



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : DEADLOCK
NAMA : DANI ADRIAN
NIM : 225150201111009
TANGGAL : 11/04/2023
ASISTEN : ZHAFRAN RAMA AZMI
GIBRAN HAKIM

6.4. Langkah Praktikum

Praktikum ini dilakukan dengan terlebih dahulu terhubung dengan layanan aws educate dengan cara mengaktifkan instance dari halaman instance summary. Pilih action dan start untuk mengaktifkan instance. Lakukan koneksi SSH dengan cara yang sama seperti pada Bab 1.

1. Login ke sistem GNU/LINUX kemudian buka terminal
2. Lakukan percobaan terhadap tiap kode program pada masing-masing sub bab berikut.
3. Lakukan kompilasi kode program dengan menggunakan perintah
Gcc -o <Nama_Output> <Nama_Source.c>
4. Jalankan hasil kompilasi menggunakan perintah ./<Nama_Output>
5. Captutre / snapshot output-nya dan simpan sebagai laporan.
6. Amati output tiap percobaan dan jawab pertanyaan pada masing-masing percobaan.

1.	#include <stdio.h>
2.	#include <stdlib.h>
3.	
4.	void print(int x[][10], int n, int m)
5.	{
6.	int i, j;
7.	for (i = 0; i < n; i++)
8.	{
9.	printf("\n");
10.	for (j = 0; j < m; j++)
11.	{
12.	printf("%d\t", x[i][j]);
13.	}
14.	}
15.	}
16.	
17.	// Resource Request algorithm
18.	void res_request(int A[10][10], int
	N[10][10], int AV[10][10], int pid, int m)
19.	{
20.	int reqmat[1][10];
21.	int i;



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
22.     printf("\n Enter additional request :- \n");
23.     for (i = 0; i < m; i++)
24.     {
25.         printf(" Request for resource %d : ", i +
26.         1);
27.         scanf("%d", &reqmat[0][i]);
28.     }
29.     for (i = 0; i < m; i++)
30.         if (reqmat[0][i] > N[pid][i])
31.         {
32.             printf("\n Error encountered.\n");
33.             exit(0);
34.         }
35.     for (i = 0; i < m; i++)
36.         if (reqmat[0][i] > AV[0][i])
37.         {
38.             printf("\n Resources unavailable.\n");
39.             exit(0);
40.         }
41.     for (i = 0; i < m; i++)
42.     {
43.         AV[0][i] -= reqmat[0][i];
44.         A[pid][i] += reqmat[0][i];
45.         N[pid][i] -= reqmat[0][i];
46.     }
47.
48. // Safety algorithm
49. int safety(int A[][10], int N[][10], int
50. AV[1][10], int n, int m, int a[])
51. {
52.     int i, j, k, x = 0;
53.     int F[10], W[1][10];
54.     int pflag = 0, flag = 0;
55.     for (i = 0; i < n; i++)
56.         F[i] = 0;
57.     for (i = 0; i < m; i++)
58.         W[0][i] = AV[0][i];
59.     for (k = 0; k < n; k++)
60.     {
61.         for (i = 0; i < n; i++)
62.         {
63.             if (F[i] == 0)
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
64.         flag = 0;
65.         for (j = 0; j < m; j++)
66.         {
67.             if (N[i][j] > W[0][j])
68.                 flag = 1;
69.         }
70.         if (flag == 0 && F[i] == 0)
71.         {
72.             for (j = 0; j < m; j++)
73.                 W[0][j] += A[i][j];
74.             F[i] = 1;
75.             pflag++;
76.             a[x++] = i;
77.         }
78.     }
79. }
80. if (pflag == n)
81.     return 1;
82. }
83. return 0;
84. }
85.
86. // Banker's Algorithm
87. void accept(int A[][10], int N[][10], int
M[10][10], int W[1][10], int *n, int *m)
88. {
89.     int i, j;
90.     printf("\n Enter total no. of processes :
");
91.     scanf("%d", n);
92.     printf("\n Enter total no. of resources :
");
93.     scanf("%d", m);
94.     for (i = 0; i < *n; i++)
95.     {
96.         printf("\n Process %d\n", i + 1);
97.         for (j = 0; j < *m; j++)
98.         {
99.             printf(" Allocation for resource %d :
", j + 1);
100.             scanf("%d", &A[i][j]);
101.             printf(" Maximum for resource %d : ", j
+ 1);
102.             scanf("%d", &M[i][j]);
103.         }
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
104.     }
105.     printf("\n Available resources : \n");
106.     for (i = 0; i < *m; i++)
107.     {
108.         printf(" Resource %d : ", i + 1);
109.         scanf("%d", &W[0][i]);
110.     }
111.     for (i = 0; i < *n; i++)
112.         for (j = 0; j < *m; j++)
113.             N[i][j] = M[i][j] - A[i][j];
114.     printf("\n Allocation Matrix");
115.     print(A, *n, *m);
116.     printf("\n Maximum Requirement Matrix");
117.     print(M, *n, *m);
118.     printf("\n Need Matrix");
119.     print(N, *n, *m);
120. }
121.
122. int banker(int A[][10], int N[][10], int
W[1][10], int n, int m)
123. {
124.     int j, i, a[10];
125.     j = safety(A, N, W, n, m, a);
126.     if (j != 0)
127.     {
128.         printf("\n\n");
129.         for (i = 0; i < n; i++)
130.             printf(" P%d ", a[i]);
131.         printf("\n A safety sequence has been
detected.\n");
132.         return 1;
133.     }
134.     else
135.     {
136.         printf("\n Deadlock has occured.\n");
137.         return 0;
138.     }
139. }
140.
141. int main()
142. {
143.     int ret;
144.     int A[10][10];
145.     int M[10][10];
146.     int N[10][10];
```



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
147.     int W[1][10];
148.     int n, m, pid, ch;
149.     printf("\n DEADLOCK AVOIDANCE USING
BANKER'S ALGORITHM\n");
150.     accept(A, N, M, W, &n, &m);
151.     ret = banker(A, N, W, n, m);
152.     if (ret != 0)
153.     {
154.         printf("\n Do you want make an additional
request ? (1=Yes|0=No)");
155.         scanf("%d", &ch);
156.         if (ch == 1)
157.         {
158.             printf("\n Enter process no. : ");
159.             scanf("%d", &pid);
160.             res_request(A, N, W, pid - 1, m);
161.             ret = banker(A, N, W, n, m);
162.             if (ret == 0)
163.                 exit(0);
164.         }
165.     }
166.     else
167.         exit(0);
168.     return 0;
169. }
```

1. Contoh jika terjadi deadlock



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
Maximum for resource 3 : 2

Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3

Available resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2

Allocation Matrix
0      1      0
2      0      0
3      0      2
2      1      1
0      0      2
Maximum Requirement Matrix
7      5      3
3      2      2
9      0      2
2      2      2
4      3      3
Need Matrix
7      4      3
1      2      2
6      0      0
0      1      1
4      3      1

P1 P3 P4 P0 P2
A safety sequence has been detected.

Do you want make an additional request ? (1=Yes|0=No)1

Enter process no. : 5

Enter additional request :-
Request for resource 1 : 3
Request for resource 2 : 3
Request for resource 3 : 0

Deadlock has occurred.
userlinux@Linux:~$
```

Open nama prak

1 Nama : Dani Adrian
2 Nim : 225150201111009

Plain Text Ln 1, Col 1

“Deadlock has occurred” menunjukkan program terhad deadlock

2. Contoh jika tidak terjadi deadlock



```
Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3

Available resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2

Allocation Matrix
0      1      0
2      0      0
3      0      2
2      1      1
0      0      2

Maximum Requirement Matrix
7      5      3
3      2      2
9      0      2
2      2      2
4      3      3

Need Matrix
7      4      3
1      2      2
6      0      0
0      1      1
4      3      1

P1 P3 P4 P0 P2
A safety sequence has been detected.

Do you want make an additional request ? (1=Yes|0=No)1

Enter process no. : 5

Enter additional request :-
Request for resource 1 : 0
Request for resource 2 : 2
Request for resource 3 : 0

P3 P4 P1 P2 P0
A safety sequence has been detected.
userlinux@Linux:~$
```

“A safety sequence has been detected”, menunjukkan program tidak terjadi deadlock.

6.5. Pembahasan

Setelah anda berhasil melakukan kompilasi program lakukan percobaan dengan langkah-langkah berikut:

1. Masukkan variable berikut :



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

a. Jumlah proses = 5

b. Jumlah resource = 3

Dengan Matriks

	Allocation			Max			Available		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

Jawab:



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
userlinux@Linux:~$ ./deadlock
```

```
DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM
```

```
Enter total no. of processes : 5
```

```
Enter total no. of resources : 3
```

```
Process 1
```

```
Allocation for resource 1 : 0
```

```
Maximum for resource 1 : 7
```

```
Allocation for resource 2 : 1
```

```
Maximum for resource 2 : 5
```

```
Allocation for resource 3 : 0
```

```
Maximum for resource 3 : 3
```

```
Process 2
```

```
Allocation for resource 1 : 2
```

```
Maximum for resource 1 : 3
```

```
Allocation for resource 2 : 0
```

```
Maximum for resource 2 : 2
```

```
Allocation for resource 3 : 0
```

```
Maximum for resource 3 : 2
```

```
Process 3
```

```
Allocation for resource 1 : 3
```

```
Maximum for resource 1 : 9
```

```
Allocation for resource 2 : 0
```

```
Maximum for resource 2 : 0
```

```
Allocation for resource 3 : 2
```

```
Maximum for resource 3 : 2
```

```
Process 4
```

```
Allocation for resource 1 : 2
```

```
Maximum for resource 1 : 2
```

```
Allocation for resource 2 : 1
```

```
Maximum for resource 2 : 2
```

```
Allocation for resource 3 : 1
```

```
Maximum for resource 3 : 2
```

```
Process 5
```

```
Allocation for resource 1 : 0
```






```
Maximum for resource 1 : 4
```

```
Allocation for resource 2 : 0
```


```
Maximum for resource 2 : 3
```

```
Allocation for resource 3 : 2
```

```
Maximum for resource 3 : 3
```

Open ▾  nama prak ~/
Save    

1 Nama : Dani Adrian
2 Nim : 2251502011111009

Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾  INS



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

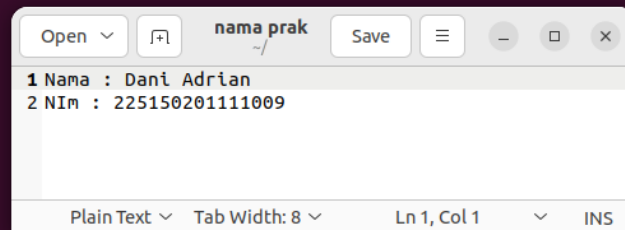
```
Process 2
Allocation for resource 1 : 2
Maximum for resource 1 : 3
Allocation for resource 2 : 0
Maximum for resource 2 : 2
Allocation for resource 3 : 0
Maximum for resource 3 : 2
```

```
Process 3
Allocation for resource 1 : 3
Maximum for resource 1 : 9
Allocation for resource 2 : 0
Maximum for resource 2 : 0
Allocation for resource 3 : 2
Maximum for resource 3 : 2
```

```
Process 4
Allocation for resource 1 : 2
Maximum for resource 1 : 2
Allocation for resource 2 : 1
Maximum for resource 2 : 2
Allocation for resource 3 : 1
Maximum for resource 3 : 2
```

```
Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3
```

```
Available resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2
```



2. Amati hasilnya dan tuliskan outputnya!

Jawab:



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
Maximum for resource 2 : 0
Allocation for resource 3 : 2
Maximum for resource 3 : 2

Process 4
Allocation for resource 1 : 2
Maximum for resource 1 : 2
Allocation for resource 2 : 1
Maximum for resource 2 : 2
Allocation for resource 3 : 1
Maximum for resource 3 : 2

Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3

Available resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2

Allocation Matrix
0      1      0
2      0      0
3      0      2
2      1      1
0      0      2

Maximum Requirement Matrix
7      5      3
3      2      2
9      0      2
2      2      2
4      3      3

Need Matrix
7      4      3
1      2      2
6      0      0
0      1      1
4      3      1

P1 P3 P4 P0 P2
A safety sequence has been detected.

Do you want make an additional request ? (1=Yes|0=No)
```

Program menampilkan output tabel allocation matrix, maximum requirement matrix, dan need matrix. Output urutan proses yang dijalankan adalah P1, P3, P4, P0 dan P2 serta keterangan “A safety sequence has been detected” yang menunjukkan tidak terjadi deadlock.

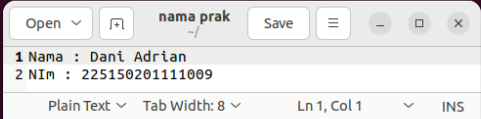
3. Apakah permintaan penambahan resource dari kondisi pada no 1 diijinkan apabila P4 melakukan permintaan tambahan alokasi sebanyak R1=3, R2=3 dan R3=0



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Jawab:

```
Do you want make an additional request ? (1=Yes|0=No)1
Enter process no. : 5
Enter additional request :-
Request for resource 1 : 3
Request for resource 2 : 3
Request for resource 3 : 0
```



4. Amati hasilnya dan tuliskan outputnya untuk kondisi dari point 3.

Jawab:



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
Maximum for resource 3 : 2

Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3

Available resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2

Allocation Matrix
0      1      0
2      0      0
3      0      2
2      1      1
0      0      2
Maximum Requirement Matrix
7      5      3
3      2      2
9      0      2
2      2      2
4      3      3
Need Matrix
7      4      3
1      2      2
6      0      0
0      1      1
4      3      1

P1 P3 P4 P0 P2
A safety sequence has been detected.

Do you want make an additional request ? (1=Yes|0=No)1

Enter process no. : 5

Enter additional request :-
Request for resource 1 : 3
Request for resource 2 : 3
Request for resource 3 : 0

Deadlock has occurred.
userlinux@Linux:~$
```

Program melakukan permintaan penambahan sumber daya pada proses ke-5 (P4). Setelah dieksekusi, output berupa "Deadlock has occurred". Sehingga dapat disimpulkan telah terjadi deadlock pada saat melakukan penambahan sumber daya.

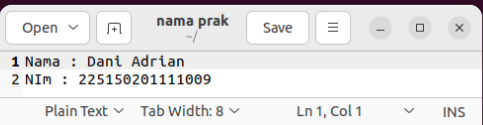
5. Apakah permintaan penambahan resource dari kondisi pada no 1 diijinkan apabila P4 melakukan permintaan tambahan alokasi sebanyak R1=0, R2=2 dan R3=0



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

Jawab:

```
Do you want make an additional request ? (1=Yes|0=No)1
Enter process no. : 5
Enter additional request :-
Request for resource 1 : 0
Request for resource 2 : 2
Request for resource 3 : 0
```



6. Amati hasilnya dan tuliskan outputnya untuk kondisi pada point 5.

Jawab:



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

```
Process 5
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3

Available resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2

Allocation Matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Maximum Requirement Matrix
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

Need Matrix
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1

P1 P3 P4 P0 P2
A safety sequence has been detected.

Do you want make an additional request ? (1=Yes|0=No)1

Enter process no. : 5

Enter additional request :-
Request for resource 1 : 0
Request for resource 2 : 2
Request for resource 3 : 0

P3 P4 P1 P2 P0
A safety sequence has been detected.
userlinux@Linux:~$
```

Program menrequest penambahan sumber daya pada proses ke-5 (P4). Namun dengan nilai yang berbeda. Setelah dieksekusi, menampilkan output “A safety sequence has been detected.” yang menandakan tidak terjadi deadlock pada saat penambahan resource.

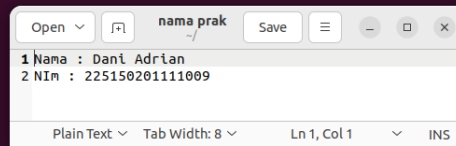
7. Pada bagian program terdapat Resource Request Alghorithm (baris 14 sampai 40) dan Safety Alghoritm (baris 42- 72). Jelaskan konsep algoritmanya dan penggunaannya!



Jawab:

Resource Request Algorithm

```
// Resource Request algorithm
void res_request(int A[10][10], int N[10][10], int AV[10][10], int pid, int m)
{
    int reqmat[1][10];
    int i;
    printf("\n Enter additional request :- \n");
    for (i = 0; i < m; i++)
    {
        printf(" Request for resource %d : ", i + 1);
        scanf("%d", &reqmat[0][i]);
    }
    for (i = 0; i < m; i++)
    {
        if (reqmat[0][i] > N[pid][i])
        {
            printf("\n Error encountered.\n");
            exit(0);
        }
    }
    for (i = 0; i < m; i++)
    {
        if (reqmat[0][i] > AV[0][i])
        {
            printf("\n Resources unavailable.\n");
            exit(0);
        }
    }
    for (i = 0; i < m; i++)
    {
        AV[0][i] -= reqmat[0][i];
        A[pid][i] += reqmat[0][i];
        N[pid][i] -= reqmat[0][i];
    }
}
```

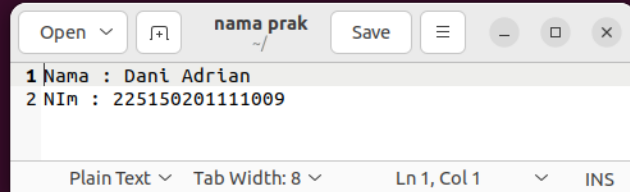


Konsep algoritma resource request adalah membandingkan input resource request dengan ketersediaan memori. Apabila request matrix lebih dari need matrix maka akan menampilkan output “Error encountered”. Namun jika request melebihi matrix available maka akan menampilkan output “Resource unavailable”.

Safety Algorithm



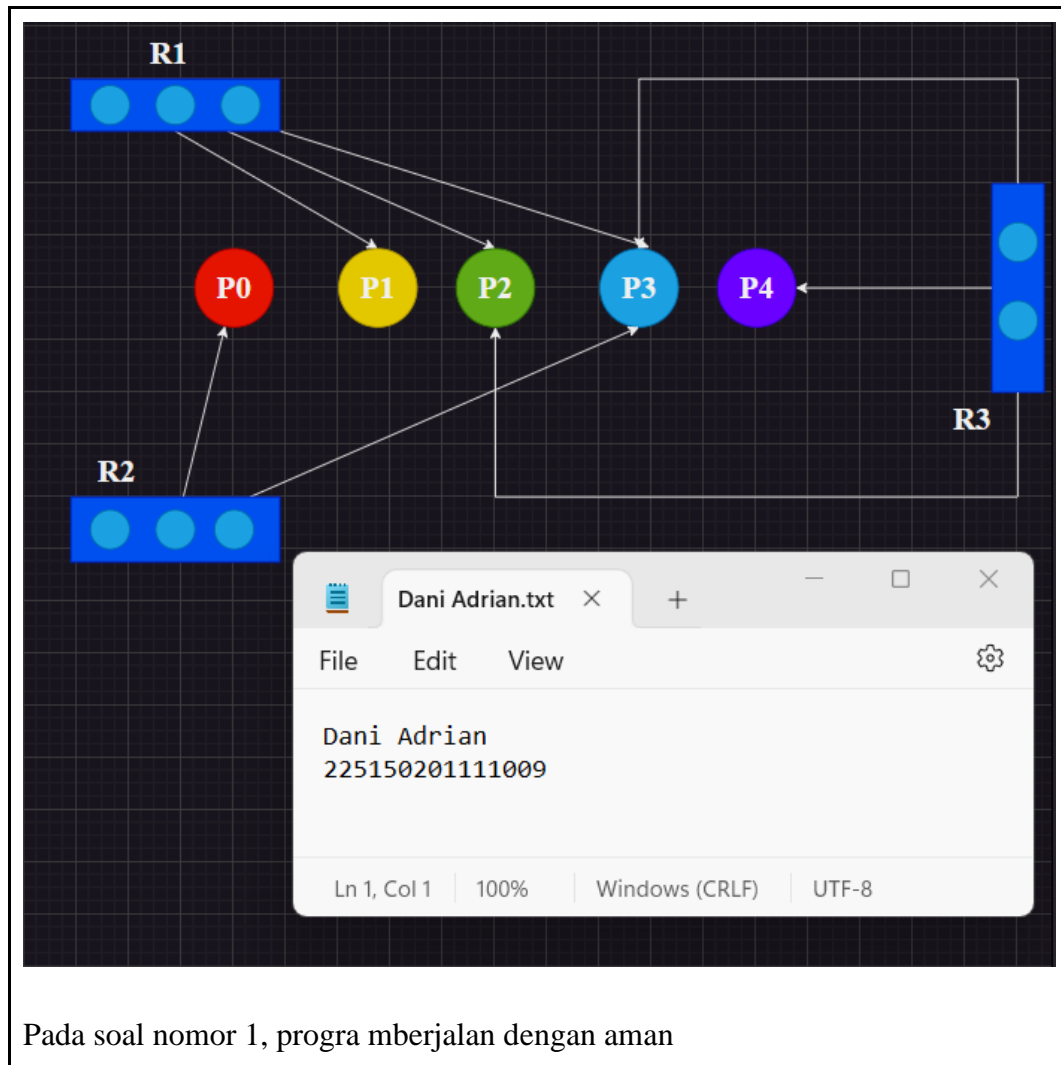
```
// Safety algorithm
int safety(int A[][10], int N[][10], int AV[1][10], int n, int m, int a[])
{
    int i, j, k, x = 0;
    int F[10], W[1][10];
    int pflag = 0, flag = 0;
    for (i = 0; i < n; i++)
        F[i] = 0;
    for (i = 0; i < m; i++)
        W[0][i] = AV[0][i];
    for (k = 0; k < n; k++)
    {
        for (i = 0; i < n; i++)
        {
            if (F[i] == 0)
            {
                flag = 0;
                for (j = 0; j < m; j++)
                {
                    if (N[i][j] > W[0][j])
                        flag = 1;
                }
                if (flag == 0 && F[i] == 0)
                {
                    for (j = 0; j < m; j++)
                        W[0][j] += A[i][j];
                    F[i] = 1;
                    pflag++;
                    a[x++] = i;
                }
            }
        }
        if (pflag == n)
            return 1;
    }
    return 0;
}
```

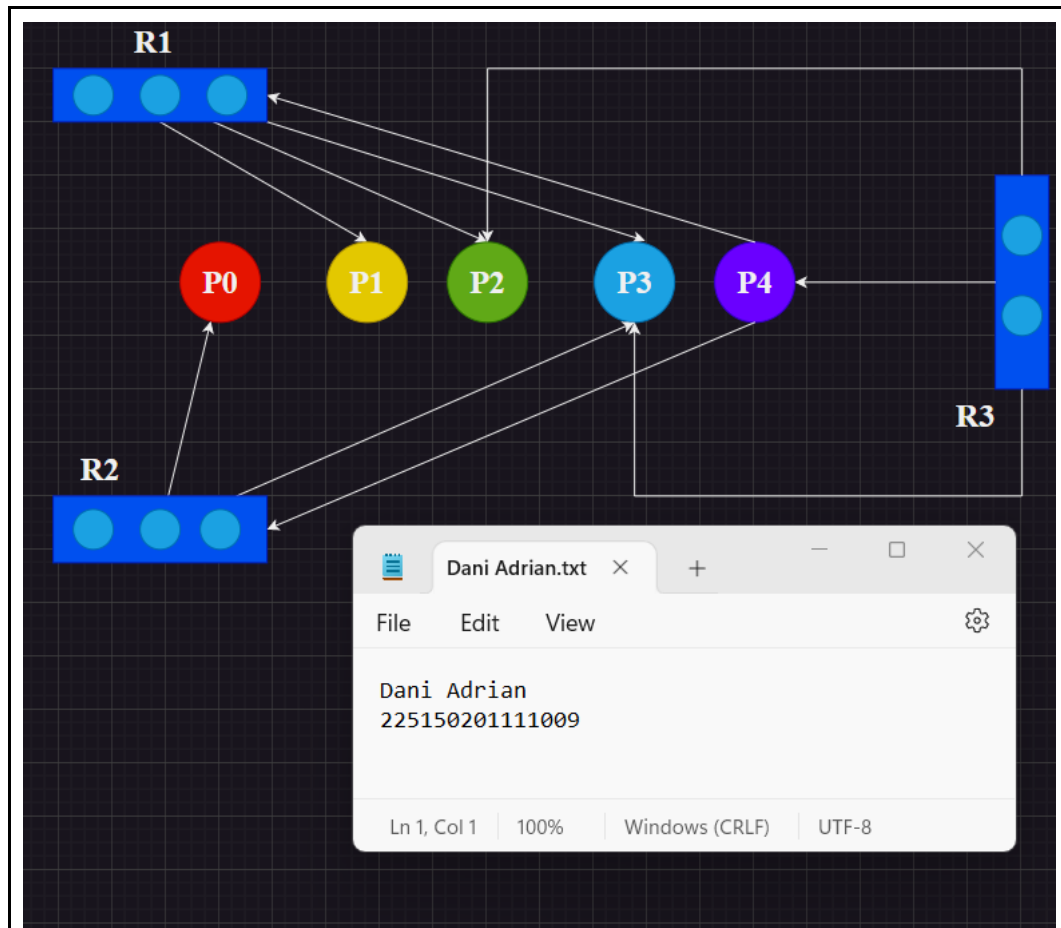


Safe algorithm memeriksa apakah kondisi suatu sistem safe atau tidak setelah memasukkan alokasi resource pada proses

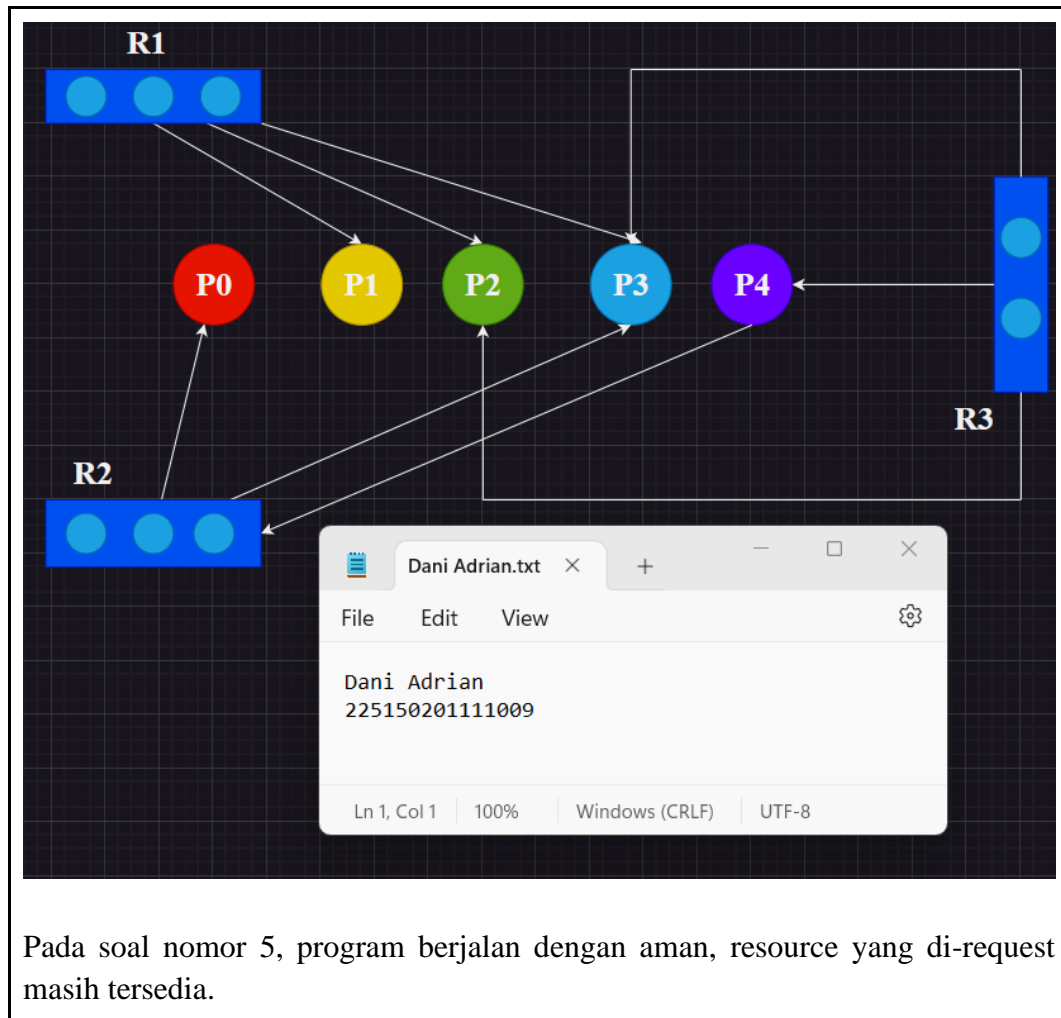
8. Dapatkan kondisi pada soal no 1, 3 dan 5 di gambarkan dengan menggunakan Resource Allocation Graph? Jelaskan!

Jawab:





Pada soal nomor 3, program mengalami deadlock dikarenakan P4 melakukan penambahan resource yang melebihi kapasitas memori yang tersedia.



6.6. Kesimpulan

Terdapat 2 jenis resource, preemptable resource dan non preemptable resource. Preemptable resource merupakan resource yang dapat diambil dari proses yang memilikinya tanpa menimbulkan efek apapun. Sedangkan non preemptable resource merupakan resource yang tidak dapat diambil dari proses karena akan menimbulkan efek kegagalan komputasi.

Ada beberapa algoritma yang dapat mencegah terjadinya deadlock, diantaranya sebagai berikut:

1. Algoritma Resource Allocation Graph

Algoritma ini hanya dapat diaplikasikan pada sistem yang mempunyai satu anggota untuk setiap sumber dayanya. Algoritma ini digambarkan



dengan graph dimana jika terjadi sebuah siklus, maka alokasi akan membawa sistem pada kondisi state tidak aman (unsafe state) yang dapat menyebabkan terjadinya deadlock. Sedangkan jika tidak terdapat siklus pada graph, maka alokasi sumber daya akan menyebabkan sistem berada pada kondisi aman (safe state) sehingga tidak akan terjadi deadlock.

2. Algoritma Banker

Algoritma ini dapat digambarkan sebagai seorang banker yang berurusan dengan kelompok orang yang meminta pinjaman kepada banker tersebut. Algoritma ini dapat diaplikasikan pada sistem yang mempunyai beberapa anggota pada setiap tipe sumber daya. Setiap proses sebelum dieksekusi harus menentukan jumlah sumber daya maksimum yang dibutuhkan. Jika suatu proses meminta sumber daya, kemungkinan proses tersebut harus menunggu dan jika suatu proses mendapatkan semua sumber daya, maka proses tersebut harus mengembalikan semua sumber daya dalam jangka waktu tertentu. Jika melewati waktu tersebut, maka kemungkinan akan terjadi deadlock.

3. Algoritma safety

Algoritma ini bekerja dengan cara algoritma mencari apakah sistem dalam status aman atau tidak. Status ini terjadi jika sistem dapat mengalokasikan sumber daya untuk setiap proses dalam keadaan tertentu dan masih dapat terjadi deadlock.

Cara agar memulihkan sistem dari deadlock ada 2, yaitu Terminasi Proses dan Rollback dan Restart. Pemulihan sistem dapat dilakukan dengan cara terminasi secara satu per satu sampai Circular Wait hilang. Dalam melakukan terminasi, terdapat faktor-faktor yang menentukan proses mana yang akan diterminasi, yaitu prioritas dari proses-proses yang terlibat deadlock, lama waktu yang dibutuhkan untuk eksekusi dan waktu proses tersebut menunggu sumber daya, banyak sumber daya yang telah dihabiskan dan yang masih dibutuhkan. Terakhir adalah factor utilitas dari proses. Selain dengan terminasi, untuk memulihkan sistem dari deadlock dapat dilakukan dengan rollback dan restart yaitu dengan cara sistem melakukan preempt terhadap sebuah proses menjadi korban, selain itu juga harus menghindari keadaan dimana proses yang sama selalu menjadi korban yang dapat mengakibatkan proses tersebut tidak akan pernah sukses menjalankan eksekusi.

Safe state merupakan suatu kondisis dimana sistem dapat mengalokasikan sumber daya untuk setiap proses secara berurutan dan menghindari



LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

deadlock. Pada saat proses meminta sumber daya yang tersedia, sistem harus menentukan apakah alokasi sumber daya tersebut mengakibatkan sistem dalam safe state. Safe state juga dapat diartikan dengan banyaknya sumber daya yang dialokasikan terhadap masing-masing proses.