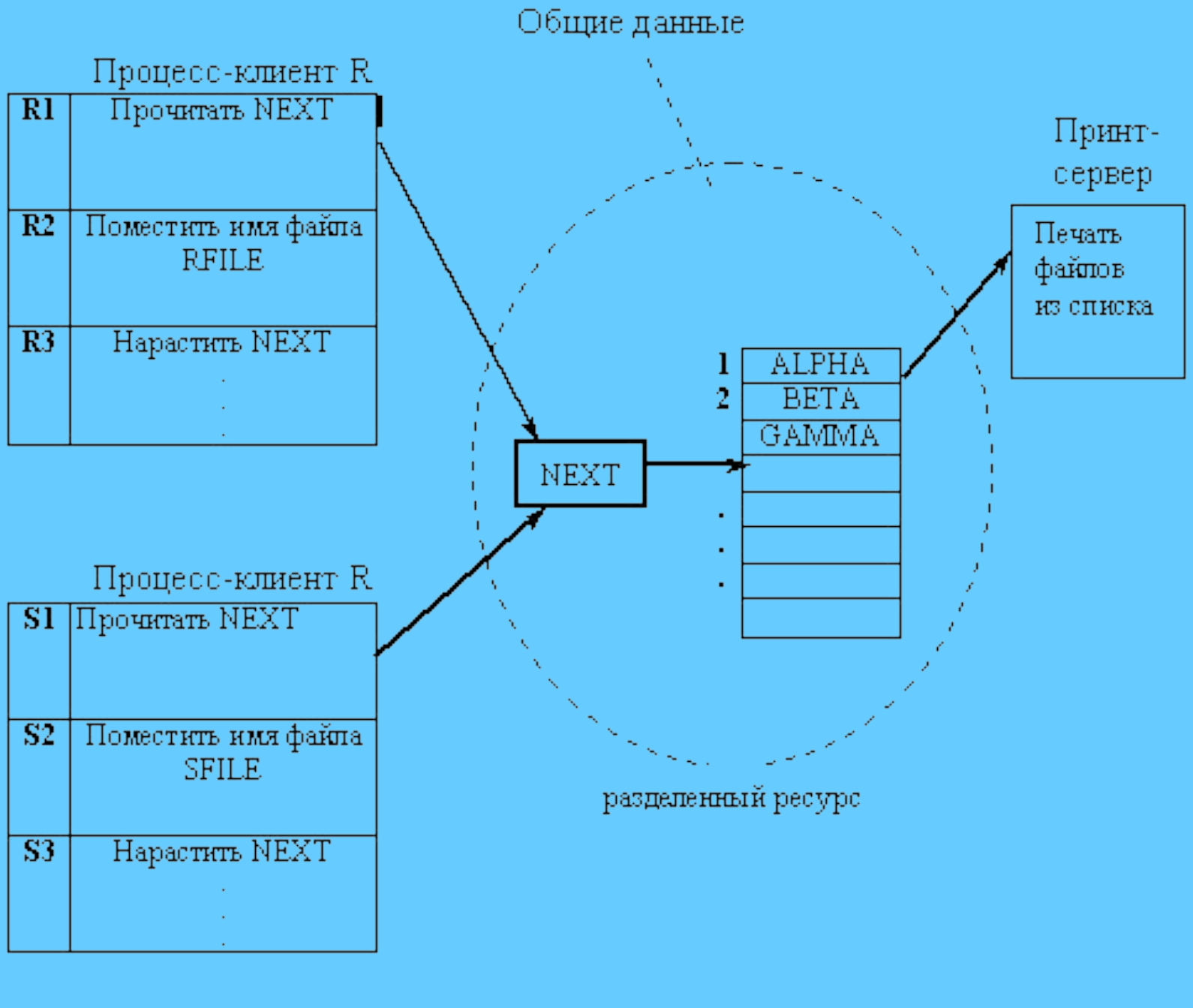
6. Средства синхронизации и взаимодействия процессов

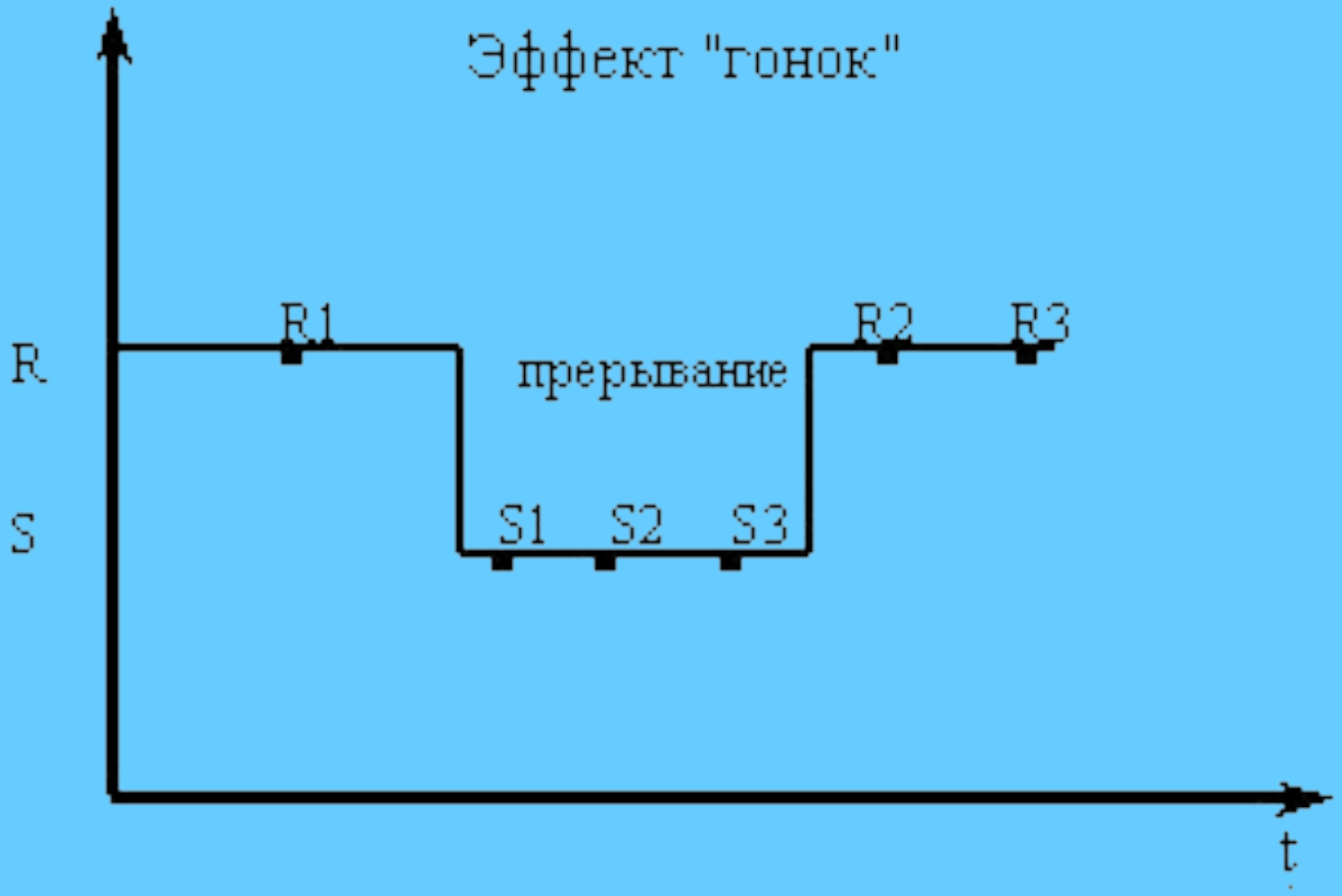
6.1. Проблема синхронизации

Процессам часто нужно взаимодействовать друг с другом, например, один процесс может передавать данные другому процессу, или несколько процессов могут обрабатывать данные из общего файла.

Во всех этих случаях возникает проблема синхронизации процессов, которая может решаться:

- приостановкой и активизацией процессов,
- организацией очередей,
- блокированием и освобождением ресурсов.


Общие данные
Процесс-клиент R
R1
Прочитать NEXT
R2
Поместить имя файла RFILE
R3
Нарастить NEXT
Принт-сервер
Печать файлов из списка
1
2
ALPHA
BETA
GAMMA
NEXT
Процесс-клиент R
S1
Прочитать NEXT
S2
Поместить имя файла SFILE
S3
Нарастить NEXT
разделенный ресурс



Пример необходимости синхронизации

6.2. Критическая секция

Важным понятием синхронизации процессов является понятие "критическая секция" программы (CS).

Критическая секция - это часть программы, в которой осуществляется доступ к разделяемым данным.

Чтобы исключить эффект гонок по отношению к некоторому ресурсу, необходимо обеспечить, чтобы в каждый момент в критической секции, связанной с этим ресурсом, находился максимум один процесс.

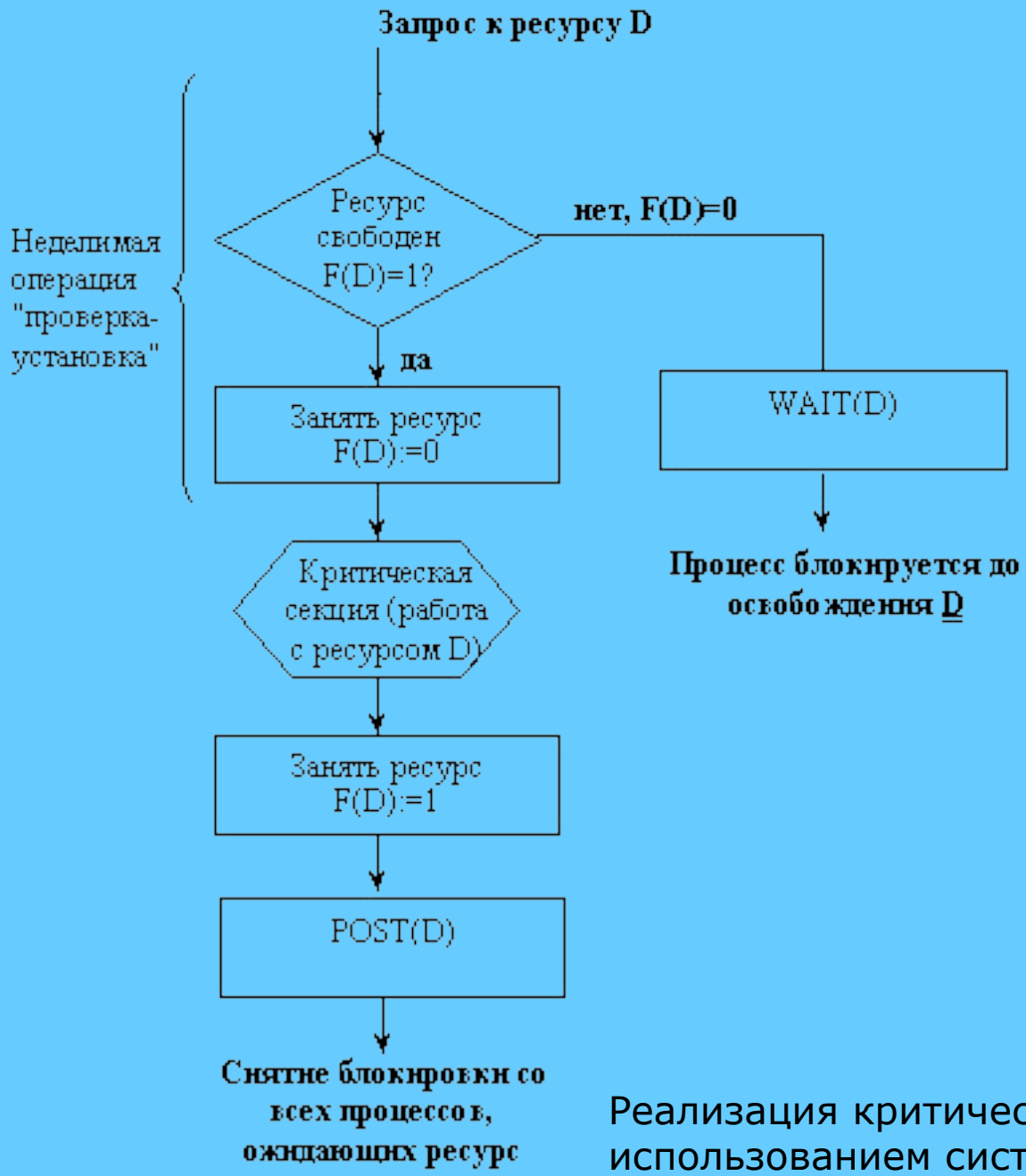
Этот прием называют взаимным исключением.

6.3. Синхронизация процессов на основе семафорных операций

Для устранения активного ожидания процесса CPU может быть использован так называемый аппарат событий.

С помощью этого средства могут решаться не только проблемы взаимного исключения, но и более общие задачи синхронизации процессов.

В разных ОС аппарат событий реализуется по своему, но в любом случае используются системные функции аналогичного назначения, которые условно назовем WAIT(x) и POST(x), где x - идентификатор некоторого события.



Реализация критической секции с использованием системных функций WAIT(D) и POST(D)

Если ресурс занят, то процесс не выполняет циклический опрос, а вызывает системную функцию WAIT(D), здесь D обозначает событие, заключающееся в освобождении ресурса D.

Функция WAIT(D) переводит активный процесс в состояние ОЖИДАНИЕ и делает отметку в его дескрипторе о том, что процесс ожидает события D.

Процесс, который в это время использует ресурс D, после выхода из критической секции выполняет системную функцию POST(D), в результате чего ОС просматривает очередь ожидающих процессов и переводит процесс, ожидающий события D, в состояние ГОТОВНОСТЬ.

Обобщающее средство синхронизации процессов предложил Дейкстра, который ввел два новых примитива.

В абстрактной форме эти примитивы, обозначаемые P и V, оперируют над целыми неотрицательными переменными, называемыми семафорами.

Пусть S такой семафор. Операции определяются следующим образом:

V(S) : переменная S увеличивается на 1 одним неделимым действием; выборка, инкремент и запоминание не могут быть прерваны, и к S нет доступа другим процессам во время выполнения этой операции.

P(S) : уменьшение S на 1, если это возможно. Если S=0, то невозможно уменьшить S и остаться в области целых неотрицательных значений, в этом случае процесс, вызывающий P-операцию, ждет, пока это уменьшение станет возможным.

6.4. Семафоры как счетчики ресурсов и синхронизаторы операций

Рассмотрим использование семафоров на классическом примере взаимодействия двух процессов, выполняющихся в режиме мультипрограммирования, один из которых пишет данные в буферный пул, а другой считывает их из буферного пула.

Пусть буферный пул состоит из N буферов, каждый из которых может содержать одну запись.

Процесс "писатель" должен приостанавливаться, когда все буфера оказываются занятыми, и активизироваться при освобождении хотя бы одного буфера.

Напротив, процесс "читатель" приостанавливается, когда все буферы пусты, и активизируется при появлении хотя бы одной записи.

Для процессов, совместно выполняющих общую работу, недостаточно, что они взаимно исключают друг друга при работе с разделяемыми переменными, им необходимо еще и передавать друг другу информацию.

Минимальной единицей передаваемой информации может быть простой временной сигнал.

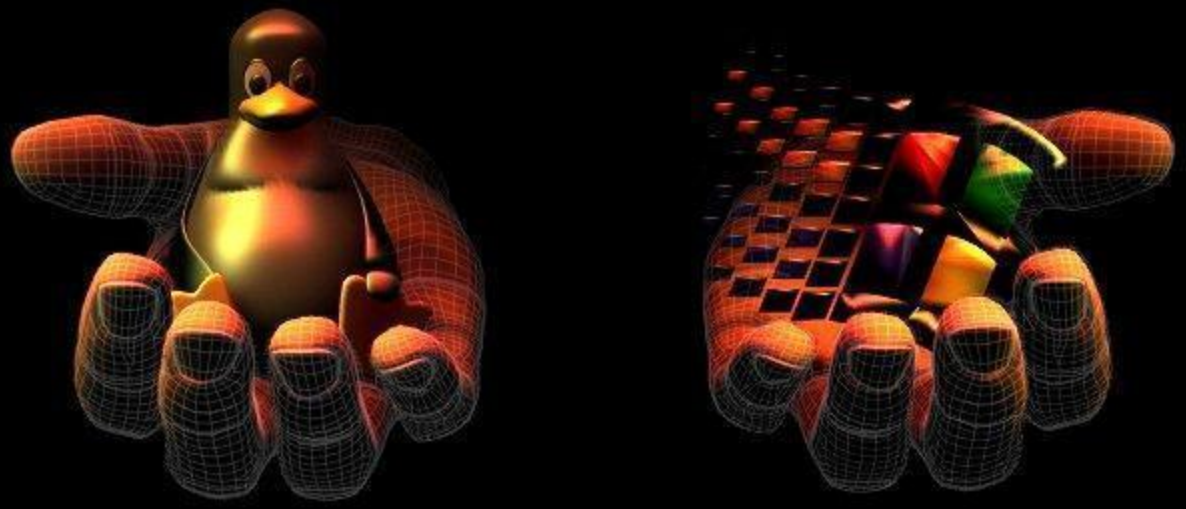
В этом случае действует следующее правило: процессу предоставляется возможность ждать, пока другой не сообщит о "свершении" определенного события.

Специально предусматривать взаимное исключение при доступе к буферу нет необходимости, т.к. в данном случае оно обеспечивается синхронизацией.

Итак - семафор может быть синхронизатором, координирующим производство и потребление ресурсов.

Процесс, потребляя ресурс, выполняет P-операцию над связанным с ресурсом семафором (т.е. P(S)), что означает изменение значения S в меньшую сторону.

Процесс производит ресурс, выполняя V(S)-операцию над тем же семафором.



Linux / Windows