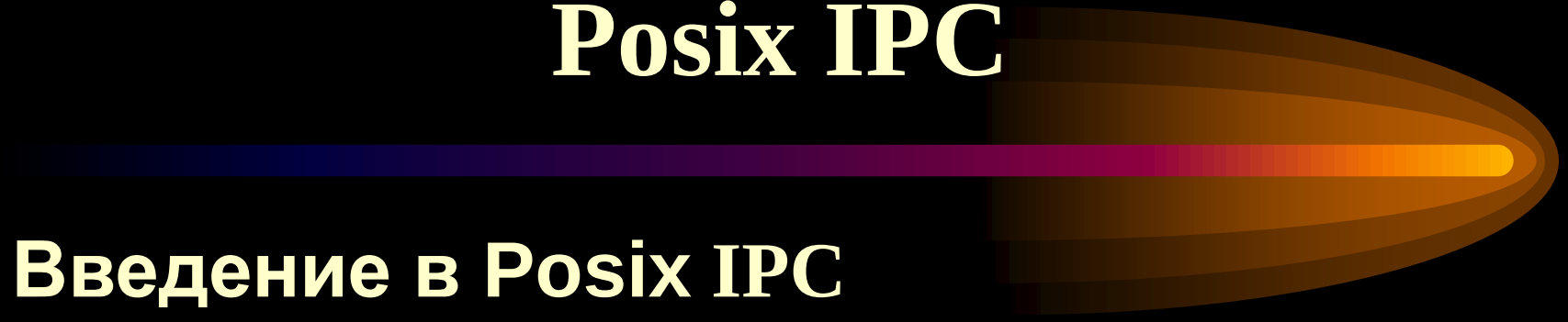
Posix IPC



- Введение в Posix IPC
- Имена объектов Posix IPC
- Создание и открытие объектов IPC
- Разрешения IPC

Введение в Posix IPC

- Из имеющихся типов IPC следующие три могут быть отнесены к Posix IPC, то есть к методам взаимодействия процессов, соответствующим стандарту Posix:
  - очереди сообщений Posix;
  - семафоры Posix;
  - разделяемая память Posix.

Введение в Posix IPC

- Эти три вида IPC обладают общими свойствами, и для работы с ними используются похожие функции. Далее речь пойдет об общих требованиях к полным именам файлов, используемых в качестве идентификаторов, о флагах, указываемых при открытии или создании объектов IPC, и о разрешениях на доступ к ним.

Введение в Posix IPC

- Полный список функций, используемых для работы с данными типами IPC, приведен в табл.

	Очереди сообщений	Семафоры	Разделяемая память
Заголовочный файл	`<mqueue.h>`	`<semaphore.h>`	`<sys/mman.h>`
Функции для создания, открытия и удаления	`mq_open`, `mq_close`, `mq_unlink`	`sem_open`, `sem_close`, `sem_unlink`, `sem_init`, `sem_destroy`	`shm_open`, `shm_unlink`

Введение в Posix IPC

	Очереди сообщений	Семафоры	Разделяемая память
Операции управления	`mq_getattr`, `mq_setattr`		`ftruncate`, `fstat`
Операции IPC	`mq_send`, `mq_receive`, `mq_notify`	`sem_wait`, `sem_trywait`, `sem_post`, `sem_getvalue`	`mmap`, `munmap`

- Три типа IPC стандарта Posix имеют идентификаторы (имена), соответствующие этому стандарту.

Введение в Posix IPC

- **Имена объектов Posix IPC**
- Имя IPC передается в качестве первого аргумента одной из трех функций: **mq_open**, **sem_open** и **shm_open**, причем оно не обязательно должно соответствовать реальному файлу в файловой системе. Стандарт Posix.1 накладывает на имена IPC следующие ограничения:
  - Имя должно соответствовать существующим требованиям к именам файлов (не превышать в длину **PATHMAX** (см. **/usr/include/limits.h**) байтов, включая завершающий символ с кодом 0).

Введение в Posix IPC

  - Если имя начинается со слэша (/), вызов любой из этих функций приведет к обращению к одной и той же очереди. В противном случае результат зависит от реализации ОС.
  - Интерпретация дополнительных слэшей в имени зависит от реализации ОС.
- Таким образом, для лучшей переносимости имена должны начинаться со слэша (/) и не содержать в себе дополнительных слэшей.

Введение в Posix IPC

- В современных Linux-системах объекты Posix создаются в виртуальной файловой системе. Эта файловая система может быть смонтирована суперпользователем, например, для очередей сообщений следующими командами:
- **[root@CentOS]# mkdir /dev/mqueue**
- **[root@CentOS]# mount -t mqueue none /dev/mqueue**
- После монтирования объекты Posix можно просматривать и удалять обычными утилитами для работы с файлами **ls**, **cat**, **rm**:
- **[gun@CentOS]$ ls -l /dev/mqueue**
- **-rw-------. 1 gun gun 80 Янв 30 14:25 tmp.123**

Введение в Posix IPC

- **Создание и открытие объектов IPC**
- Все три функции, используемые для создания или открытия объектов IPC: **mq_open**, **sem_open** и **shm_open**, — принимают специальный флаг **oflag** в качестве второго аргумента. Он определяет параметры открытия запрашиваемого объекта аналогично второму аргументу стандартной функции **open**. Все константы, из которых можно формировать этот аргумент, приведены в табл.:

Введение в Posix IPC

Описание	mq_open	sem_open	shm_open
Только чтение	O_RDONLY		O_RDONLY
Только запись	O_WRONLY		
Чтение и запись	O_RDWR		O_RDWR
Создать, если не существует	O_CREAT	O_CREAT	O_CREAT
Исключающее создание	O_EXCL	O_EXCL	O_EXCL
Без блокировки	O_NONBLOCK		
Сократить (truncate) существующий			O_TRUNC

- Первые три строки описывают тип доступа к создаваемому объекту: только чтение, только запись, чтение и запись.

Введение в Posix IPC

- Очередь сообщений может быть открыта в любом из трех режимов доступа, тогда как для семафора указание этих констант не требуется (для любой операции с семафором требуется доступ на чтение и запись). Наконец, объект разделяемой памяти не может быть открыт только на запись.
- Указание прочих флагов из табл. не является обязательным.
- При создании новой очереди сообщений, семафора или сегмента разделяемой памяти требуется указание по крайней мере одного дополнительного аргумента, определяющего режим. Этот аргумент указывает биты разрешения на доступ к файлу и формируется путем побитового логического сложения констант из табл. :

Введение в Posix IPC

Константа	Описание
S_IRUSR	Владелец — чтение
S_IWUSR	Владелец — запись
S_IRGRP	Группа — чтение
S_IWGRP	Группа — запись
S_IROTH	Прочие — чтение
S_IWOTH	Прочие — запись

- Как и со вновь созданным файлом, при создании очереди сообщений, семафора или сегмента разделяемой памяти им присваивается идентификатор пользователя, соответствующий действующему идентификатору пользователя процесса.

Введение в Posix IPC

- Идентификатор группы пользователей семафора или сегмента разделяемой памяти устанавливается равным действующему групповому идентификатору процесса. Групповой идентификатор очереди сообщений всегда устанавливается равным действующему групповому идентификатору процесса.
- Флаг **O_CREAT** обеспечивает создание очереди сообщений, семафора или сегмента разделяемой памяти, если таковой еще не существует.
- Флаг **O_EXCL**, если он указан одновременно с **O_CREAT**, требует создания новой очередь сообщений, семафора или объекта разделяемой памяти только в том случае, если таковой не существует.

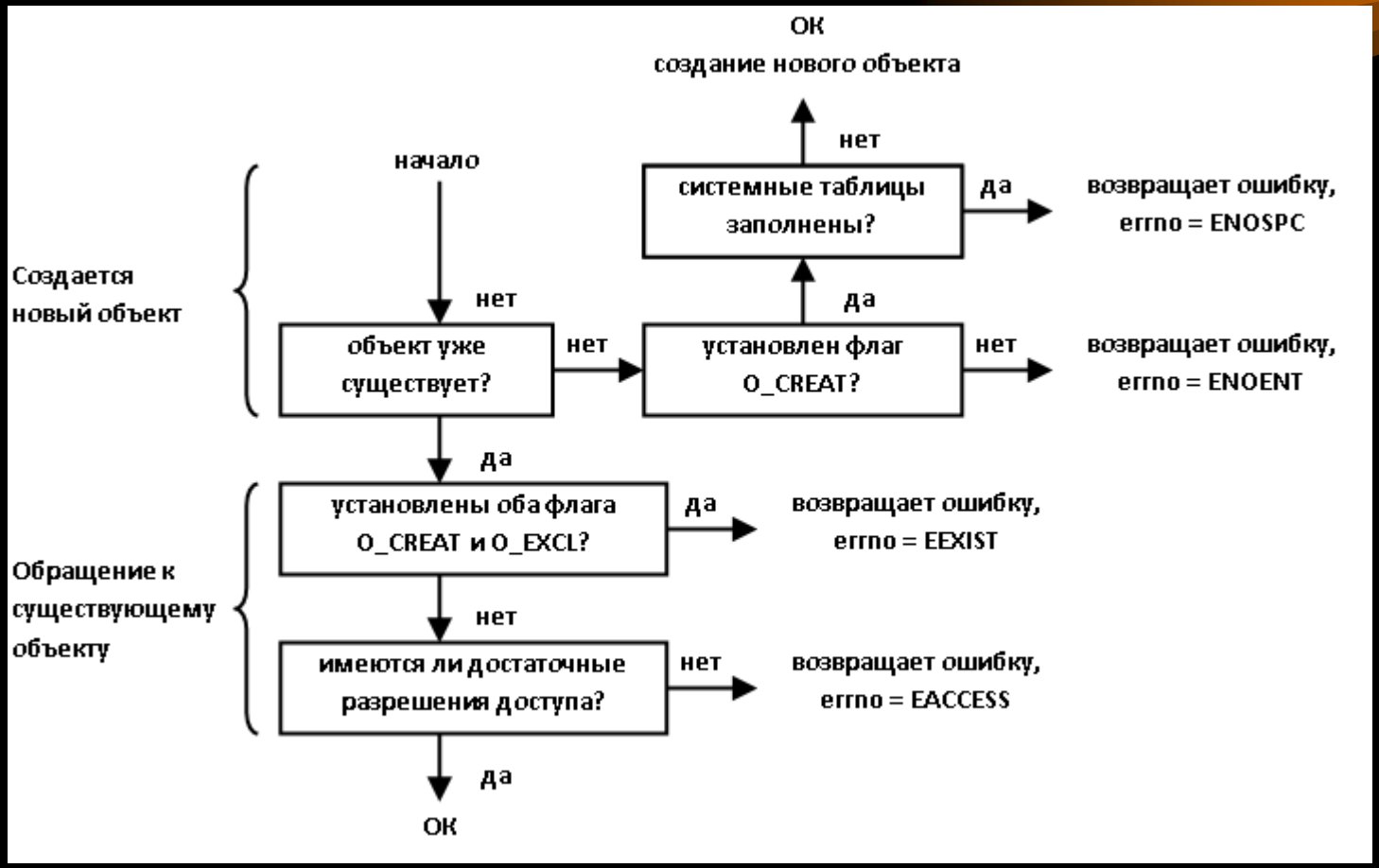
Введение в Posix IPC

- Если объект уже существует и указаны флаги **O_CREAT | O_EXCL**, возвращается ошибка **EEXIST**.
- Проверка существования очереди сообщений, семафора или сегмента разделяемой памяти и его создание (в случае отсутствия) должны производиться только одним процессом. Два аналогичных флага имеются и в System V IPC, они описаны в параграфе 6.3.1.
- Флаг **O_NONBLOCK** создает очередь сообщений без блокировки. Блокировка обычно устанавливается для считывания из пустой очереди или записи в полную очередь. Об этом более подробно рассказано в параграфе, посвященном функциям **mq_send** и **mq_receive.**

Введение в Posix IPC

- Флаг **O_TRUNC** в случае, если уже существующий сегмент общей памяти открыт на чтение и запись, указывает на необходимость сократить его размер до 0.
- На рис. ниже показана логическая диаграмма последовательности логических операций при открытии объекта IPC. Другой (текстовый) подход к алгоритму на рис. представлен в табл.
- Обратите внимание, что в средней строке табл., где задан только флаг **O_CREAT**, мы не получаем никакой информации о том, был ли создан новый объект или открыт существующий.

Введение в Posix IPC



Введение в Posix IPC

Аргумент `oflag`	Объект не существует	Объект уже существует
Нет специальных флагов	Ошибка, **`errno=ENOENT`**	OK, открывается существующий объект
`O_CREAT`	OK, создается новый объект	OK, открывается существующий объект
`O_CREAT|O_EXCL`	OK, создается новый объект	Ошибка, **`errno=EEXIST`**

- Новая очередь сообщений, именованный семафор или сегмент разделяемой памяти создается функциями **mq_open**, **sem_open** и **shm_open**, при условии, что аргумент **oflag** содержит константу **O_CREAT**. Согласно табл., любому из данных типов IPC присваиваются определенные права доступа, аналогичные разрешениям доступа к файлам в Unix.

Введение в Posix IPC

- При открытии существующей очереди сообщений, семафора или сегмента разделяемой памяти теми же функциями (в случае, когда не указан флаг **O_CREAT** или указан **O_CREAT** без **O_EXCL** и объект уже существует) производится проверка разрешений:
- 1. Проверяются биты разрешений, присвоенные объекту IPC при создании.
- 2. Проверяется запрошенный тип доступа (**O_RDONLY**, **O_WRONLY**, **O_RDWR**).
- 3. Проверяется действующий идентификатор пользователя вызывающего процесса, действующий групповой идентификатор процесса.

Введение в Posix IPC

- Большинством систем Unix производятся следующие конкретные проверки:
- 1. Если действующий идентификатор пользователя для процесса есть 0 (привилегированный пользователь), доступ будет разрешен.
- 2. Если действующий идентификатор пользователя процесса совпадает с идентификатором владельца объекта IPC и если соответствующий бит разрешения для пользователя установлен, доступ разрешен, иначе в доступе отказывается.
- Под соответствующим битом разрешения подразумевается, например, бит разрешения на чтение, если процесс открывает объект только для чтения.

Введение в Posix IPC

- Если процесс открывает объект для записи, должен быть установлен соответственно бит разрешения на запись для владельца.
- 3. Если действующий идентификатор группы процесса совпадает с групповым идентификатором объекта IPC и если соответствующий бит разрешения для группы установлен, доступ будет разрешен, иначе в доступе отказывается.
- 4. Если соответствующий бит разрешения доступа для прочих пользователей установлен, доступ будет разрешен, иначе в доступе будет отказано.

Введение в Posix IPC

- Эти четыре проверки производятся в указанном порядке. Следовательно, если процесс является владельцем объекта IPC (шаг 2), доступ разрешается или запрещается на основе одних только разрешений пользователя (владельца). Разрешения группы при этом не проверяются. Аналогично, если процесс не является владельцем объекта IPC, но принадлежит к нужной группе, доступ разрешается или запрещается на основе разрешений группы — разрешения для прочих пользователей при этом не проверяются.