Семинары 9-10. Семафоры

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».
  Изначально все IPC семафоры инициируются нулевым значением.

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. Вновь созданные семафоры инициируются нулевым значением.

Прототип системного вызова

```
#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>



int semget(key_t key, int nsems, int semflg);
```

Описание системного вызова

Системный вызов semget предназначен для выполнения операции доступа к массиву IPCсемафоров и, в случае её успешного завершения, возвращает дескриптор System V IPC для этого массива (целое неотрицательное число, однозначно характеризующее массив семафоров внутри вычислительной системы и использующееся в дальнейшем для других операций с ним).

Параметр key является ключом System V IPC для массива семафоров, т. е. фактически его именем из пространства имен System V IPC.

В качестве значения этого параметра может использоваться значение ключа, полученное с помощью функции ftok(), или специальное значение IPC_PRIVATE. Использование значения IPC_PRIVATE всегда приводит к попытке создания нового массива семафоров с ключом, который не совпадает со значением ключа ни одного из уже существующих массивов и не может быть получен с помощью функции ftok() ни при одной комбинации её параметров.

Параметр nsems определяет количество семафоров в создаваемом или уже существующем массиве. В случае, если массив с указанным ключом уже имеется, но его размер не совпадает с указанным в параметре nsems, констатируется возникновение ошибки.

Параметр semflg – флаги – играет роль только при создании нового массива семафоров и определяет права различных пользователей при доступе к массиву, а также необходимость создания нового массива и поведение системного вызова при попытке создания.

Он является некоторой комбинацией (с помощью операции побитовое или – «|») следующих предопределённых значений и восьмеричных прав доступа:

IPC_CREAT – если массива для указанного ключа не существует, он должен быть создан;

IPC_EXCL – применяется совместно с флагом IPC_CREAT. При совместном их использовании и существовании массива с указанным ключом, доступ к массиву не производится и констатируется ошибка, при этом переменная errno, описанная в файле <errno.h>, примет значение EEXIST

0400 – разрешено чтение для пользователя, создавшего массив;

0200 – разрешена запись для пользователя, создавшего массив;

0040 – разрешено чтение для группы пользователя, создавшего массив;

0020 – разрешена запись для группы пользователя, создавшего массив;

0004 – разрешено чтение для всех остальных пользователей;

0002 – разрешена запись для всех остальных пользователей.

Вновь созданные семафоры инициируются нулевым значением.

```
Системный вызов возвращает значение дескриптора System V IPC для массива семафоров при
нормальном завершении и значение -1 при возникновении ошибки.
```

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.

```
Прототип системного вызова

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>



int semop(int semid, struct sembuf *sops, int nsops);









Описание системного вызова
```

```
Системный вызов semop предназначен для выполнения операций A, D и Z. Данное описание не
является полным описанием системного вызова, а ограничивается рамками текущего курса.
Для полного описания обращайтесь к UNIX Manual.



Параметр semid является дескриптором System V IPC для набора семафоров, т. е. значением,
которое вернул системный вызов semget() при создании набора семафоров или при его поиске
по ключу.

Каждый из nsops элементов массива, на который указывает параметр sops, определяет
операцию, которая должна быть совершена над каким-либо семафором из массива IPC
семафоров, и имеет тип структуры struct sembuf, в которую входят следующие переменные:

short sem_num – номер семафора в массиве IPC-семафоров (нумеруются, начиная с 0);

short sem_op – выполняемая операция;

short sem_flg – флаги для выполнения операции. В нашем курсе всегда будем считать эту
переменную равной 0.

Значение элемента структуры sem_op определяется следующим образом:

- для выполнения операции A(S,n) значение должно быть равно n;

- для выполнения операции D(S,n) значение должно быть равно -n;

- для выполнения операции Z(S) значение должно быть равно 0.

Семантика системного вызова подразумевает, что все операции будут в реальности выполнены
над семафорами только перед успешным возвращением из системного вызова. Если при
выполнении операций D или Z процесс перешёл в состояние ожидания, то он может быть
выведен из этого состояния при возникновении следующих форс-мажорных ситуаций:

- массив семафоров был удален из системы;

- процесс получил сигнал, который должен быть обработан.

В этом случае происходит возврат из системного вызова с констатацией ошибочной ситуации.

Системный вызов возвращает значение 0 при нормальном завершении и значение -1 при
возникновении ошибки.
```

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 — не предсказуемо. 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 **ex1a.c** и **ex1b.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.

```
Прототип системного вызова

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>



int semctl(int semid, int semnum, int cmd, union semun arg);



Описание системного вызова

Системный вызов semctl предназначен для получения информации о массиве IPC семафоров,
изменения его атрибутов и удаления его из системы.

Данное описание не является полным описанием системного вызова, а ограничивается рамками
текущего курса.

Для изучения полного описания обращайтесь к UNIX Manual.

В нашем курсе мы будем применять системный вызов semctl только для удаления массива
семафоров из системы. Параметр semid является дескриптором System V IPC для массива
семафоров, т. е. значением, которое вернул системный вызов semget() при создании массива
или при его поиске по ключу.

В качестве параметра cmd в рамках нашего курса мы всегда будем передавать значение
IPC_RMID – команду для удаления сегмента разделяемой памяти с заданным идентификатором.
Параметры semnum и arg для этой команды не используются, поэтому мы всегда будем
подставлять вместо них значение 0.

Если какие-либо процессы находились в состоянии ожидание для семафоров из удаляемого
массива при выполнении системного вызова semop(), то они будут разблокированы и вернутся
из вызова semop() с индикацией ошибки.

Системный вызов возвращает значение 0 при нормальном завершении и значение -1 при
возникновении ошибки.
```

Понятие о POSIX семафорах

В стандарте POSIX вводятся другие семафоры, полностью аналогичные семафорам Дейкстры. Для инициализации значения таких семафоров применяется функция *sem_init()*, аналогом операции **P** служит функция *sem_wait()*, а аналогом операции **V** - функция *sem_post()*. К

сожалению, во многих версиях Linux такие семафоры реализованы только для нитей исполнения внутри одного процесса, и поэтому подробно мы на них останавливаться не будем.

На семинаре 5, когда вы изучали связь родственных процессов через pipe, мы говорили о том, что pipe является однонаправленным каналом связи, и что для организации связи через один pipe в двух направлениях необходимо использовать механизмы взаимной синхронизации процессов.