1. **ESSAY(100%)**
2. **Computer and O/S Overview**
3. Jelaskan bagaimana interrupt meningkatkan efisiensi prosesor, sedangkan komputer saat ini sudah memiliki clock speed yang tinggi dan memory yang besar?

**Jawaban:**

**Memori adalah ruang kerja** sedangkan **prosesor adalah pekerja** dan **CPU adalah "otak" dari sistem komputer** yang bertanggung jawab **untuk pemrosesan seperti Input, Eksekusi, dan Output**. **Jadi tanpa adanya interupsi, CPU dapat membuang banyak waktu menunggu perangkat eksternal seperti clock speed untuk mencocokkan kecepatan dan CPU-nya** sehingga mengurangi efisiensi dari CPU. **Oleh karena itu, interupsi diperlukan untuk megurangi keterbatasan ini**.

**REFERENCES :**

<https://ieeexplore.ieee.org/document/386984>

1. Beberapa Sistem Operasi contohnya LINUX, tidak membutuhkan Hardware Driver. Jelaskan alasannya. Apa yang terjadi jika kernel tidak mengenali sebuah hardware yang terpasang?

**Jawaban:**

LINUX tidak membutuhkan Hardware Driver **dikarenakan sudah disediakan pada saat Installasi yang disebut sebagai Firmware** dan **di dalam Firmware terdapat banyak Kernel Modules disana**.

**Jika kernel tidak mengenali hardware yang diinstal maka perlu menginstal driver vendor yang kompatibel dengan sistem Linux**. Hal tersebut bisa terjadi ketika memang hardware yang diinstal belum ada di Kernel Modules Linux nya.

**REFERENCES :**

<https://www.researchgate.net/publication/316511108_Comparison_of_the_Linux_and_Windows_Device_Driver_Architectures>

1. Pada servis yang disediakan oleh Sistem Operasi, jelaskan yang dimaksud dengan pernyataan “controlled access to files” dan berikan contoh.

**Jawaban:**

**“controlled access to files”** di Sistem Operasi **berkaitan dengan izin untuk file dan direktori**. **Izin dapat diatur untuk memberikan atau menolak akses ke file dan direktori tertentu**.

**Contohnya ketika izin diberikan**, maka dapat mengakses dan melakukan fungsi pada file atau direktori seperti “Read”, “Write”, dan “Execute”.

**REFERENCES :**

<https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/access-control>

1. **Process Description and Control**
2. Suatu proses dapat terganggu eksekusinya oleh interrupt maupun trap. Jelaskan perbedaan antara interrupt dan trap dan berikan contoh untuk interrupt dan trap.

**Jawaban:**

|  |  |
| --- | --- |
| **Trap** | **Interrupt** |
| Sinyal diberikan oleh User meminta sistem operasi untuk segera mengerjakan suatu tindakan | Sinyal diberikan oleh perangkat yang menunjukkan suatu peristiwa yang memerlukan perhatian segera |
| Informasi tindakan ditransfer dari "Controller" ke "Handler" | Perhatian akan memicu prosesor untuk menjalankan "Routine" penanganan interupsi yang sesuai. |
| Dikenal juga sebagai "Software Interrupt" | Dikenal juga sebagai "Hardware Interrupt" |

**Contoh Trap:**

Ketika pengguna membagi bilangan bulat dengan 0 maka program akan membuat "Trap/Exception" di sistem.

**Contoh Interrupt:**

Instruksi seperti pengguna menekan tombol pada keyboard oleh pengguna yang membuat perangkat keras mengganggu proses yang sedang berlangsung seperti menghentikan atau menjeda sistem.

**REFERENCES :**

<https://www.geeksforgeeks.org/difference-between-interrupt-and-exception/>

1. Jelaskan bagaimana system call fork() dapat membuat proses-proses baru yang dapat berjalan secara parallel. Berikan contoh program dengan menggunakan system call fork() dimana parent process membuat proses baru dan membentuk suatu hiraki seperti tergambar dibawah ini:

Diagram

Description automatically generated

**Jawaban:**

**Proses system call fork() dapat membuat proses-proses baru secara parallel dengan menjalankan PID child process dan parent process dengan wait(NULL)**.

**Contoh Program:**

#include <stdio.h>

#include <unistd.h>

int main **(){**

pid\_t pid1 **=** fork**();**

**if** **(**pid1 **==** 0**){**

wait**(NULL);**

printf**(**"[Child] process ID %d, whose [Parent] process ID is %d.\n"**,** getpid**(),** getppid**());**

**}else** **if(**pid1 **>** 0**){**

wait**(NULL);**

pid\_t pid3 **=** fork**();**

**if(**pid3 **==** 0**){**

printf**(**"[Child] process ID %d, whose [Parent] process ID is %d.\n"**,** getpid**(),** getppid**());**

wait**(NULL);**

pid\_t pid4 **=** fork**();**

**if(**pid4 **==** 0**){**

printf**(**"[Child] process ID %d, whose [Parent] process ID is %d.\n"**,** getpid**(),** getppid**());**

wait**(NULL);**

**}**

**}**

**}**

**return** 0**;**

**}**

**CODE OUTPUT :**

Text

Description automatically generated

**REFERENCES :**

<https://www.geeksforgeeks.org/fork-system-call/>

1. **Multiprocessor, Multicore and Embedded System**
2. Kenapa jam tangan pintar, mobil otonom, sensor parkir dikategorikan sebagai deeply embedded system? Kendala apa saja yang timbul dalam implementasi deeply embedded system (min 3)?

**Jawaban:**

**Deeply Embedded Systems adalah** perangkat tunggal yang digunakan untuk mendeteksi sesuatu di lingkungan, melakukan pemrosesan tingkat dasar, dan kemudian mengeluarkan hasilnya.

**Jam tangan pintar, mobil otonom, sensor parkir dikategorikan sebagai Deeply Embedded System karena** mereka bekerja secara real time dan tidak ada transfer antara cloud atau apa pun, ini bekerja langsung di compiler alatnya di mana semua pemrosesan dan handling harus dilakukan langsung di sana.

**Kendala yang timbul dalam implementasi Deeply Embedded System:**

1. **Stabilitas**

Perangkat lunak Embedded harus memiliki perilaku yang seragam dalam semua keadaan dan dapat beroperasi secara tahan lama.

1. **Batasan Desain**

Embedded System dikemas untuk lebih banyak daya pemrosesan dan masa pakai baterai yang lebih lama ke dalam ruang yang lebih kecil.

1. **Keamanan**

Embedded System menggunakan konsep Internet of Things dimana terdapat risiko seperti malware atau ransomware, dan pencurian data meliputi kekayaan intelektual yang bersifat rahasia bocor.

**REFERENCES :**

<https://ieeexplore.ieee.org/document/6376832>

1. Jelaskan permasalahan-permsalahan yang ada dalam penjadwalan proses pada multiprocessor system (min 2).

**Jawaban:**

1. **Konsistensi Data**

Ketika beberapa prosesor mengakses data yang sama pada saat yang sama maka ada kemungkinan terjadinya ketidak-konsistenan data sehingga harus menggunakan beberapa ***scenario Locking***.

1. **Pemberitahuan Perubahan**

**Sumber data digunakan bersama yang disimpan di beberapa cache local** seperti ada dua orang memiliki salinan memori yang di-cache dan satu orang mengubah blok memori dan orang lainnya dapat dibiarkan dengan cache yang tidak valid tanpa pemberitahuan perubahan **sehingga diselesaikan dengan mempertahankan tampilan data yang saling berkaitan satu sama lainnya**.

**REFERENCES :**

<https://www.geeksforgeeks.org/multiple-processor-scheduling-in-operating-system/>

1. **Threads**
2. Dalam situasi apa single thread memberikan keuntungan lebih dibandingkan muli-thread.

**Jawaban:**

**Multiple threads memerlukan sinkronisasi**, artinya beralih dari 1 thread ke thread lain sehingga membuat menjadi lebih lambat.

**Dari sisi penggunaan resources daya**, itu juga membutuhkan pemahaman tentang perangkat keras untuk mengetahui kapan untuk menjalankan banyak Thread.

**Jadi, Situasi Single Thread memberi keuntungan lebih ketika** data yang ingin diproses tidak dalam jumlah yang cukup besar dikarenakan jika data sangat besar dan dijalankan Single threaded, maka akan sama saja seperti multi thread.

**REFERENCES :**

<https://www.geeksforgeeks.org/multithreading-in-operating-system/>

1. Jelaskan apa perbedaan dan hubungan antara proses dan multi-thread, termasuk hubungan state, address, data.

**Jawaban:**

**Multithreading** menjalankan beberapa Thread sekaligus yang dapat berjalan secara independen dari Thread utama tapi masih bagian dari proses yang sama.

**State adalah** data yang disimpan dalam memori dimana hanya ada satu salinan status objek dalam memori pada satu waktu.

**Processes** memiliki state yang menyatakan setiap instruksi dalam proses, dan proses memiliki alamat yang unik karena alamat tersebut merupakan lokasi dari proses tersebut dan memiliki state yang unik karena state dari proses adalah kumpulan instruksi yang ada di dalam proses.

**Misalnya, ketika menjalankan perintah java**, aplikasi java dijalankan dalam suatu proses. Dalam hal ini, Thread utama adalah Thread eksekusi untuk aplikasi java.

**Address adalah** lokasi di memori tempat data suatu objek disimpan sedangkan Data adalah objek yang dimanipulasi oleh thread dimana data sebenarnya disimpan dalam register.

**Dapat dianggap data dan alamat sebagai hal yang sama** dimana Alamat hanyalah variabel yang digunakan untuk mengakses Data yang sebenarnya disimpan di lokasi memori yang ditunjuk oleh alamat tersebut.

**REFERENCES :**

<https://www.geeksforgeeks.org/difference-between-multiprocessing-and-multithreading/>

<https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/>

1. **Process Scheduling**

Dalam suatu sistem operasi, terdapat 4 proses yang sedang berjalan dengan rincian sbb:

Table

Description automatically generated

Anda ingin memastikan, penjadwalan mana yang lebih baik, diantara penjadwalan ***First Come First Serve, Shortest Job First (preemptive)*** dan ***Round Robin*** dengan ***time slice = 4***. Simulasikan ke tiga algoritma tersebut, hitung rata-rata turnaround time dan waiting time. Lakukan perbandingan dan tentukan algoritma yang paling efisien. Dalam melakukan perbandingan, anda harus juga melihat dari sisi ***CPU Utilization***.

***CPU Utilization*** dapat dihitung berdasarkan formula:

Dimana adalah turnaround time

adalah rata-rata service time

adalah ***CPU Utilization***

**Jawaban:**

1. **Menggunakan First Come First Serve Scheduling**

**Running Queue (Gantt Chart)**

A picture containing text, clock, watch

Description automatically generated

**Turn Around Time = Completion Time – Arrival Time**

**Waiting Time = Turn Around Time – Burst Time**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Arrival Time** | **Completion Time** | **Turn Around Time** | **Waiting Time** |
| A | 12 | 0 | 12 | 12 | 0 |
| B | 1 | 2 | 13 | 11 | 10 |
| C | 4 | 7 | 17 | 10 | 6 |
| D | 3 | 10 | 20 | 10 | 7 |

**Rata-rata Turn Around Time = = = 10,75**

**Rata-rata Waiting Time = = = 5,75**

1. **Menggunakan Shortest Job First Scheduling (Preemptive)**

**Running Queue (Gantt Chart)**

A picture containing text

Description automatically generated

**Turn Around Time = Completion Time – Arrival Time**

**Waiting Time = Total Waiting Time – Total Process executed – Arrival Time**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Arrival Time** | **Completion Time** | **Turn Around Time** | **Waiting Time** |
| A | 12 | 0 | 20 | 20 | 8 |
| B | 1 | 2 | 3 | 1 | 0 |
| C | 4 | 7 | 11 | 4 | 0 |
| D | 3 | 10 | 14 | 4 | 1 |

**Rata-rata Turn Around Time = = = 7,25**

**Rata-rata Waiting Time = = = 2,25**

1. **Menggunakan Round Robin Scheduling dengan Time Slice = 4**

**Ready Queue (Queue Paling depan saya anggap di Ujung Kiri)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B | A | C | A | B |

**Running Queue (Gantt Chart)**

A white paper with black writing

Description automatically generated with low confidence

**Turn Around Time = Completion Time – Arrival Time**

**Waiting Time = Turn Around Time – Burst Time**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process** | **Burst Time** | **Arrival Time** | **Completion Time** | **Turn Around Time** | **Waiting Time** |
| A | 12 | 0 | 17 | 17 | 5 |
| B | 1 | 2 | 5 | 3 | 2 |
| C | 4 | 7 | 13 | 6 | 2 |
| D | 3 | 10 | 20 | 10 | 7 |

**Rata-rata Turn Around Time = = = 9**

**Rata-rata Waiting Time = = = 4**

**Tabel Perbandingan Hasil Akhir Average Turn Around Time, Average Waiting Time, dan CPU Utilization**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scheduling** | **Average Turn Around Time** | **Average Waiting Time** | **CPU Utilization** |
| First Come First Serve | 10,75 | 5,75 | 0,465 |
| Short Job First (Preemptive) | 7,25 | 2,25 | 0,689 |
| Round Robin (Time Slice = 4) | 9 | 4 | 0,555 |
| **Overall Average** | 9 | 4 | 0,569666667 |

**Jadi, dari data diatas, penjadwalan yang lebih baik menggunakan Round Robin Scheduling dengan Time Slice = 4** **dikarenakan dari sisi Average Turn Around Time tidak melebihi rata rata total** dan juga **dari sisi Average Waiting Time tidak melebihi rata rata total**. **Dari Sisi CPU Utilization juga lebih stabil dan lebih kecil penggunaan memorinya** **sehingga dipilihkan yang lebih baik menggunakan Round Robin Scheduling dengan Time Slice = 4**.

1. **Synchronization and Deadlock**
2. Sebutkan dan jelaskan Common concurrency mechanism (min. 3) dan berikan contoh penggunaannya dalam bentuk pseudo-code atau program code.
3. **Producer Consumer Problem**

**Pernyataan Producer Consumer Problem:**

**Buffer memiliki ukuran tetap**. **Produsen dapat menghasilkan suatu barang dan dapat menempatkannya di penyangga**. Demikian pula, **konsumen dapat memilih item dari buffer dan dapat mengkonsumsinya**. **Harus mengatur kondisi bahwa ketika produsen menempatkan item di buffer**, maka **konsumen TIDAK boleh mengonsumsi item apa pun pada waktu yang sama**.

**Contoh Penggunaannya:**

**Untuk mengatasi masalah ini, diperlukan DUA Semaphore:**

* Full
* Semaphore ini untuk melacak jumlah item dalam buffer.
* Empty
* Semaphore ini untuk melacak jumlah slot kosong di buffer.

**Dan juga diperlukan 2 Operasi Berikut:**

* wait()
* Untuk **MENGURANGI value Semaphore Sebanyak 1**.
* signal()
* Untuk **MENAMBAHKAN value Semaphore Sebanyak 1**.

**Langkah 1: Inisialisasi Semaphores:**

mutex = 1

Full = 0 // Semua slot di buffer tidak diisi. Jadi, slot penuh sama dengan 0

Empty = n // Awalnya, Semua slot kosong.

**Langkah 2: Solusi untuk Producer:**

do{

// menghasilkan sebuah item

wait (Empty);

wait (mutex);

// tempatkan di buffer

signal(mutex);

signal (Full);

} while (true);

**Penjelasan:**

* Ketika produsen memproduksi satu item maka satu slot akan terisi sekarang, sehingga nilai “kosong” dikurangi 1. Untuk mencegah konsumen mengakses buffer, nilai mutex juga dikurangi.
* Sekarang, produsen telah menempatkan item di buffer.
* Jadi, nilai "penuh" dinaikkan 1.
* Begitu juga nilai mutex juga bertambah 1, hal ini dikarenakan produsen telah menyelesaikan tugasnya dan sekarang konsumen dapat mengakses buffer tersebut.

**Langkah 3: Solusi untuk Consumer:**

do{

wait (Full);

wait (mutex);

// hapus item dari buffer

signal (mutex);

signal (Empty);

// mengkonsumsi item

} while (true);

**Penjelasan:**

* Saat konsumen menghapus item dari buffer, maka nilai "full" dikurangi 1. Demikian pula, nilai mutex juga berkurang sehingga produsen tidak dapat mengakses buffer saat ini.
* Sekarang konsumen sudah mengkonsumsi barang tersebut.
* Jadi, nilai "kosong" dinaikkan 1.
* Demikian pula, Nilai mutex juga dinaikkan sehingga produsen dapat mengakses buffer sekarang.

1. **Readers Writers Problem**

**Pernyataan Readers Writers Problem:**

**Terdapat suatu file yang akan dibagikan ke banyak orang**. **Jika penulis mencoba mengedit file, tidak ada pembaca yang boleh membaca pada saat yang sama, jika tidak, perubahan tidak akan terjadi**. Demikian pula, **Jika pembaca sedang membaca file, penulis tidak boleh menulis atau mengedit file pada saat yang sama, jika tidak, perubahan tidak akan terjadi**. **Pembaca hanya dapat membaca dan tidak dapat mengedit file**.

**Contoh Penggunaannya:**

**Untuk mengatasi masalah ini, diperlukan DUA Semaphore:**

* Full
* Semaphore ini untuk melacak jumlah item dalam buffer.
* Empty
* Semaphore ini untuk melacak jumlah slot kosong di buffer.

**Dan juga diperlukan 2 Operasi Berikut:**

* wait()
* Untuk **MENGURANGI value Semaphore Sebanyak 1**.
* signal()
* Untuk **MENAMBAHKAN value Semaphore Sebanyak 1**.

**Langkah 1: Inisialisasi Semaphores:**

mutex = 1 // sinyal ketika reader ingin membaca filenya

readcnt = 0 // tidak ada lagi yang membaca bukunya

Full = 0 // Semua slot di buffer tidak diisi. Jadi, slot penuh sama dengan 0

Empty = n // Awalnya, Semua slot kosong.

**Langkah 2: Solusi untuk Writers:**

do{

// menghasilkan sebuah item

wait (Empty);

wait (writer);

// tempatkan di buffer

signal(writer);

signal (Full);

} while (true);

**Penjelasan:**

* Ketika writer mengedit file maka satu slot akan terisi sekarang, sehingga nilai “kosong” dikurangi 1. Untuk mencegah reader mengakses buffer, nilai mutex juga dikurangi.
* Sekarang, writer telah menempatkan item di buffer.
* Jadi, nilai "penuh" dinaikkan 1.
* Begitu juga nilai mutex juga bertambah 1, hal ini dikarenakan writer telah menyelesaikan tugasnya dan sekarang reader dapat mengakses buffer tersebut.

**Langkah 3: Solusi untuk Reader:**

do {

// Sinyal Reader ingin membaca file

wait(mutex);

// tambahkan jumlah Reader menjadi 1

// untuk menjamin writer tidak dapat mengedit file nya

readcnt++;

if (readcnt==1)

wait(writer);

// Reader lain dapat meminta sinyal untuk membaca selama sinyal ada

signal(mutex);

wait(mutex);

readcnt--; // Ada reader yang sudah selesai membaca

if (readcnt == 0) // kondisi jika sudah tidak ada yang membaca file nya,

signal(writer); // writers dapat mengedit

signal(mutex);

} while(true);

**Penjelasan:**

* Saat reader ingin membaca file, maka berikan sinyal ke buffer, maka nilai "full" dikurangi 1. Demikian pula, nilai mutex juga berkurang sehingga reader tidak dapat mengakses buffer saat ini.
* Sekarang reader sudah membaca file tersebut.
* Jadi, nilai "kosong" dinaikkan 1.
* Demikian pula, Nilai mutex juga dinaikkan sehingga reader dapat mengakses buffer sekarang.

1. **Sleeping Barber Problem**

**Pernyataan Sleeping Barber Problem:**

**Ada toko tukang cukur yang memiliki satu orang tukang cukur, satu kursi, dan n kursi tunggu pelanggan jika ada yang duduk di kursi tersebut**. **Jika tidak ada pelanggan, maka tukang cukur tidur di kursinya sendiri**. **Ketika pelanggan datang, dia harus membangunkan tukang cukur**. **Jika ada banyak pelanggan dan tukang cukur memotong rambut pelanggan, maka pelanggan yang tersisa menunggu jika ada kursi kosong di ruang tunggu atau mereka pergi jika tidak ada kursi yang kosong**.

**Contoh Penggunaannya:**

**Untuk mengatasi masalah ini, diperlukan DUA Semaphore:**

* Customer
* Semaphore ini untuk melacak jumlah customer dalam buffer.
* Barber
* Semaphore ini untuk melacak status barber di buffer.

**Dan juga diperlukan 2 Operasi Berikut:**

* wait()
* Untuk **MENGURANGI value Semaphore Sebanyak 1**.
* signal()
* Untuk **MENAMBAHKAN value Semaphore Sebanyak 1**.

**Langkah 1: Inisialisasi Semaphores:**

Chairs = 5

Customer = 0

Barber = 0

mutex = 1

waiting = 0

**Langkah 2: Solusi untuk Customer:**

do{

wait (mutex); // menghasilkan sebuah item

if(waiting < chair){

waiting = waiting + 1;

signal(customer);

signal(mutex); // tempatkan di buffer

wait(barber);

get\_haircut();

}else wait (mutex);

} while (true);

**Penjelasan:**

* Ketika customer datang dan kursi kosong maka value waiting akan bertambah 1 dan akan ditempatkan di buffer.
* Customer membangunkan tukang cukur dan dapat dipotong rambutnya.
* Nilai mutex juga bertambah 1, hal ini dikarenakan tukang cukur sedang menjalankan tugasnya.
* Sehingga Customer lain yang datang, dapat menunggu dengan menambah item mutex dan menambah nilai mutexnya atau pergi saja tanpa mencukur rambutnya.

**Langkah 3: Solusi untuk Barber:**

while(true){

wait(customer);

wait(mutex);

signal(barber);

cut\_hair();

signal(mutex);

waiting = waiting - 1;

}

**Penjelasan:**

* Barber menunggu pelanggan dan buffer akan dinaikkan menjadi 1 sehingga customer lain tidak bisa mengakses buffer kursi cukur.
* Barber akan mencukur rambut customer dan memberi sinyal ke mutex jika sudah selesai.
* Jadi, nilai "mutex" dikurangi 1.
* Dan nilai waiting berkurang 1, dan akan terus begitu sampai tidak ada antrian customer tersisa dan Barber akan tidur kembali.

1. Dalam sebuah sistem 4 proses (A, B, C dan D) dan 4 sumber daya (S, T, U dan V) sedang berjalan. Pada saat tersebut, proses sedang menggunakan sumber daya seperti tertera pada matrix dibawah ini. Sistem tersebut secara keseluruhan 5 unit sumber daya S, 5 unit sumber daya T, 5 unit sumber daya U dan 5 unit sumber daya V.

A picture containing text, crossword puzzle

Description automatically generated

Matrix dibawah ini menunjukkan sumber daya yang masih dibutuhkan untuk dapat menyelesaikan proses.

A picture containing text, crossword puzzle

Description automatically generated

Menggunakan algoritma Banker's, tentukan urutan proses yang harus dijalankan agar tidak terjadi deadlock (Jika ada).

**Jawaban:**

**Terdapat 4 sumber daya, yaitu**: 5S 5T 5U 5V

**Dengan 4 proses sebagai berikut:**

* A = menggunakan 1S 1T 2V meminta 1S 1T
* B = menggunakan 1S 1U 1V meminta 3U
* C = menggunakan 2S 2T 1U 1V meminta 3S 4T 1U 1V
* D = menggunakan 1T 3U 1V meminta 2S 1T 1V

**Total Penggunaan** = 4S 4T 5U 5V

**Sisa Sumber Daya** = 1S 1T 0U 0V

1. **Bisa Memproses Permintaan A sehingga Sisa Sumber Daya** = 2S 2T 0U 2V
2. **Bisa Memproses Permintaan D sehingga Sisa Sumber Daya** = 2S 3T 3U 3V
3. **Bisa Memproses Permintaan B sehingga Sisa Sumber Daya** = 3S 3T 4U 4V

**Sisa sumber daya** 3S 3T 4U 4V, maka **tidak dapat menyelesaikan proses C**

**Dikarenakan kurang 1T** dan **AKAN TERJADI DEADLOCK**.