# Pengantar Sistem Operasi

CSIM602155 2024/2025

## PR 2

**Deadline**: **8 Oktober 2024, Pukul 23.55**

**Format pengumpulan**: File yang dikumpulkan berbentuk **PDF**

**Format penamaan file**: **PR2_[Kode Asdos]_[Kelas]_[Nama]_[NPM].pdf**

Contoh penamaan file: PR2_XXX_B_AhmadBahri_2306812345.pdf

Ketentuan:

- Tugas dikerjakan dengan **tulis tangan** (penggunaan *digital pen* diperbolehkan) dengan cara pengerjaan sedetail mungkin.
- Scan hasil pekerjaan dengan scanner atau aplikasi telepon pintar (misal: camscanner).
- Mahasiswa yang terlambat akan dikenakan pengurangan nilai sebanyak 25% per 2 jam keterlambatan dihitung dari deadline dengan maksimal keterlambatan 6 jam. Contoh deadline jam 23.55, maka pengumpulan pada jam 23.56 sampai 01.55 mendapatkan pengurangan sebanyak 25%.
- Mahasiswa yang tidak mengikuti format penamaan akan mendapatkan pengurangan sebanyak 3 poin dan yang tidak mengikuti format file akan mendapatkan pengurangan sebanyak 5 poin.

## SOAL

### NOMOR 1 Sinkronisasi Thread

```
1  void* thread1() {
2      sem_wait(&sem[1]);
3      printf("A ");
4      sem_post(&sem[3]);
5      sem_post(&sem[8]);
6  }
7
8  void* thread2() {
9      sem_wait(&sem[2]);
10     printf("B ");
11     sem_post(&sem[4]);
12     sem_post(&sem[1]);
13 }
14
15 void* thread3() {
16     printf("C ");
17     sem_post(&sem[5]);
18 }
19
20 void* thread4() {
21     sem_wait(&sem[3]);
22     printf("D ");
23     sem_wait(&sem[4]);
24     printf("E ");
25     sem_post(&sem[6]);
26 }
27
28 void* thread5() {
29     sem_wait(&sem[5]);
30     printf("F ");
31     sem_post(&sem[2]);
32 }
33
34 void* thread6() {
35     sem_wait(&sem[6]);
36     printf("G ");
37     sem_post(&sem[7]);
38 }
39
40 void* thread7() {
41     sem_wait(&sem[7]);
42     printf("H ");
43 }
```

Sebuah program mempunyai tujuh (7) semaphore sem[1..7], 7 thread

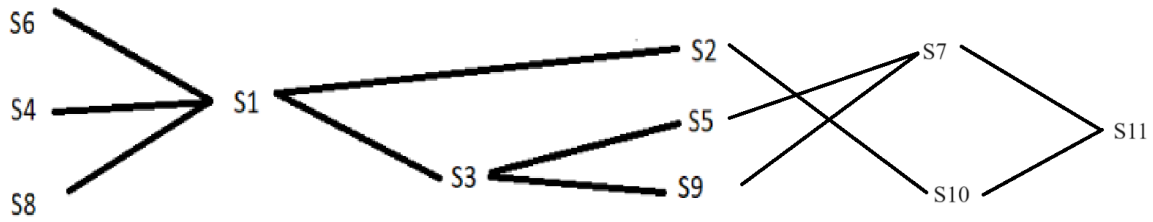
(Thread1..7) dan 7 fungsi thread1()...thread7(). Setiap semaphore diinisialisasi dengan nilai 0 dan lima thread dijalankan secara berurutan, mulai dari Thread 1 mengeksekusi fungsi thread1(), Thread 2 mengeksekusi thread2() dan seterusnya hingga Thread7 mengesekusi thread7(). Setelah semua thread dijalankan, **tentukan output program !**

**NOMOR 2 Sinkronisasi Thread**

**Diketahui:**

i. Ada 4 buah thread T1 (S1, S2), T2 (S3, S4,S11), T3 (S5, S6,S10), T4 (S7, S8, S9), di mana Tn(Sx) = Statement x dari Thread Ti, Ti = Id Thread

ii. Statement untuk setiap thread dieksekusi berdasarkan urutan pada grafik dibawah ini.




Eksekusi S6, S4, S8 dilakukan secara simultan dilanjutkan S1 hingga diakhiri dengan S11.

Berdasarkan urutan eksekusi diatas, inisialisasi dan implementasikan variabel semaphore sem[i], fungsi wait(sem[i]), dan signal(sem[i]) pada serangkaian blok kosong dibawah ini. Index ipada sem[i] dimulai dari sem[0], sem[1], …, sem[n].

| // inisiasi variabel semaphore disini : |
|---|
| |

Implementasikan kode program yang berisikan statement Sx, fungsi wait(sem[i]) dan signal(sem[i]) pada blok kosong setiap thread dibawah ini.

| T1 | T2 | T3 | T4 |
|---|---|---|---|
| | | | |

**Nomor 3 CPU Scheduling**

Jika terdapat lima proses sebagai berikut :

| Process ID | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 25 | 2 |
| P2 | 5 | 35 | 3 |
| P3 | 10 | 15 | 1 |
| P4 | 15 | 20 | 4 |
| P5 | 30 | 25 | 3 |

Algoritma penjadwalan proses yang digunakan :

1. **First-Come First-Served (FCFS)**
2. **Shortest Job First (SJF) (Non-preemptive)**
3. **Shortest Remaining Time First (SRTF) (nama lain: SJF-Preemptive).** Jika terdapat lebih dari satu proses yang mempunyai BURST-TIME yang sama, maka yang didahulukan adalah proses dengan ARRIVAL TIME TERBESAR.
4. **Priority (Preemptive).** Prioritas paling tinggi ditandai dengan nomor prioritas paling besar. Jika terdapat lebih dari satu proses yang mempunyai prioritas sama, maka yang didahulukan adalah proses dengan ARRIVAL TIME TERKECIL.
5. **Priority (Preemptive) dengan Round Robin (RR) (Time Quantum = 5)**
   - Proses yang didahulukan adalah proses dengan prioritas tertinggi, yaitu nomor prioritas paling besar
   - Round Robin hanya berlaku untuk proses-proses dengan nomor prioritas sama. Jika pada antrian terdapat lebih dari satu proses dengan prioritas sama, maka yang didahulukan untuk dieksekusi adalah proses dengan ARRIVAL TIME TERBESAR.

Untuk setiap algoritma penjadwalan buatlah :

a. Gantt chart
b. Total Turn-Around Time setiap proses
c. Total Waiting-Time setiap proses
d. Rata-rata Waiting-Time untuk seluruh proses

**Nomor 4 Banker's Algorithm**

Sebuah sistem mempunyai resource:

- A (11 instance)
- B (9 instance)
- C (15 instance)

State sistem saat ini adalah sebagai berikut:

| | Allocation | | | Max | | |
|---|---|---|---|---|---|---|
| | **A** | **B** | **C** | **A** | **B** | **C** |
| **T0** | 3 | 1 | 0 | 6 | 5 | 3 |
| **T1** | 4 | 3 | 1 | 10 | 9 | 6 |
| **T2** | 1 | 0 | 9 | 7 | 6 | 13 |
| **T3** | 0 | 1 | 2 | 5 | 6 | 5 |
| **T4** | 2 | 2 | 0 | 3 | 4 | 3 |

a. Tentukan Tabel Available!

| Available | | |
|---|---|---|
| **A** | **B** | **C** |
| | | |

b. Tentukan Tabel Need!

| | Need | | |
|---|---|---|---|
| | **A** | **B** | **C** |
| **T0** | | | |
| **T1** | | | |
| **T2** | | | |
| **T3** | | | |
| **T4** | | | |

c. Tentukan apakah sistem dalam kondisi deadlock atau safe! Jika safe, tentukan semua kemungkinan safe sequence-nya!