(void)
{                 {                 {                 {
...               ...               ...               ...
}                 }                 }                 }
```

Semaphore tu punya 2, wait() dan signal()

Kayanya nilai awal semaphore nya diset 0(?) as tanda dia gabole jalan dulu sblm thread sblmnya slese

// Deklarasi awal inisiasi semua = 0

semaphore T1 = 0

semaphore T2 = 0

semaphore T3 = 0

void *T1(void *args) {

    P1

```
        signal(T1)
}

void *T2(void *args) {
        wait(T1)
        P2
        signal(T2)
}

void *T3(void *args) {
        wait(T1)
        P3
        signal(T3)
}

void *T4(void *args) {
        wait(T2)
        wait(T3)
        P4
}
```

UTS 2014/2015

1. Introduction and OS Structure (bobot 15)
    a. Sebutkan 3 tujuan utama dari eksistensi sistem operasi
    b. Jelaskan hubungan Multiprogramming, Multitasking, dan Process.
    c. Apa bedanya System Call dengan System Program?

a) Tujuan OS
   - Memudahkan user dalam menggunakan komputer (ease of use)
   - Manajemen & koordinasi antar hardware (resource utilization).
   - Mengalokasikan sebagian hardware untuk setiap proses (resource allocator).
   - Menjalankan program user/sistem (control program)

b) Multiprogamming : menjalankan bbrp program dlm satu waktu dgn tujuan memaksimalkan kinerja CPU
   Multitasking : menjalankan bbrp program dgn switching frekuensi tinggi shg response time ke user mnjd cepat
   Proses : program yang dieksekusi secara sekuensial
   Multiprogramming dan multitasking merupakan proses namun dengan mekanisme yg beda

c) System call : interface terhadap service pada OS yang dilapisi oleh API
   System program : interface terhadap system call
   System program = printf() tp sbnrnnya printf ini tuh manggil fungsi write() dmn write() adalah system call yg dimiliki C

2. Process Scheduling (Bobot 20)

Misalkan ada 7 proses P1, P2, ..., P7 dengan arrival times dan CPU Burst time sebagai berikut:

Process	P1	P2	P3	P4	P5	P6	P7
Arrival Time	3	5	6	8	10	16	17
CPU Burst Time	3	2	1	4	2	6	8

Dengan masing-masing algoritma FCFS, Round Robin (quantum: 1 unit waktu), lakukan hal berikut:

a. Gambarkan diagram yang mengilustrasikan eksekusi proses-proses tersebut.
b. Hitung Average Waiting Time,
c. Hitung Average Turn Around Time,
d. Apakah CPU sempat mengalami idle? Kapan?

3. Process Creation (Bobot 15)

Berapakah nilai variable value, tepat setelah masing-masing kode Line A, Line B, dan Line C dieksekusi?

```
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int value = 2015;
void *fungsi(void *param);  /*thread*/

int main(int argc, const char *argv[]) {
   pid_t pid;
   pthread_t tid;
   pthread_attr_t attr;

   pid = fork();
   if (pid==0) {
         pthread_attr_init(&attr);
         pthread_create(&tid, &attr, fungsi, NULL);
         pthread_join(tid, NULL);
         sleep(5);

         value += -1;
         printf("Child %d: value = %d \n", pid, value); /*LINE  A*/
   }
```



```
   else if (pid > 0) {
         value += -1;
         wait(NULL);
         printf("PARENT %d: value = %d \n", pid, value); /*LINE  B*/
   }
}

void *fungsi(void *param) {
   value += 1;
   printf("ini Thread Fungsi %d\n", value); /*LINE  C*/

   pthread_exit(0);
}
```

Line A → 2015 krn dia jalanin fungsi thread dulu jd dr 2015 → 2016 trus jadi → 2015

Line B → 2014 krn dr 2015 → 2014

Line C → dr 2015 → 2016

4. Synchronization (Bobot 25)

   Buatlah implementasi barrier dengan menggunakan mutex+condition variable atau semaphore. Fungsi barrier yang akan diimplementasikan:

   ```
   barrier (int nprocess)
   ```

   di mana thread yang memanggil barrier akan terblok sampai sejumlah n thread telah memanggil fungsi barrier ini, dan setelah terdapat n thread yang memanggil fungsi barrier, maka seluruh thread yang sedang terblok pada barrier ini akan melanjutkan eksekusinya. Diasumsikan semua thread akan memanggil fungsi barrier dengan nilai parameter nproses yang sama.

5. Interprocess Communication (Bobot 15)
   a. Jelaskan kelebihan dan kekurangan antara menggunakan shared memory dan message passing untuk komunikasi antarproses
   b. Jelaskan perbedaan antara komunikasi antarproses menggunakan pipe dan fifo pada Linux.

a) Shared memory lebih cepat karena pakai kernel di awal doang, bisa saja ada 2 proses yang sedang mengakses memori dalam waktu yang bersamaan. Karena memory yg dipakai ini sama, maka bisa proses bisa masuk critical section secara bersamaan karena akses memori yang sama tadi. Keadaan ini tidak memenuhi syarat mutex.
   Message passing memakai memori besar karena memproses satu-satu. Dia langsung kasih tau proses mana yang akan dieksekusi sehingga tidak akan terjadi penggunaan item yang bersamaan (memenuhi mutex). Memungkinkan terjadinya duplicate data.

b)