

# **Laporan Praktikum**

## **Sistem Operasi**

**Dosen pengampu : (*Iwan Lesmana, S.Kom., M.Kom.*)**

### **Modul 4**



**Nama : Muhammad Rizal Nurfirdaus**

**NIM : 20230810088**

**Kelas : TINFC – 2023 – 04**

**Teknik Informatika**

**Fakultas Ilmu Komputer**

**Universitas Kuningan**

## Pre test

1. Apa tujuan dari sinkronisasi proses dalam sistem operasi?

Tujuan utama dari sinkronisasi proses adalah untuk mengatur interaksi antar proses atau thread yang berjalan secara bersamaan (concurrent) agar tidak terjadi konflik atau kerusakan data. Sinkronisasi memastikan bahwa akses ke sumber daya bersama (seperti memori atau file) dilakukan dengan cara yang terkoordinasi, sehingga tidak ada proses yang mengaksesnya secara bersamaan dengan cara yang merusak integritas data. Dengan sinkronisasi, sistem operasi dapat memastikan bahwa proses berjalan secara aman, efisien, dan sesuai urutan yang diinginkan, menghindari kondisi balapan (race condition), dan menjaga konsistensi data.

2. Masalah apa yang bisa terjadi ketika beberapa thread mengakses data Bersama tanpa sinkronisasi?

- **Race Condition:** Kondisi di mana hasil eksekusi proses tergantung pada urutan eksekusi yang tidak dapat diprediksi. Hal ini dapat menyebabkan hasil yang tidak diinginkan atau inkonsistensi data.
- **Data Corruption:** Ketika dua thread menulis data pada lokasi yang sama secara bersamaan tanpa pengaturan, data tersebut dapat rusak karena satu thread menimpa data yang ditulis oleh thread lain.
- **Deadlock:** Proses atau thread saling menunggu satu sama lain untuk melepaskan sumber daya yang dibutuhkan, mengakibatkan semuanya terhenti dan sistem tidak dapat melanjutkan.
- **Starvation:** Beberapa thread tidak mendapatkan akses ke sumber daya yang dibutuhkan karena thread lain selalu mendapatkan prioritas lebih tinggi.

Itulah beberapa masalah yang bisa terjadi ketika beberapa thread mengakses data Bersama tanpa sinkronisasi.

## Praktikum

### 1.

program CriticalRegion

```
var g integer
```

```
sub thread1 as thread
```

```
    writeln("In thread1")
```

```
    g = 0
```

```
    for n = 1 to 20
```

```
        g = g + 1
```

```
    next
```

```
        writeln("thread1 g = ", g)
        writeln("Exiting thread1")
end sub
```

```
sub thread2 as thread
    writeln("In thread2")
    g = 0
    for n = 1 to 12
        g = g + 1
    next
    writeln("thread2 g = ", g)
    writeln("Exiting thread2")
end sub
```

```
writeln("In main")
```

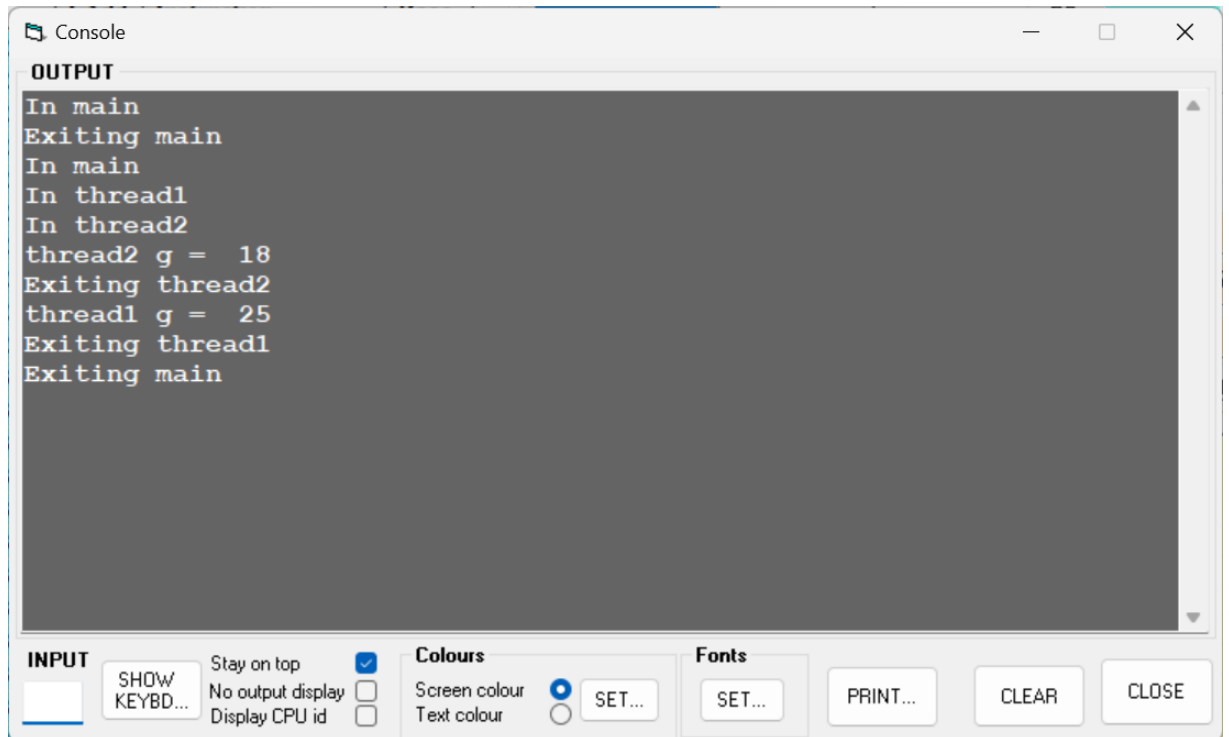
```
call thread1
```

```
call thread2
```

```
wait
```

```
writeln("Exiting main")
```

```
end
```



#### Analisis :

Program yang menggunakan threading di mana terdapat tiga bagian utama: main, thread1, dan thread2. Eksekusi dimulai dari main, lalu thread1 dan thread2 dijalankan secara paralel. Variabel g dimodifikasi oleh kedua thread dengan nilai 18 pada thread2 dan 25 pada thread1. Program menunjukkan bahwa thread selesai dieksekusi secara terpisah dengan mencetak pesan seperti "Exiting thread2" dan "Exiting thread1". Hal ini mencerminkan kerja simultan antar-thread dalam program.

2.

program CriticalRegion

var g integer

sub thread1 as thread synchronise

writeln("In thread1")

g = 0

for n = 1 to 20

g = g + 1

next

writeln("thread1 g = ", g)

writeln("Exiting thread1")

end sub

sub thread2 as thread synchronise

writeln("In thread2")

g = 0

for n = 1 to 12

g = g + 1

next

writeln("thread2 g = ", g)

writeln("Exiting thread2")

end sub

writeln("In main")

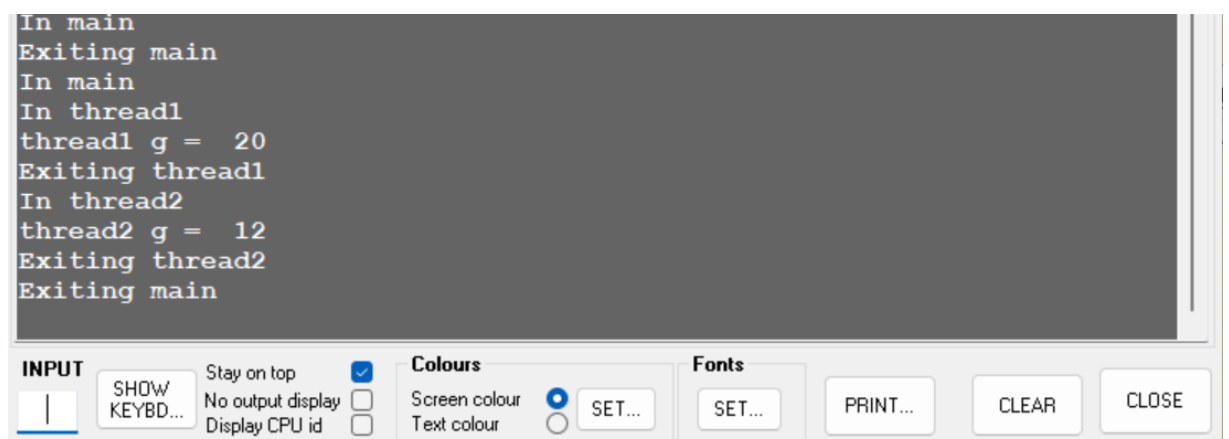
call thread1

call thread2

wait

writeln("Exiting main")

end



```
In main
Exiting main
In main
In thread1
thread1 g = 20
Exiting thread1
In thread2
thread2 g = 12
Exiting thread2
Exiting main
```

**INPUT**   ☐ Stay on top ☒ No output display ☐ Display CPU id

**Colours** Screen colour ☒ SET... Text colour ☐ SET...

**Fonts** SET...

**Analisis :**

Hasil output menunjukkan implementasi multithreading, di mana fungsi utama (main) memulai dua *thread* (thread1 dan thread2) yang berjalan secara bergantian. Thread1 memodifikasi nilai variabel *g* menjadi 20 dan thread2 mengubahnya menjadi 12, dengan masing-masing *thread* menyelesaikan eksekusinya tanpa konflik atau gangguan satu sama lain.

3.

program CriticalRegion

var g integer

sub thread1 as thread

    writeln("In thread1")

    enter

    g = 0

    for n = 1 to 20

        g = g + 1

    next

    writeln("thread1 g = ", g)

    leave

    writeln("Exiting thread1")

end sub

sub thread2 as thread

    writeln("In thread2")

    enter

    g = 0

    for n = 1 to 12

        g = g + 1

next

    writeln("thread2 g = ", g)

    leave

    writeln("Exiting thread2")

end sub

writeln("In main")

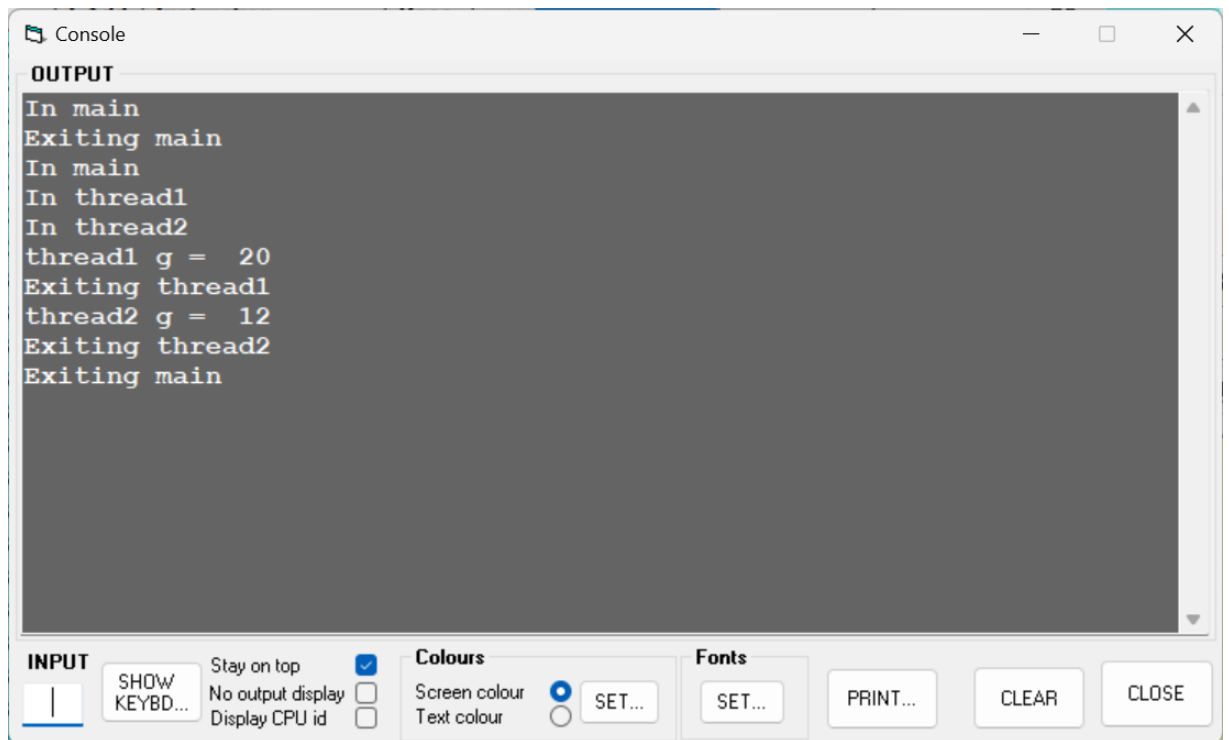
call thread1

call thread2

wait

writeln("Exiting main")

end



```
Console
OUTPUT
In main
Exiting main
In main
In thread1
In thread2
thread1 g = 20
Exiting thread1
thread2 g = 12
Exiting thread2
Exiting main

INPUT
SHOW KEYBD...
Stay on top [checked]
No output display [unchecked]
Display CPU id [unchecked]
Colours
Screen colour [selected]
Text colour [unselected]
SET...
Fonts
SET...
PRINT...
CLEAR
CLOSE
```

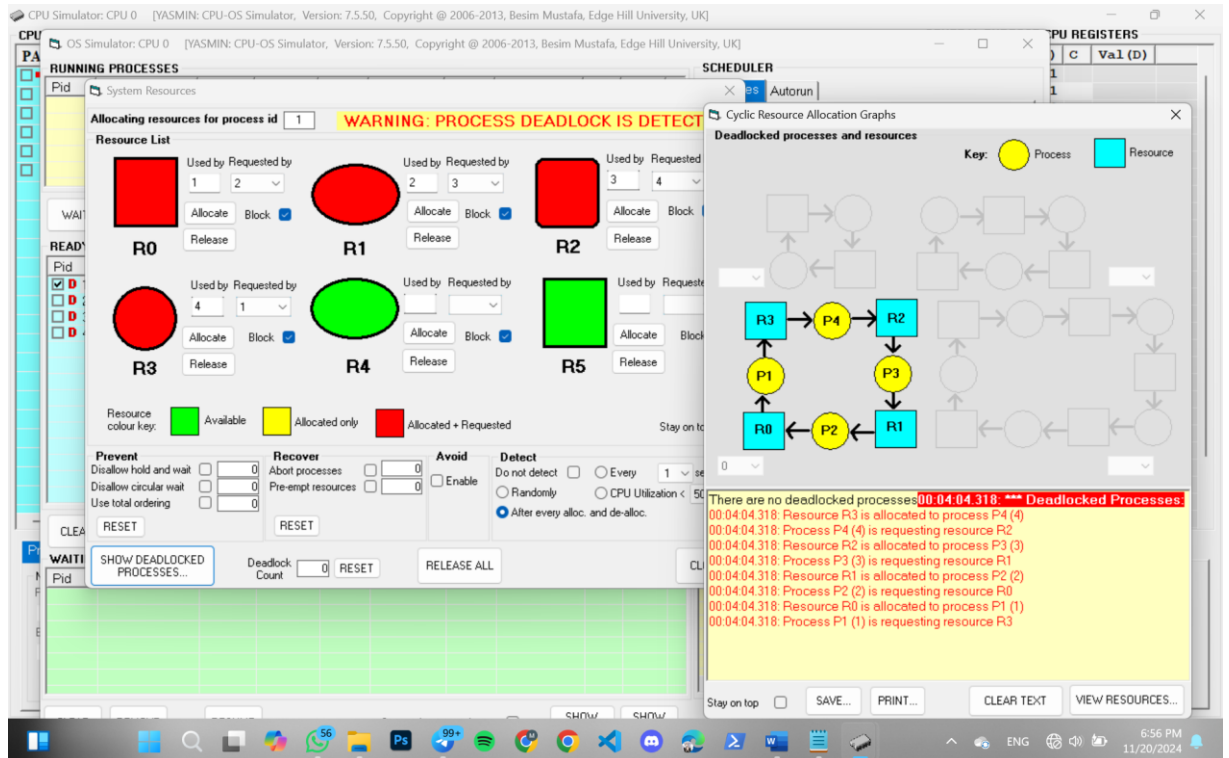
### Analisis :

Program menunjukkan implementasi multithreading, di mana fungsi utama memulai dua *thread* (thread1 dan thread2) yang berjalan secara paralel. Kedua *thread* memodifikasi variabel *g* secara independen (thread1: *g* = 20, thread2: *g* = 12) dan menyelesaikan eksekusi tanpa konflik, karena tidak ada tumpang tindih yang mengganggu hasil akhir.

### Langkah Praktikum Deadlock

Program Deadlock

```
while true
    n = 1
wend
end
```



## Post Test

1. Jelaskan peran perangkat keras dalam mendukung mekanisme sinkronisasi  
Perangkat keras memainkan peran penting dalam mendukung mekanisme sinkronisasi dengan menyediakan instruksi dan fitur untuk mengatur akses yang terkoordinasi ke sumber daya bersama di lingkungan multi-thread atau multi-proses.
2. Mengapa penting untuk menerapkan sinkronisasi saat thread mengakses data global?  
Penting untuk menerapkan sinkronisasi saat thread mengakses data global karena data global seringkali digunakan bersama oleh banyak thread, dan tanpa kontrol yang tepat, data ini bisa rusak atau menghasilkan hasil yang tidak konsisten.  
Agar tidak terjadi masalah seperti pertanyaan pretest soal kedua diatas

## Tugas

1. Beri penjelasan dan Kesimpulan terhadap hasil simulasi deadlock yang telah dilakukan diatas!

Deadlock terjadi akibat kondisi klasik yang mencakup empat karakteristik:

- **Mutual Exclusion:** Sumber daya hanya dapat digunakan oleh satu proses dalam satu waktu.
- **Hold and Wait:** Proses memegang sumber daya yang sudah dialokasikan sambil menunggu sumber daya lain.



- **No Preemption:** Sumber daya tidak dapat direbut paksa dari proses yang sedang memegangnya.
- **Circular Wait:** Ada siklus dalam antrian proses dan sumber daya yang saling menunggu.