Семафоры в UNIX. Отличие операций над UNIX семафорами от классических операций

На предыдущем семинаре мы говорили о необходимости синхронизации работы процессов для их корректного взаимодействия через разделяемую память. Как упоминалось на лекции, одним из первых механизмов, предложенных для синхронизации поведения процессов, стали семафоры, концепцию которых описал Дейкстра (Dijkstra) в 1965 году. При разработке средств System V IPC семафоры вошли в их состав как неотъемлемая часть. Следует отметить, что набор операций над семафорами System V IPC отличается от классического набора операций **{P, V}**, предложенного Дейкстрой. Он насчитывает три операции:

- **A(S, n)** - увеличить значение семафора **S** на величину **n > 0**; Название операции происходит от слова «Add».
- **D(S, n)** - пока значение семафора **S < n** (**n** должно быть больше 0), процесс блокируется. Далее
  S = S - n;
  Название операции происходит от слова «Decrease».
- **Z(S)** - процесс блокируется до тех пор, пока значение семафора **S** не станет равным **0**. Название операции происходит от слова «Zero».

Начальное значение IPC семафоров не определено в стандарте POSIX. Во многих UNIX-подобных системах и почти во всех Linux семафоры инициируются нулевым значением, чем мы и воспользуемся. Строгое создание семафоров IPC с выставлением им ненулевого значения после создания достаточно трудоемко (см., например, Стивенса), и мы его рассматривать не будем.

Легко видеть, что классической операции **P(S)** соответствует операция **D(S,1)**, а классической операции **V(S)** соответствует операция **A(S,1)**. Аналогом ненулевой инициализации семафоров Дейкстры значением **n** может служить выполнение операции **A(S,n)** сразу после эксклюзивного создания семафора **S**, с обеспечением атомарности создания семафора и её выполнения посредством другого семафора, или до начала взаимодействия процессов. Мы показали, что классические семафоры реализуются через семафоры System V IPC. Обратное не является верным при использовании только одного семафора. Используя операции **P(S)** и **V(S)**, мы не сумеем реализовать операцию **Z(S)**. Для реализации операции **Z(S)** необходимо использовать как минимум два классических семафора и разделяемые переменные.

Поскольку IPC семафоры являются составной частью средств System V IPC, то для них верно всё, что говорилось об этих средствах в целом на предыдущем семинаре. IPC семафоры являются средством связи с непрямой адресацией, требуют инициализации для организации взаимодействия процессов и специальных действий для освобождения системных ресурсов по его окончании. Пространством имён IPC семафоров является множество значений ключа, генерируемых с помощью функции

`ftok()`. Для совершения операций над семафорами системным вызовам в качестве параметра передаются IPC дескрипторы семафоров, однозначно идентифицирующих их во всей вычислительной системе, а вся информация о семафорах располагается в адресном пространстве ядра операционной системы. Это позволяет организовывать через семафоры взаимодействие процессов, даже не находящихся в системе одновременно.

Создание массива семафоров или доступ к уже существующему. Системный вызов semget()

В целях экономии системных ресурсов операционная система UNIX позволяет создавать не по одному семафору для каждого конкретного значения ключа, а связывать с ключом целый массив семафоров (ограничение на количество семафоров в массиве задается при генерации операционной системы, впоследствии это количество может быть уменьшено системным администратором). Для создания массива семафоров, ассоциированного с определенным ключом, или доступа по ключу к уже существующему массиву используется системный вызов `semget()`, являющийся аналогом системного вызова `shmget()` для разделяемой памяти, который возвращает значение IPC дескриптора для этого массива. При этом существуют те же способы создания и доступа, что и для разделяемой памяти.

Выполнение операций над семафорами. Системный вызов semop()

Для выполнения операций **A**, **D** и **Z** над семафорами из массива используется системный вызов `semop()`, обладающий довольно сложной семантикой. Разработчики System V IPC явно перегрузили этот вызов, применяя его не только для выполнения всех трёх операций, но ещё и для нескольких семафоров в массиве IPC семафоров одновременно. Для правильного использования этого вызова необходимо выполнить следующие действия:

1. Определиться, для каких семафоров из массива вы хотите выполнить операции. Необходимо иметь в виду, что все операции реально совершаются только перед успешным возвращением из системного вызова, т.е. если вы хотите выполнить операции **A(S_1,5)** и **Z(S_2)** в одном вызове и оказалось, что S_2 **!= 0**, то значение семафора S_1 не будет изменено до тех пор, пока значение S_2 не станет равным **0**. Порядок выполнения операций в случае, когда процесс не переходит в состояние **ожидание** не определён. Так, например, при одновременном выполнении операций **A(S_1,1)** и **D(S_2,1)** в случае $S_2 > 1$ неизвестно, что выполнится раньше - уменьшение значения семафора S_2 или увеличение значения семафора S_1. Если порядок является для вас существенным, лучше применить несколько вызовов вместо одного.

2. После того как вы определились с количеством семафоров и совершаемыми операциями, необходимо завести в программе массив из элементов типа `struct sembuf` с размерностью равной определённому количеству семафоров (если операция совершается только над одним семафором можно, естественно, обойтись просто переменной). Каждый элемент этого массива будет соответствовать операции над одним семафором.
3. Заполнить элементы массива. В поле `sem_flg` каждого элемента занести значение **0** (с другими значениями флагов мы на семинарах работать не будем). В поля `sem_num` и `sem_op` занести номера семафоров в массиве IPC семафоров и соответствующие коды операций. Семафоры нумеруются, начиная с **0**. Если у вас в массиве всего один семафор, то он будет иметь номер **0**. Операции кодируются так:
    - для выполнения операции **A(S,n)** значение поля `sem_op` должно быть равно **n**.
    - для выполнения операции **D(S,n)** значение поля `sem_op` должно быть равно **-n**.
    - для выполнения операции **Z(S)** значение поля `sem_op` должно быть равно **0**.
4. В качестве второго параметра системного вызова `semop()` указать адрес заполненного массива, а в качестве третьего параметра - ранее определённое количество семафоров, над которыми совершаются операции.

Важное дополнение: Использование системного вызова `semop()` для одновременного атомарного выполнения нескольких операций над одним семафором является категорически недопустимым из-за непереносимости! Так, например, если вы хотите выполнить над семафором **S**, который в данный момент времени имеет значение 0, одновременно операции **Z(S)** и **A(S,1)**, то поведение процесса непредсказуемо. В некоторых ОС выполнение операций в порядке **Z(S) && A(S,1)** (т.е. в нулевом элементе массива сидит код операции 0, а в первом — код операции 1 для одного и того же номера семафора) пройдёт без блокировки, а выполнение операций в порядке **A(S,1) && Z(S)** заблокируется, а в некоторых наоборот!

Для иллюстрации вышесказанного давайте рассмотрим простейшие программы, синхронизирующие свои действия с помощью семафоров (файлы **09-1a.c** и **09-1b.c**). Первая программа выполняет над семафором **S** операцию **D(S,1)**, вторая программа выполняет над тем же семафором операцию **A(S,1)**. Если семафор не существует в системе, любая программа создает его перед выполнением операции. Поскольку при создании семафор всегда инициируется **0**, то программа 1 может работать без блокировки только после запуска программы 2.

Удаление набора семафоров из системы с помощью команды ipcrm или системного вызова semctl()

Как вы видели из примеров, массив семафоров может продолжать существовать в системе и после завершения использовавших его процессов, а семафоры будут сохранять своё значение. Это способно привести к неправильному поведению программ, предполагающих, что семафоры были только что созданы и, следовательно, имеют нулевое значение. Необходимо удалять семафоры из системы перед запуском таких программ или перед их завершением. Для удаления семафоров можно воспользоваться командами `ipcs` и `ipcrm`, рассмотренными на предыдущем семинаре. Команда `ipcrm` в этом случае должна иметь вид

```
$ ipcrm sem <IPC идентификатор>
```

Для этой же цели вы можете применять системный вызов `semctl()`, который умеет выполнять и другие операции над массивом семафоров, но их рассмотрение выходит за рамки нашего курса.

Задачи на семинар

***Задача 1 (1 балл)**:*

*Измените примеры 09-1, так чтобы первая программа (**09-1a.c**) могла работать без блокировки после не менее 5 запусков второй программы (**09-1b.c**).*

***Задача 2 (10 баллов)**:*

*На прошлом занятии вы установили, что любые неатомарные операции, связанные с изменением содержимого разделяемой памяти, представляют собой критическую секцию процесса или нити исполнения. Модифицируйте программы **07-3a.c** и **07-3b.c**, которые иллюстрировали некорректное взаимодействие процессов через разделяемую память, обеспечив с помощью одного IPC семафора взаимоисключения для их правильной работы.*

*Аналогично задаче с чтением из файла (**5.1**), данная задача не принимается, пока вы неверно определяете границы критической секции, в случае последующего исправления принимается со штрафом в 20%.*

***Примечание**:*

Считать, что инициализация разделяемой памяти и задание начального значения семафора происходят при первом отдельном запуске одной из программ. В рамках задачи взаимодействие процессов (а следовательно, и синхронизация) происходит только в ветке с длинным пустым циклом.