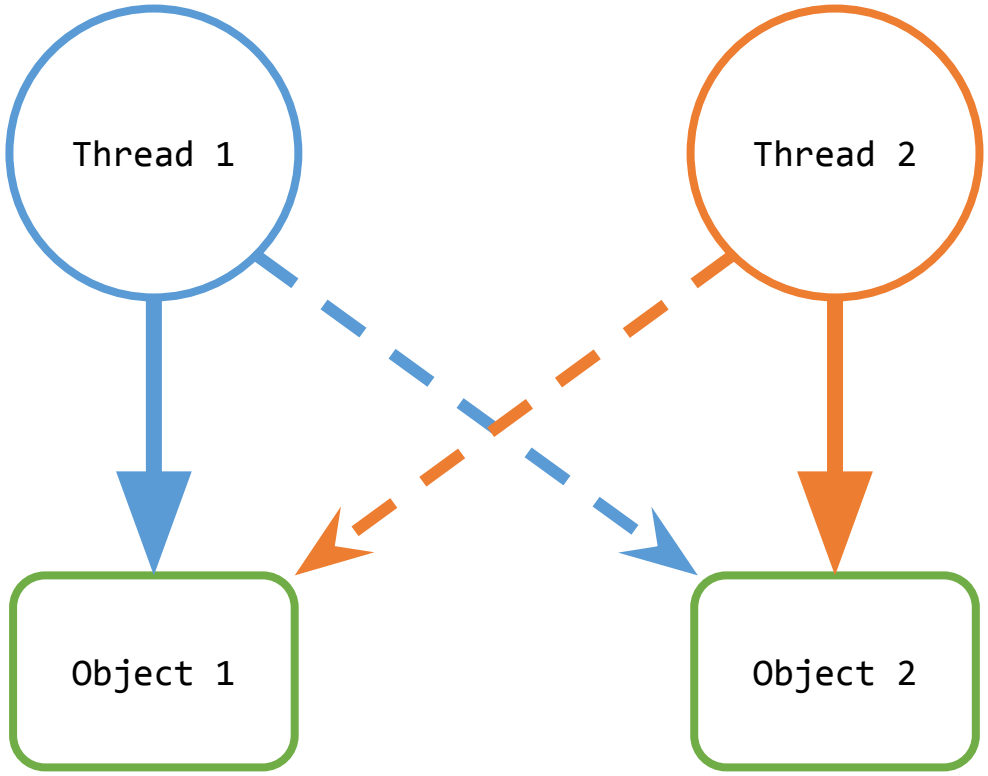
Избегание взаимоблокировок

Чиркин К.Д.,
гр.16ВВ1

Взаимоблокировка
?

Когда несколько процессов/потоков ожидают ресурсы, занятые друг другом, и ни один из них не может продолжать свое выполнение.

Взаимоблокировка
?



DEADLOCK

Способы **предотвращения** тупиков основаны на **“атаке” одного из условий** возникновения взаимоблокировки

Избегание основано на использовании специальных алгоритмов **распределения ресурсов,** не допускающих возникновения взаимоблокировки, т.е. исключается возможность ожидания ресурса.

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B

Need

	A	B
P0		
P1		
P2		

Finish

P0	
P1	
P2	

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B
1	2

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Finish

P0	F
P1	F
P2	F

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B
1	2

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Finish

P0	F
P1	F
P2	F

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation
2. Искать i, которое удовлетворяет условиям:
   1. Finish[i] = False
   2. Need[i] <= Work

   Если таких i нет, то переходим на шаг 4
3. Work = Work + Allocation[i]

   Finish[i] = True

   Переходим на шаг 2
4. Если все значения Finish = True, то система в безопасном состоянии

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B
1	2

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Finish

P0	F
P1	F
P2	F

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation
2. Искать i, которое удовлетворяет условиям:
   1. Finish[i] = False
   2. Need[i] <= Work

   Если таких i нет, то переходим на шаг 4
3. Work = Work + Allocation[i]

   Finish[i] = True

   Переходим на шаг 2
4. Если все значения Finish = True, то система в безопасном состоянии

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B
1	2

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Finish

P0	F
P1	T
P2	F

P1

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation
2. Искать i, которое удовлетворяет условиям:
   1. Finish[i] = False
   2. Need[i] <= Work

   Если таких i нет, то переходим на шаг 4
3. Work = Work + Allocation[i]

   Finish[i] = True

   Переходим на шаг 2
4. Если все значения Finish = True, то система в безопасном состоянии

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Work

A	B
1	2

Finish

P0	F
P1	T
P2	F

P1

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation
2. Искать i, которое удовлетворяет условиям:
   1. Finish[i] = False
   2. Need[i] <= Work

   Если таких i нет, то переходим на шаг 4
3. Work = Work + Allocation[i]

   Finish[i] = True

   Переходим на шаг 2
4. Если все значения Finish = True, то система в безопасном состоянии

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B
4	4

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Finish

P0	F
P1	T
P2	T

P1 P2

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation
2. Искать i, которое удовлетворяет условиям:
   1. Finish[i] = False
   2. Need[i] <= Work

   Если таких i нет, то переходим на шаг 4
3. Work = Work + Allocation[i]

   Finish[i] = True

   Переходим на шаг 2
4. Если все значения Finish = True, то система в безопасном состоянии

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B
4	4

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Finish

P0	F
P1	T
P2	T

P1 P2

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation
2. Искать i, которое удовлетворяет условиям:
   1. Finish[i] = False
   2. Need[i] <= Work

   Если таких i нет, то переходим на шаг 4
3. Work = Work + Allocation[i]

   Finish[i] = True

   Переходим на шаг 2
4. Если все значения Finish = True, то система в безопасном состоянии

Алгоритм банкира (Banker's Algorithm)

Allocation

	A	B
P0	2	1
P1	0	0
P2	3	2

Max

	A	B
P0	5	3
P1	1	0
P2	4	2

Available

A	B
1	2

Work

A	B
6	5

Need

	A	B
P0	3	2
P1	1	0
P2	1	0

Finish

P0	T
P1	T
P2	T

P1 P2 P0

Алгоритм:

1. Инициализировать Work значениями из Available, инициализировать Finish значениями False, посчитать матрицу Need = Max – Allocation
2. Искать i, которое удовлетворяет условиям:
   1. Finish[i] = False
   2. Need[i] <= Work

   Если таких i нет, то переходим на шаг 4
3. Work = Work + Allocation[i]

   Finish[i] = True

   Переходим на шаг 2
4. Если все значения Finish = True, то **система в безопасном состоянии**

Спасибо за внимание!