Разделяемая память System V

- Введение
- Функция shmget
- Функция shmat
- Функция shmdt
- Функция shmctl
- Простые программы
- Ограничения, накладываемые на разделяемую память

Введение

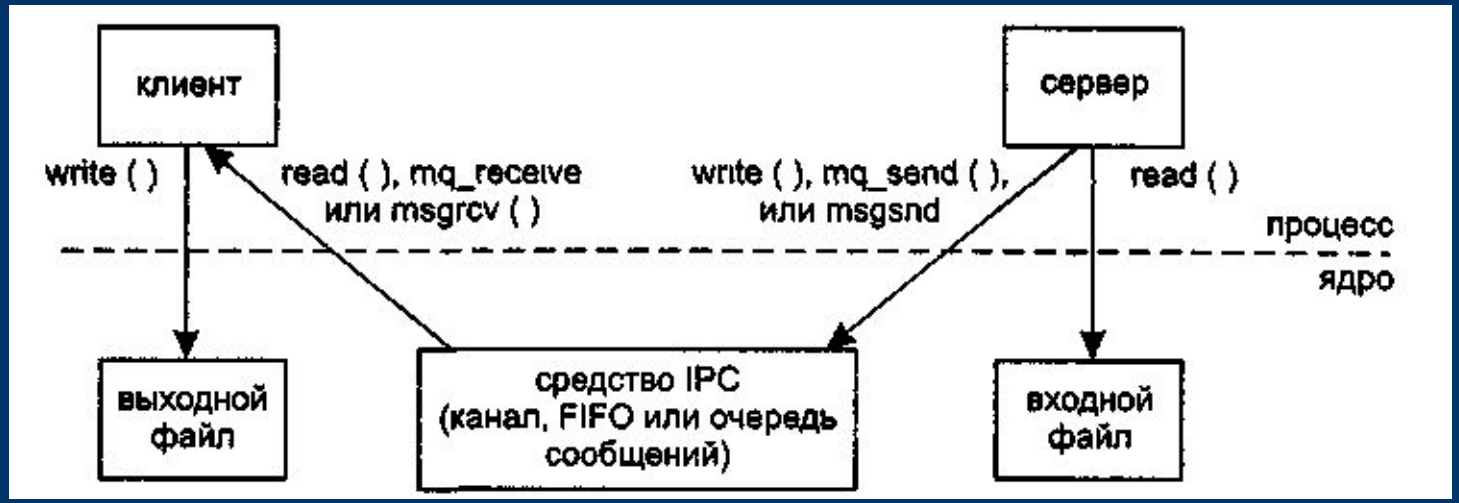
- Разделяемая память является наиболее быстрым средством межпроцессного взаимодействия. После отображения области памяти в адресное пространство процессов, совместно ее использующих, для передачи данных между процессами больше не требуется участие ядра. Обычно, однако, требуется некоторая форма синхронизации процессов, помещающих данные в разделяемую память и считывающих ее оттуда.
- Рассмотрим по шагам работу программы копирования файла типа клиент-сервер, которую мы использовали в качестве примера для иллюстрации различных способов передачи сообщений.

Введение

- Сервер считывает данные из входного файла. Данные из файла считываются ядром в свою память, а затем копируются из ядра в память процесса.
- Сервер составляет сообщение из этих данных и отправляет его, используя именованный или неименованный канал или очередь сообщений. Эти формы IPC обычно требуют копирования данных из процесса в ядро.
- Клиент считывает данные из канала IPC, что обычно требует их копирования из ядра в пространство процесса.
- Наконец, данные копируются из буфера клиента (второй аргумент вызова write) в выходной файл.

Введение

- Видно, что для копирования файла обычно требуются четыре операции копирования данных. Эти операции копирования осуществляются между процессами и ядром и являются дорогостоящими операциями (более дорогостоящей, чем копирование данных внутри ядра или внутри одного процесса). На рис. 1 изображено перемещение данных между клиентом и сервером через ядро.



Введение

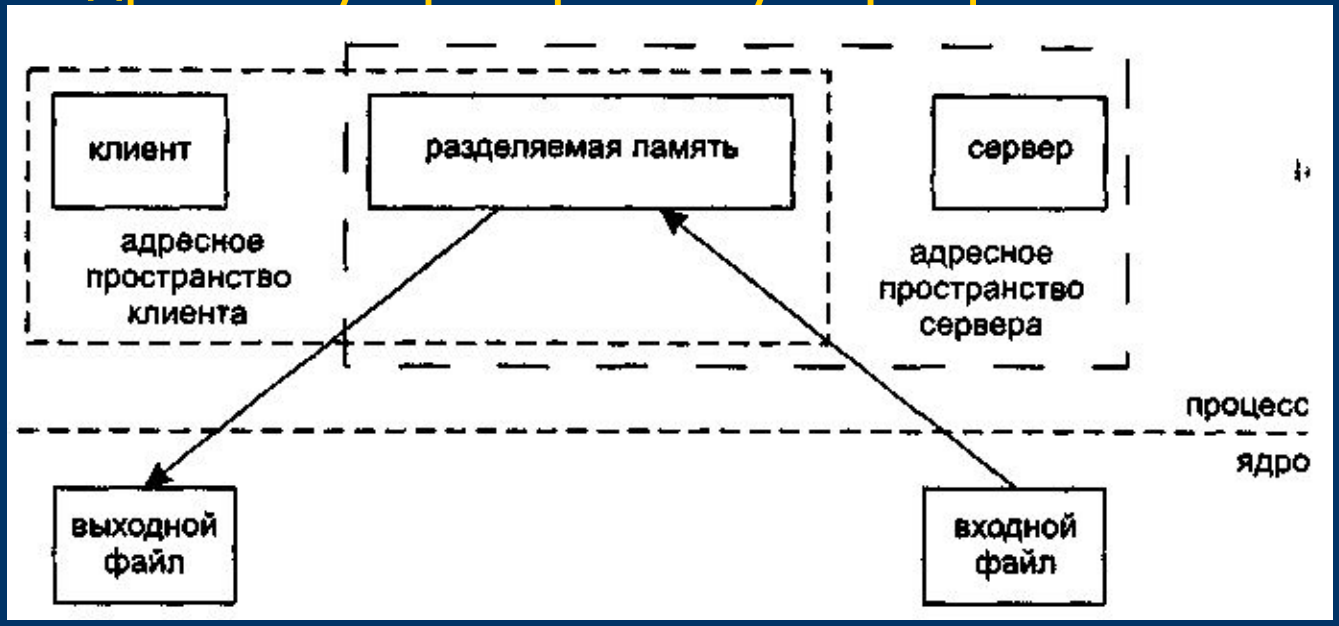
- Недостатком этих форм IPC — именованных и неименованных каналов — является то, что для передачи между процессами информация должна пройти через ядро. Разделяемая память дает возможность обойти этот недостаток, поскольку ее использование позволяет двум процессам обмениваться данными через общий участок памяти. Процессы, разумеется, должны синхронизировать и координировать свои действия. Одновременное использование участка памяти во многом аналогично совместному доступу к файлу, например к файлу с последовательным номером.

Введение

- Теперь информация передается между клиентом и сервером в такой последовательности:
- сервер получает доступ к объекту разделяемой памяти, используя для синхронизации семафор (например);
- сервер считывает данные из файла в разделяемую память. Второй аргумент вызова read (адрес буфера) указывает на объект разделяемой памяти;
- после завершения операции считывания клиент уведомляется сервером с помощью семафора;
- клиент записывает данные из объекта разделяемой памяти в выходной файл.
- Этот сценарий иллюстрирует рис. 2.

Введение

- Из рисунка видно, что копирование данных происходит лишь дважды: из входного файла в разделяемую память и из разделяемой памяти в выходной файл. Два прямоугольника штриховыми линиями подчеркивают, что разделяемая память принадлежит как адресному пространству клиента, так и адресному пространству сервера.



Введение

- Основные принципы разделяемой памяти System V совпадают с концепцией разделяемой памяти Posix. Вместо вызовов shm_open и imap в этой системе используются вызовы shmget и shmat.
- Для каждого сегмента разделяемой памяти ядро хранит нижеследующую структуру **`shimd_ds`**, определенную в заголовочном файле <sys/shm.h>:
- Структура ipc_perm содержит разрешения доступа к сегменту разделяемой памяти.

Введение

```
struct shimd_ds {
struct ipc_perm shm_perm; /* структура разрешений */
size_t shm_segsz; /* размер сегмента */
pid_t    shm_lpid;    /*    идентификатор    процесса,
  выполнившего последнюю операцию*/
pid_t shm_cpid; /*идентификатор процесса-создателя*/
shmatt_t     shm_nattch;     /*текущее     количество
  подключений*/
shmat_t   shm_cnattch;   /*   количество   подключений
  in-core */
time_t shm_atime; /* время последнего подключения */
time_t shm_dtime; /* время последнего отключения */
time_t   shm_ctime;   /*время   последнего   изменения
  данной структуры */
};
```

Функция shmget

- С помощью функции shmget можно создать новый сегмент разделяемой памяти или подключиться к существующему:
- `#include <sys/shm.h>`
- `int shmget(key_t key, size_t size, int oflag);`
- /* Возвращает идентификатор разделяемой памяти в случае успешного завершения, -1 в случае ошибки */
- Возвращаемое целочисленное значение называется идентификатором разделяемой памяти. Он используется с тремя другими функциями shmXXX.
- Аргумент *key* может содержать значение, возвращаемое функцией ftok, или константу IPC_PRIVATE.

Функция shmget

- Аргумент *size* указывает размер сегмента в байтах. При создании нового сегмента разделяемой памяти нужно указать ненулевой размер. Если производится обращение к существующему сегменту, аргумент *size* должен быть нулевым.
- Флаг *oflag* представляет собой комбинацию флагов доступа на чтение и запись. К ним могут быть добавлены с помощью логического сложения флаги IPC_CREAT или IPC_CREAT | IPC_EXCL. Новый сегмент инициализируется нулями.
- Функция shmget создает или открывает сегмент разделяемой памяти, но не дает вызвавшему процессу доступа к нему. Для подключения сегмента памяти предназначена функция shmat, описанная далее.

Функция shmat

- После создания или открытия сегмента разделяемой памяти вызовом shmget его нужно подключить к адресному пространству процесса вызовом shmat:
- `#include <sys/shm.h>`
- `void *shmat(int shmid, const void *shmaddr, int flag);`
- `/* Возвращает начальный адрес полученной области в случае успешного завершения. -1 -в случае ошибки */`
- Аргумент *shmid* — это идентификатор разделяемой памяти, возвращенный shmget. Функция shmat возвращает адрес начала области разделяемой памяти в адресном пространстве вызвавшего процесса.

Функция shmat

- Правила, по которым формируется этот адрес, таковы:
  - если аргумент *shmaddr* представляет собой нулевой указатель, система сама выбирает начальный адрес для вызвавшего процесса. Это рекомендуемый (и обеспечивающий наилучшую совместимость) метод;
  - если *shmaddr* отличен от нуля, возвращаемый адрес зависит от того, был ли указан флаг SHM_RND (в аргументе *flag*);
  - если флаг SHM_RND не указан, разделяемая память подключается непосредственно с адреса, указанного аргументом *shmaddr* ;
  - если флаг SHM_RND указан, сегмент разделяемой памяти подключается с адреса, указанного аргументом *shmaddr,* округленного вниз до кратного константе SHMLBA. Аббревиатура LBA означает lower boundary address — нижний граничный адрес.

Функция shmat

- По умолчанию сегмент подключается для чтения и записи, если процесс обадает соответствующими разрешениями. В аргументе *flag* можно указать константу SHM_RDONLY, которая позволит установить доступ только на чтение.

Функция shmdt

- После завершения работы с сегментом разделяемой памяти его следует отключить вызовом shmdt:
- `#include <sys/shm.h>`
- `int shmdt(const void *shmaddr);`
- `/* Возвращает 0 в случае успешного завершения, -1 - в случае ошибки */`
- При завершении работы процесса все сегменты, которые не были отключены им явно, отключаются автоматически.
- Обратите внимание, что эта функция не удаляет сегмент разделяемой памяти. Удаление осуществляется функцией shmctl с командой IPC_RMID.

Функция shmctl

- Функция shmctl позволяет выполнять различные операции с сегментом разделяемой памяти:
- `#include <sys/shm.h>`
- `int shmctl(int shmid, int cmd, struct shmid_ds *buff);`
- `/* Возвращает 0 в случае успешного завершения, -1 в случае ошибки */`

Функция shmctl

- Команд (значений аргумента *cmd)* может быть три:

1. IPC_RMID — удаление сегмента разделяемой памяти с идентификатором *shmid* из системы;
2. IPC_SET — установка значений полей структуры shmid_ds для сегмента разделяемой памяти равными значениям соответствующих полей структуры, на которую указывает аргумент *buff:* shm_perm.uid, shm_perm.gid, shm_perm.mode. Значение поля shm_ctime устанавливается равным текущему системному времени;
3. IPC_STAT — возвращает вызывающему процессу (через аргумент *buff)* текущее значение структуры shmid_ds для указанного сегмента разделяемой памяти.

Простые программы

- Приведём несколько примеров простых программ, иллюстрирующих работу с разделяемой памятью System V.
- Программа shmget, текст которой приведен в листинге 1, создает сегмент разделяемой памяти, принимая из командной строки полное имя и длину сегмента.

Простые программы

- Вызов shmget создает сегмент разделяемой памяти указанного размера. Полное имя, передаваемое в качестве аргумента командной строки, преобразуется в ключ IPC System V вызовом ftok. Если указан параметр -e, наличие существующего сегмента с тем же именем приведет к возвращению ошибки. Если мы знаем, что сегмент уже существует, в командной строке должна быть указана нулевая длина.
- Вызов shmat подключает сегмент к адресному пространству процесса. После этого программа завершает работу. Разделяемая память System V обладает по меньшей мере живучестью ядра, поэтому сегмент разделяемой памяти при этом не удаляется.

Простые программы

- В листинге 2 приведен текст тривиальной программы shmrmid, которая вызывает shmctl с командой IPC_RMID для удаления сегмента разделяемой памяти из системы.
- В листинге 3 приведен текст программы shmwrite, которая заполняет сегмент разделяемой памяти последовательностью значений 0,1,2,…,254,255,0,1…
- 10-12 Сегмент разделяемой памяти открывается вызовом shmget и подключается вызовом shmat. Его размер может быть получен вызовом shmctl с командой IPC_STAT.
- 13-15 В разделяемую память записывается последовательность значений.

Простые программы

- Программа shmread, текст которой приведен в листинге 4, проверяет последовательность значений, записанную в разделяемую память программой shmwrite.
- 10-12 Открываем и подключаем сегмент разделяемой памяти. Его размер может быть получен вызовом shmctl с командой IPC_STAT.
- 13-15 Проверяется последовательность, записанная программой shmwrite.

Простые программы

- Создадим сегмент разделяемой памяти длиной 1234 байта в системе ASPlinux 9.0. Для идентификации сегмента используем полное имя нашего исполняемого файла shmget. Это имя будет передано функции ftok. Имя исполняемого файла сервера часто используется в качестве уникального идентификатора для данного приложения:
- `[root@gun_linux_vm]# ./shmget shmget 1234`

Простые программы

- `[root@gun_linux_vm]# ipcs -m`
- Программу ipcs мы запускаем для того, чтобы убедиться, что сегмент разделяемой памяти действительно был создан и не был удален по завершении программы shmcreate. Количество подключений (хранящееся в поле shm_nattch структуры shmid_ds) равно нулю, как мы и предполагали.

Простые программы

- Теперь запустим программу shmwrite, чтобы заполнить содержимое разделяемой памяти последовательностью значений. Затем проверим содержимое сегмента разделяемой памяти программой shmread и удалим этот сегмент:
- `[root@gun_linux_vm]# ./shmwrite shmget`
- `[root@gun_linux_vm]# ./shmread shmget`
- `[root@gun_linux_vm]# ./shmrmid shmget`
- `[root@gun_linux_vm]# ipcs -m`
- Мы используем программу ipcs, чтобы убедиться, что сегмент разделяемой памяти действительно был удален.

Ограничения, накладываемые на разделяемую память

- На разделяемую память System V накладываются определенные ограничения точно так же, как и на семафоры и очереди сообщений System V. В табл. 1 приведены значения этих ограничений для разных реализаций. В первом столбце приведены традиционные для System V имена переменных ядра, в которых хранятся эти ограничения.

Ограничения, накладываемые на разделяемую память

Имя	Описание	DUnix 4.0	Solaris 2.6
shmmax	Максимальный размер сегмента в байтах	4 194 304	1 048 576
shmmnb	Минимальный размер сегмента разделяемой памяти в байтах	1	1
shmmni	Максимальное количество идентификаторов разделяемой памяти в системе	128	100
shmseg	Максимальное количество сегментов, подключенных к процессу	32	6

Ограничения, накладываемые на разделяемую память

- Программа в листинге 5 определяет значения четырех ограничений, приведенных в табл.1.
- Запустив эту программу, увидим:
- `[root@gun_linux_vm]# ./limits`
- `4096 identifiers open at once`
- `4096 shared memory segments attached at once`
- `minimum size of shared memory segment=1`
- `max size of shared memory segment=33554432`
- Проверить ограничения на разделяемую память в Linux можно командой `sysctl -a | grep kernel.shm*`.