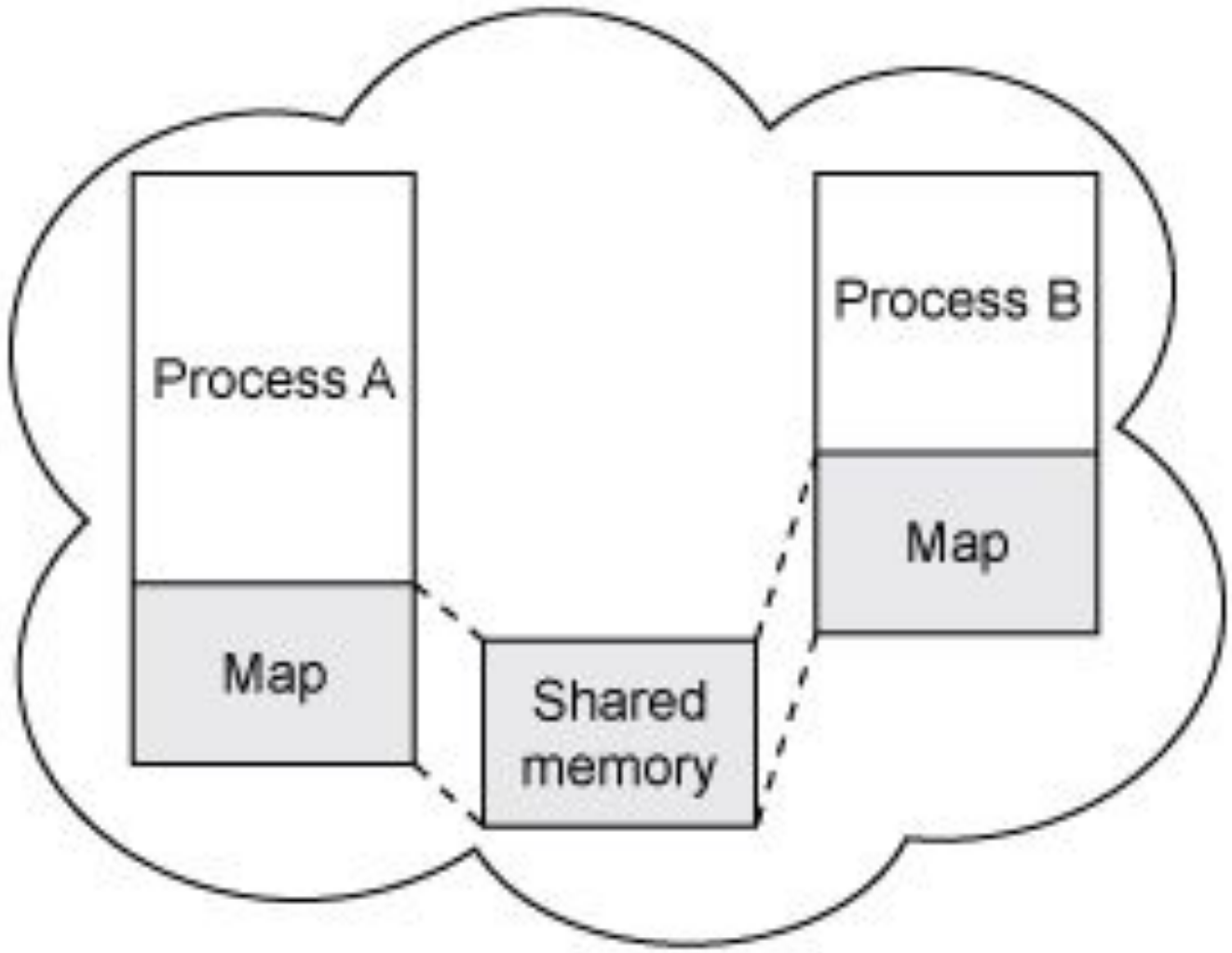
Архитектура ЭВМ. Операционные системы

Власов Евгений

Разделяемая память (shared memory)

применяют для того, чтобы увеличить скорость прохождения данных между процессами. В обычной ситуации обмен информацией между процессами проходит через ЯДРО. Техника разделяемой памяти позволяет осуществить обмен информацией не через ядро, а используя некоторую часть виртуального адресного пространства, куда помещаются и откуда считываются данные.

После создания разделяемого сегмента памяти любой из пользовательских процессов может подсоединить его к своему собственному виртуальному пространству и работать с ним, как с обычным сегментом памяти. Недостатком такого обмена информацией является отсутствие каких бы то ни было средств синхронизации, однако для преодоления этого недостатка можно (но необязательно) применять средства сихронизации обмена данными.


Process A
Map
Shared memory
Process B
Map

Системные вызовы работы с разделяемой памятью

- `shmget` — создание сегмента разделяемой памяти;
- `shmctl` — установка параметров;
- `shmat` — подсоединение сегмента памяти;
- `shmdt` — отсоединение сегмента

Получение IPC идентификатора

```
#include <sys/ipc.h>
key_t     ftok(const char *path, int id);
```

использует файл с именем *pathname* (которое должно указывать на существующий файл к которому есть доступ) и младшие 8 бит *id* (который должен быть отличен от нуля) для создания ключа с типом **key_t.**

Возвращаемое значение одинаково для всех имен, указывающих на один и тот же файл при одинаковом значении *id*. Возвращаемое значение должно отличаться, когда (одновременно существующие) файлы или идентификаторы проекта различаются.

Создание разделяемой памяти

```
#include <sys/ipc.h>
#include <sys/shm.h>
int shmget(key_t key, size_t size, int shmflg);
```

возвращает идентификатор разделяемого сегмента памяти, соответствующий значению аргумента `key`. Создается новый разделяемый сегмент памяти с размером `size` округленным до размера, кратного `PAGE_SIZE`), если значение `key` равно `IPC_PRIVATE` или если значение `key` не равно `IPC_PRIVATE` и нет идентификатора, соответствующего `key` и выражение `shmflg&IPC_CREAT` истинно .

Значение `shmflg`

играет роль только при создании нового сегмента разделяемой памяти и определяет права различных пользователей при доступе к сегменту, а также необходимость создания нового сегмента и поведение системного вызова при попытке создания

- **IPC_CREAT** служит для создания нового сегмента. Если этого флага нет, то функция **`shmget()`** будет искать сегмент, соответствующий ключу *key* и затем проверит, имеет ли пользователь права на доступ к сегменту.
- **IPC_EXCL** используется совместно с **IPC_CREAT** для того, чтобы не создавать существующий сегмент заново.

Если создается новый сегмент, то права доступа копируются из *shmflg* в *shm_perm*, являющийся членом структуры *shmid_ds*, которая определяет сегмент.

```
struct shmid_ds {
    struct ipc_perm shm_perm;    /* права операции */
    int             shm_segsz;   /* размер сегмента (в байтах) */
    time_t          shm_atime;   /* время последнего подключения */
    time_t          shm_dtime;   /* время последнего отключения */
    time_t          shm_ctime;   /* время последнего изменения */
    unsigned short  shm_cpid;    /* идентификатор процесса создателя */
    unsigned short  shm_lpid;    /* идентификатор последнего пользователя */
    short           shm_nattch;  /* количество подключений */
};
```

```
struct ipc_perm {
    key_t  key;
    ushort uid;   /* действующие идентификаторы владель-ца и группы euid и
egid */
    ushort gid;
    ushort cuid;  /* действующие идентификаторы создате-ля euid и egid */
    ushort cgid;
    ushort mode;  /* младшие 9 битов shmflg */
    ushort seq;   /* номер последовательности */
};
```

Ошибки при создании разделяемой памяти

Ошибка	Описание
EINVAL	**если создается новый сегмент, а *size* < SHMMIN или *size* > SHMMAX, либо новый сегмент не был создан. Сегмент с данным ключем существует, но *size* больше чем размер этого сегмента.**
EEXIST	**если значение IPC_CREAT | IPC_EXCL было указано, а сегмент уже существует.**
ENOSPC	**если все возможные идентификаторы сегментов уже распределены (SHMMNI) или если размер выделяемого сегмента превысит системные лимиты (SHMALL).**
ENOENT	**если не существует сегмента для ключа *key*, а значение IPC_CREAT не указано.**
EACCES	**EACCES** если у пользователя нет прав доступа к сегменту разделяемой памяти.
ENOMEM	если в памяти нет свободного для сегмента пространства

Присоединение к разделяемой памяти

```
#include <sys/types.h>
#include <sys/shm.h>
void *shmat(int shmid, const void *shmaddr, int
shmflg);
```

Функция **shmat** подстыковывает сегмент разделяемой памяти **shmid** к адресному пространству вызывающего процесса.

При завершении работы процесса сегмент будет отстыкован. Один и тот же сегмент может быть подстыкован в адресное пространство процесса несколько раз, как "только для чтения", так и в режиме "чтение-запись".

Адрес подстыковываемого сегмента определяется *shmaddr* с помощью одного из критериев:

- Если *shmaddr* равен **NULL**, то система выбирает для подстыкованного сегмента подходящий (неиспользованный) адрес
- Если *shmaddr* не равен **NULL**, а в поле *shmflg* включен флаг **SHM_RND**, то подстыковка производится по адресу *shmaddr*, округленному вниз до ближайшего кратного **SHMLBA**. В противном случае *shmaddr* должен быть округленным до размера страницы адресом, к которому производится подстыковка.
- Если в поле *shmflg* включен флаг **SHM_RDONLY**, то подстыковываемый сегмент будет доступен только для чтения, и вызывающий процесс должен иметь права на чтение этого сегмента. Иначе, сегмент будет доступен для чтения и записи, и у процесса должны быть соответствующие права. Сегментов "только-запись" не существует.
- *Флаг **SHM_REMAP** (специфичный для Linux) может быть указан в shmflg для обозначения того, что распределение сегмента должно замещать любые существующие распределения в диапазоне, начиная с shmaddr и до размера сегмена. (Обычно выдается ошибка **EINVAL** если уже существует распределение в этом диапазоне адресов.) В этом случае shmaddr не должно быть равно **NULL**.*

Отсоединение от процесса разделяемого участка памяти

```
#include <sys/types.h>
#include <sys/shm.h>
int shmdt(const void *shmaddr);
```

отстыковывает сегмент разделяемой памяти, находящийся по адресу *shmaddr*, от адресного пространства вызвающего процесса. Отстыковываемый сегмент должен быть среди пристыкованных ранее функцией **shmat**. Параметр *shmaddr* должен быть равен значению, которое возвратила соответствующая функция **shmat**.

Управление разделяемым участком памяти

```
#include <sys/ipc.h>
#include <sys/shm.h>
int shmctl(int shmid, int cmd, struct shmid_ds
*buf);
```

получать информацию о разделяемых сегментах памяти, устанавливать владельца, группу разделяемого сегмента, права на него; эта функция может также удалить сегмент. Информация о сегменте, которая находится в *shmid*, возвращается в структуру *shmid_ds*.

Пользователь *должен* удостовериться, что сегмент удален; иначе страницы, которые не были удалены, останутся в памяти или в разделе подкачки.