Семафоры.

это защищенная переменная, которую можно опрашивать и менять только при помощи специальных операций P и V и операции инициализации семафора.

Операция P(S) (вход взаимоисключение):

если S>0
 то S := S – 1
 иначе ждать на S

Операция V(S) (выход взаимоисключение):

если есть процесс, ожидающий на S, разрешить одному продолжить работу
иначе S := S + 1

Семафоры бывают:

- Двоичные – 0, 1
- Считающие – N > 0

Их реализация бывает:

- Аппаратная
- Программная

Реализация взаимоисключений при помощи семафоров.

```
PROGRAMM
VAR S: (семафор) LOGICAL;
PROCEDURE 1
BEGIN
WHILE TRUE DO
        BEGIN
        начальные операторы
        P(S);
        крититеская область
        V(S);
        остальные операторы
        END
END
PROCEDURE 2
[ то же самое, что и в PROCEDURE 1 ]
BEGIN
        инициализция семафора (S,1)
        PARBEGIN
        PROCEDURE 1
        PROCEDURE 2
        PAREND
END
```

Синхронизация при помощи семафоров.

```
PROGRAMM
VAR S: (семафор) LOGICAL;
PROCEDURE 1
BEGIN
      начальные операторы
      P(S);
      остальные операторы
      END
END
PROCEDURE 2
BEGIN
      начальные операторы
      V(S);
      остальные операторы
      END
END
BEGIN
      S := 0;
      PARBEGIN
      PROCEDURE 1
      PROCEDURE 2
      PAREND
END
```

Пусть есть процесс, который ждет некоторого события PROCEDURE 1 и PROCEDURE 2, которая это событие фиксирует. Тогда надо использовать P и V следующим образом:

Реализация процесса потребитель-производитель при помощи семафоров.

```
PROGRAMM
VAR доступ: (семафор) LOGICAL;
VAR число записано: (семафор) LOGICAL;
PROCEDURE 1 (производитель)
BEGIN
WHILE TRUE DO
        BEGIN
        A= ...;
        P(доступ);
        BYFER = A;
        V(доступ);
        V(число записано);
        END
END
PROCEDURE 2(потребитель)
BEGIN
        WHILE TRUE DO
        BEGIN
        P(число записано);
        P(доступ);
        B=BYFER;
        V(доступ);
        обработка B;
        END
END
BEGIN
        доступ := 1;                    | инициали -
        число записано := 0;            | зация
        PARBEGIN
        ...
        PAREND
END
```

Пусть есть PROCEDURE 1, которая выдает данные в буфер (***производитель***), а PROCEDURE 2, что-то делает с числом в буфере (***потребитель***).

Как сделать, чтобы данные не пропадали?

Реализация семафоров через P и V.

Если есть команда testandset, то P и V реализуются достаточно просто (***с циклом активного ожидания***). Но это приводит к потере эффективности.(Состояние ***блокировки*** процесса - это не состояние ***активного ожидания***)

Применение считающих семафоров.

Используются в случае, если запрашиваемый ресурс может быть ***разделен на части*** (например, несколько принтеров). Тогда семафор будет считать ***число свободных единиц ресурса***.

Событийное программирование.

Формирование.

Событийно-ориентированный стиль ***программирования исторически сформировался в области разработки ОС, где естественно связывать понятие события с прерыванием. Когда происходит прерывание, ОС распознает, по какой причине оно было вызвано, и далее формирует событие, вызывающее реакцию программной системы.***

Определение.

Событийное программирования характеризуется тем, что любое действие происходит по некоторому событию.

Применение.

Наиболее очевидная область применения - реализация интерактивных взаимодействий программы с пользователем и решение других подобных задач (например, тех, которые требуют опроса датчиков состояния каких-либо технологических процессов).

Межпроцессорные коммуникации.

- ***Разделяемая память***
- ***Семафоры***
- ***Очереди сообщений***
- ***Программные каналы***
- ***Программные гнезд***
- ***Потоки***

Разделяемая память (shared memory).

Позволяют процессам иметь ***общие области виртуальной памяти***. Единицей разделяемой памяти являются сегменты, свойства которых зависят от аппаратных особенностей управления памятью.

Этот способ обеспечивает ***наиболее быстрый обмен*** данными между процессами.

POSIX – функции для работы с разделяемой памятью.

- **`shmget`** – создание разделяемого сегмента
- **`shmat`** – получение доступа к сегменту и размещение сегмента в памяти
- **`shmdt`** – удаление сегмента
- **`shmctl`** – выполнение управляющих действий (предписание удерживать сегмент в оперативной памяти и обратное предписание о снятии удержания)

Очереди сообщений (messages).

- Являются системным разделяемым ресурсом
- Каждая очередь сообщений имеет свой ***уникальный идентификатор***
- Процессы могут записывать сообщения в очередь и читать сообщения из очереди
- При этом процесс, пославший сообщение в очередь, не обязан ожидать приема этого сообщения другим процессом

POSIX – функции для работы с очередью сообщений.

- **`msgget`** – создать очереди сообщений
- **`msgsnd`** – послать сообщение в очередь, зная ее идентификатор
- **`msgrcv`** – прочитать сообщение, находящееся в очереди
- **`msgctl`** – удалить очередь

Программные каналы (pipes)

Однонаправленная передача данных другому процессу, причем только «***родственному***».

POSIX – функции для работы с программным каналом.

- **`pipe`** – создать не именованный программный канал
- **`open`** – создать именованный канал или получить доступ к уже существующему каналу
- **`read`** – прочитать из канала
- **`write`** – записать в канал

Программные гнезда (sockets)

Взаимодействие основано на модели "***клиент-сервер***". Процесс сервер "слушает" свое программное гнездо, а процесс-клиент пытается общаться с процессом-сервером через другое программное гнездо.

POSIX – функции для работы с программным гнездом.

- ***socket*** – создать новое программное гнездо
- ***bind*** – связать ранее созданный программное гнездо с именем
- ***connect*** – запросить систему связаться с существующим программным гнездом (у процесса-сервера)
- ***listen*** – информировать систему о том, что процесс-сервер планирует установление виртуальных соединений через указанное гнездо
- ***accept*** – для выборки процессом-сервером запроса на установление соединения с указанным программным гнездом
- ***send/sendto*** – послать сообщение
- ***recv/recvfrom*** – принять сообщение
- ***shutdown*** – удалить соединение