MIDDLE EAST TECHNICAL UNIVERSITY

CNG 334: INTRODUCTION TO OPERATING SYSTEMS

ASSIGNMENT 2

NAME: MERT CAN BİLGİN

STUDENT ID: 2453025

TASK 1:

Firstly, let's find the required column by subtracting the Alloc from Max:

Process	Max	Alloc	Req	Available
A	0012	0012	0000	1524
B	1750	1000	0750	
C	2356	1354	1002	
D	0652	0632	0020	
E	0656	0014	0642	

Now, we start to check from process A. If we can provide its needs, we can release its allocation. Because we can satisfy requirement of process A, we can update our available resources.

Process	Max	Alloc	Req	Available
A	0012	0012	0000	~~1524~~
B	1750	1000	0750	1536
C	2356	1354	1002	
D	0652	0632	0020	
E	0656	0014	0642	

Then, we check for Process B. Because we can't provide its needs for R2 and R3, we skip to process C. Process C's needs can be provided, so we can release it allocated resources.

Process	Max	Alloc	Req	Available
A	0012	0012	0000	~~1 5 2 4~~
B	1750	1000	0750	~~1 5 3 6~~
C	2356	1354	1002	2 8 8 10
D	0652	0632	0020	
E	0656	0014	0642	

After that, we check for Process D. Since its requirements can be provided, we can release its allocated resources.

Process	Max	Alloc	Req	Available
A	0012	0012	0000	~~1 5 2 4~~
B	1750	1000	0750	~~1 5 3 6~~
C	2356	1354	1002	~~2 8 8 10~~
D	0652	0632	0020	2 14 11 12
E	0656	0014	0642	

Now, check for Process E. Process E's requirements can be satisfied, so release its allocated resources.

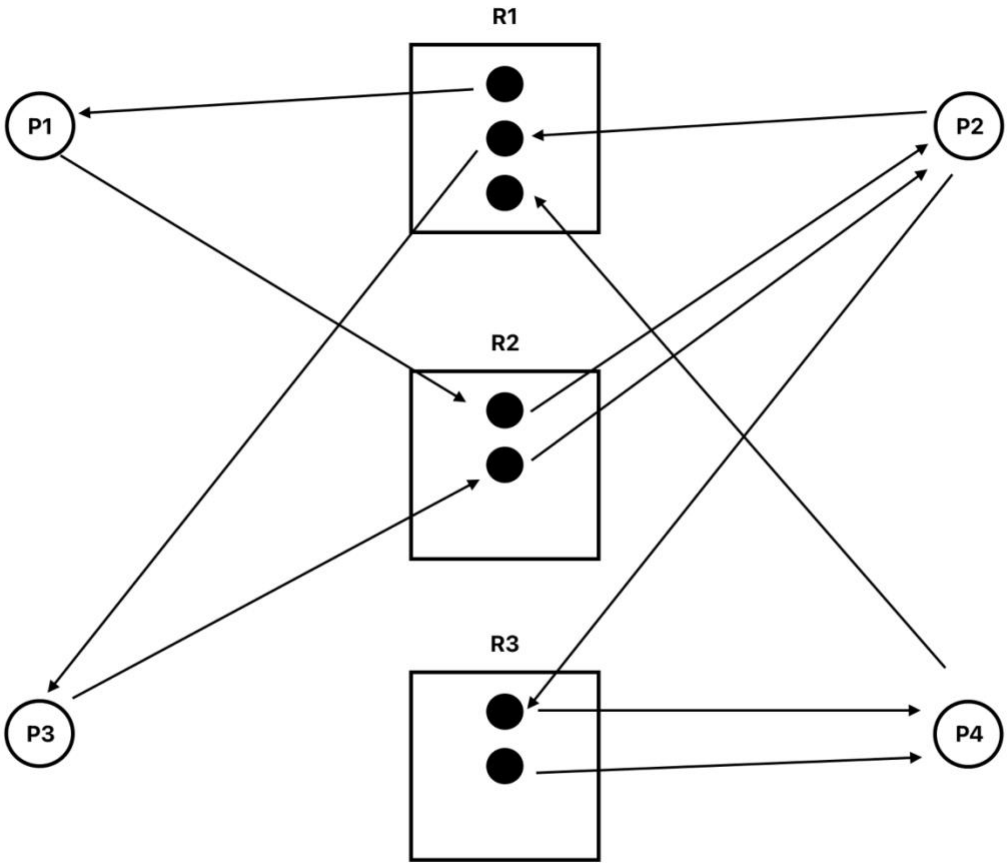
Process	Max	Alloc	Req	Available
A	0012	0012	0000	~~1 5 2 4~~
B	1750	1000	0750	~~1 5 3 6~~
C	2356	1354	1002	~~2 8 8 10~~
D	0652	0632	0020	~~2 14 11 12~~
E	0656	0014	0642	2 14 12 16

Go back, and check for Process B. Now, its requirements can be satisfied, so we can release its allocated resources.

Process	Max	Alloc	Req	Available
A	0012	0012	0000	~~1 5 2 4~~
B	1750	1000	0750	~~1 5 3 6~~
C	2356	1354	1002	~~2 8 8 10~~
D	0652	0632	0020	~~2 14 11 12~~
E	0656	0014	0642	~~2 14 12 16~~
				3 14 12 16

Hence, the safe sequence is A -> C -> D -> E -> B.

TASK 2:



Even though there are cycles in the graph, there is no deadlock. There is one available resource in R1, it can be given to P4. Then, the allocated resources for P4 will be released, and we will have two resources for R3. With that, P2 will be able to be completed because there are enough resources in R1 and R3. After that, its allocated resources will be released, and P1 and P3 will be able to complete their requests. Therefore, it's a deadlock-free graph.

TASK 3:

Let's find out the requirements first:

Process	Max	Alloc	Req	Available
A	1 1 2 1 3	1 0 2 1 1	0 1 0 0 2	0 0 2 1 1
B	2 2 X 1 1	2 0 1 1 1	0 2 X-1 0 0	
C	2 1 3 1 0	1 1 0 1 0	1 0 3 0 0	
D	1 1 2 2 1	1 1 1 1 0	0 0 1 1 1	

Check for process D, and release its allocated resources:

Process	Max	Alloc	Req	Available
A	1 1 2 1 3	1 0 2 1 1	0 1 0 0 2	~~0 0 2 1 1~~
B	2 2 X 1 1	2 0 1 1 1	0 2 X-1 0 0	1 1 3 2 1
C	2 1 3 1 0	1 1 0 1 0	1 0 3 0 0	
D	1 1 2 2 1	1 1 1 1 0	0 0 1 1 1	

Check for process C, and release its allocated resources:

Process	Max	Alloc	Req	Available
A	1 1 2 1 3	1 0 2 1 1	0 1 0 0 2	~~0 0 2 1 1~~
B	2 2 X 1 1	2 0 1 1 1	0 2 X-1 0 0	~~1 1 3 2 1~~
C	2 1 3 1 0	1 1 0 1 0	1 0 3 0 0	2 2 3 3 1
D	1 1 2 2 1	1 1 1 1 0	0 0 1 1 1	

Now, we need to check for process B. X-1 should be less than or equal to 3. So, the maximum value for X is 4 for a safe sequence.

TASK 4:

Best-Fit:

P3	P4	P2	P5	P6	2K	Opr. Sys.

First-Fit:

P3	P4	P2	P5	P6	2K	Opr. Sys.

Worst-Fit:

P3	P6	P2	P4	P5	2K	Opr. Sys.

TASK 5:

FIFO:

2	**1**	**4**	**3**	**1**	**2**	**4**	**3**	**2**	**1**
2	2	2	~~2~~	3	3	3	3	3	3
	1	1	1	1	~~1~~	2	2	2	2
		4	4	4	4	4	4	4	~~4~~
F	**F**	**F**	**F**	**X**	**F**	**X**	**X**	**X**	**F**

Hence, 6 page faults are generated by FIFO.

LRU:

2	**1**	**4**	**3**	**1**	**2**	**4**	**3**	**2**	**1**
2	2	2	~~2~~	3	3	~~3~~	4	4	4
	1	1	1	1	1	1	~~1~~	3	3
		4	4	4	~~4~~	2	2	2	2
F	**F**	**F**	**F**	**X**	**F**	**F**	**F**	**X**	**F**

Hence, 8 page faults are generated by LRU.

TASK 6:

Let's use the example of Task 5. To decide, we also need to perform Second-Chance, and Optimal Replacement algorithms.

Optimal Replacement:

2	**1**	**4**	**3**	**1**	**2**	**4**	**3**	**2**	**1**
2	2	2	2	2	2	2	2	2	2
	1	1	1	1	1	~~1~~	~~4~~	4	4
		4	4	3	3	3	3	3	3
F	**F**	**F**	**F**	**X**	**X**	**F**	**X**	**X**	**F**

Hence, 6 page faults are generated by Optimal Replacement algorithm.

Second-chance Replacement:

2	**1**	**4**	**3**	**1**	**2**	**4**	**3**	**2**	**1**
2 (0)	2 (0)	2 (0)	~~2~~ (0)	3 (0)	~~3~~ (0)	2 (0)	~~2~~ (0)	~~3~~ (0)	2 (0)
	1 (0)	1 (0)	1 (0)	1 ~~(0)~~	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
		4 (0)	4 (0)	4 (0)	4 (0)	4 ~~(0)~~	4 (1)	4 (1)	4 (1)
F	**F**	**F**	**F**	**X**	**F**	**X**	**F**	**F**	**X**

Hence, 7 page faults are generated by Second-chance Replacement algorithm.

5 – Perfect: -
4- Good: Optimal, FIFO
3- Average: Second-chance
2 – Poor: LRU
1 – Bad: -

While FIFO and Second-Chance algorithms suffer from Belady's anomaly, the others do not.