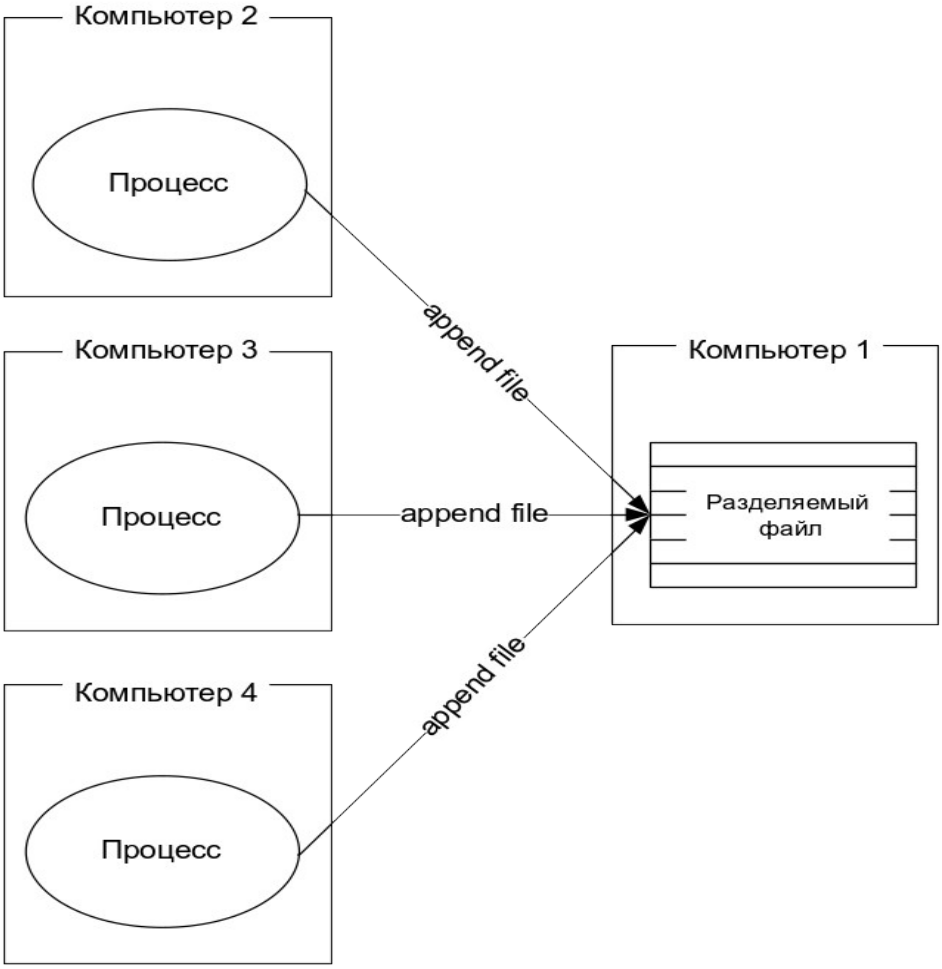
Лекция 04
РИС, III курс, ИСиТ

**Алгоритмы взаимного исключения**

1. **Напоминание.**

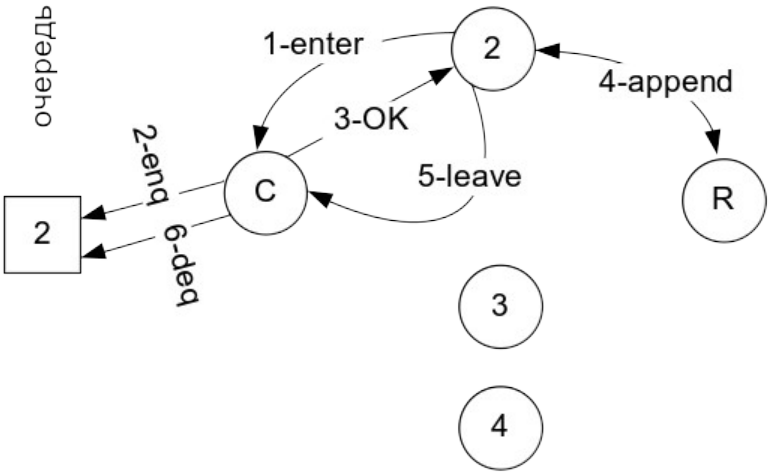

2. **Постановка задачи:** несколько распределенных процессов дописывают данные в конец общего файла на файловом сервере.



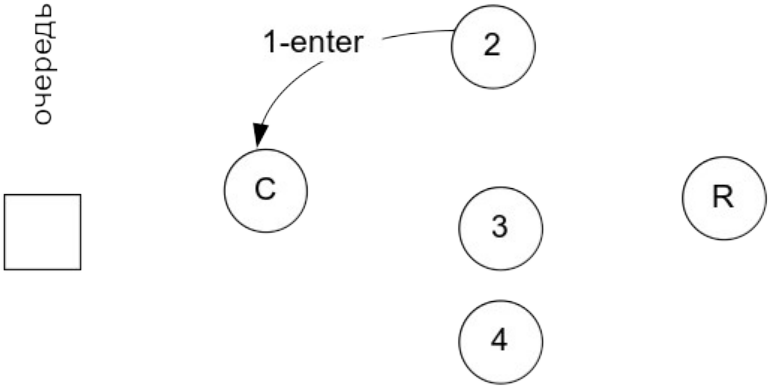

3. **Механизмы синхронизации:** критическая секция – механизм синхронизации (взаимного исключение) потоков в рамках одного процесса.

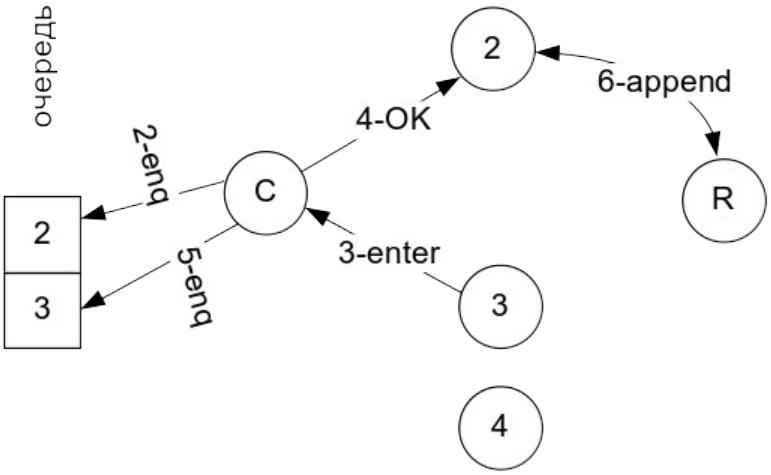
4. **Алгоритмы взаимного исключения:** централизованный алгоритм.

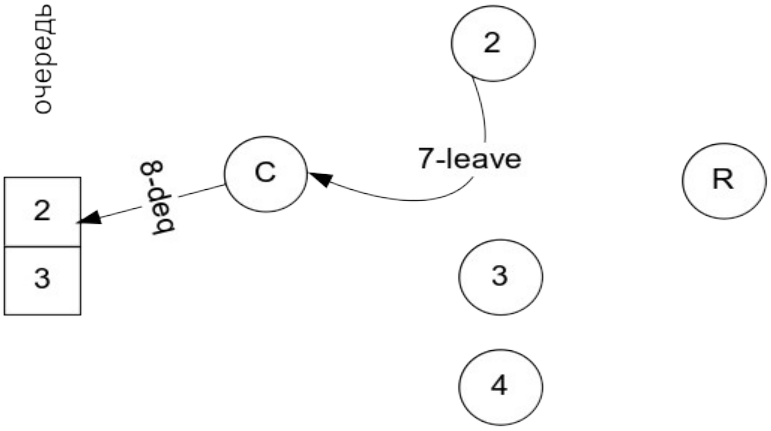
Один процесс, работающий с распределенным ресурсом

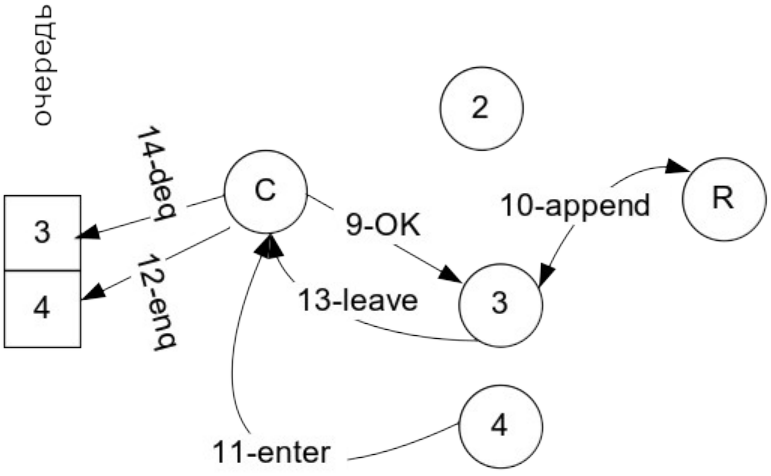


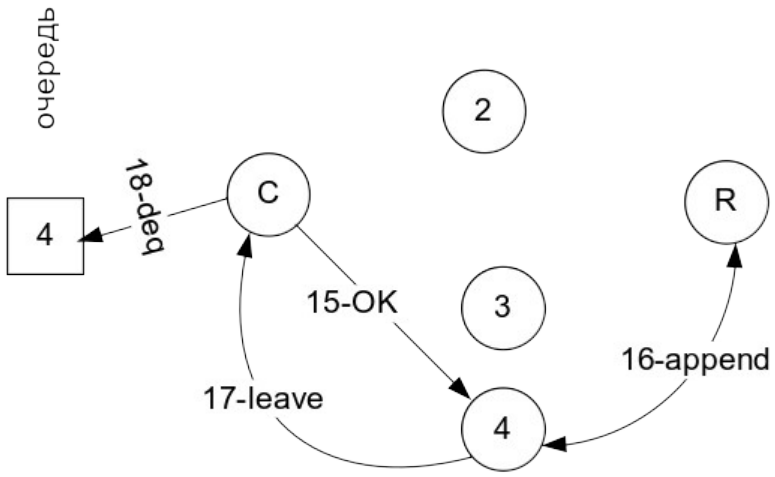

Три процесса, работающих с распределенным ресурсом




очередь
2
3
2-enq
5-enq
C
4-OK
3-enter
2
6-append
R
3
4


очередь
2
3
8-deq
C
7-leave
2
R
3
4


очередь
3
4
14-deq
12-enq
C
9-OK
13-leave
2
10-append
R
3
4
11-enter




5. **Алгоритмы взаимного исключения:** уровень доступа к ресурсу (файлы: open/close, write record).

6. **Алгоритмы взаимного исключения:** распределенный алгоритм. Надежная связь между узлами. Каждый узел, должен знать о других узлах участвующих в синхронизации.

**Принцип:**

- процесс не собирается входить: на все запросы **enter(метка времени)** отвечает **OK;**
- процесс собирается войти в критическую область: рассылает всем запрос **enter(метка времени)** и ждет от всех **OK;** если получил от всех **OK** выполняет действие;
- процесс собирается войти в критическую область (разослал **enter(метка времени)**) и одновременно получает **enter(метка времени)** от другого процесса: сравнивает свою метку времени и чужую; если чужая метка с меньшим временем – отсылает **OK** и ждет **OK** от всех других; если собственное время меньше, то чужой **enter(метка времени)** поместить в очередь, после выхода всем тем, кто в очереди – оправляется **OK** и соответствующие **enter(метка времени)** извлекаются из очереди;
- процесс вошел в критическую область: поступает **enter(метка времени)** – помещается в очередь; после

выхода всем тем, кто в очереди – оправляется **OK** и соответствующие **enter(метка времени)** извлекаются из очереди.

7. **Алгоритмы взаимного исключения:** кольцевой маркер. Узлы связываются в односвязный кольцевой список. По списку циркулирует маркер, предлагающий войти в критическую область; если входить в критическую область не требуется, то маркер отправляется дальше; если необходимо войти, то выполняется критическая операция и после выполнения маркер отправляется дальше.

8. **Сравнение:** общая проблема – разрыв связи; централизованный алгоритм – сбой координатора; распределенный алгоритм – сбой любого процесса; маркер – потеря маркера (сбой в одном из процессов).

9. **Лабораторная работа**

10.

```
struct CA // блок управления секцией
{
    char ipaddr[15];             //  ip-адрес (xxx.xxx.xxx.xxx)  координтора
    char resurce[20];            //  имя ресурса
    enum STATUS {
                 NOINIT,        // начальное состояние
                 INIT,          // выполнена инициализация
                 ENTER,         // выполнен вход в секцию
                 LEAVE,         // выполнен выход из секции
                 WAIT           // ожидание входа
                } status;       // состояние
};

CA  InitCA(   //  инициализировать критическую секцию   status-> INIT/NOINIT
            char ipaddr[15],            //  ip-адрес (xxx.xxx.xxx.xxx)  координтора
            char resurce[20]            //  имя ресурса
           );

bool EnterCA(  // войти в секцию   status-> ENTER/WAIT -> ENTER
             CA& ca   // блок управления секцией
            );

bool LeaveCA(  // покинуть секцию status-> LEAVE
             CA& ca   // блок управления секцией
            );

bool CloseCA(  // закрыть секцию  status-> NOINIT
             CA& ca   // блок управления секцией
            );
```