Спільна пам'ять в ОС UNIX

1. Загальні відомості

1.1. Пам'ять що розділяється – засіб міжпроцесного зв'язку, що дозволяє процесам мати загальні області віртуальної пам'яті і, як наслідок, розділяти інформацію, що міститься в них. Одиницею пам'яті, що розділяється є сегменти, властивості яких залежать від апаратних особливостей керування пам'яттю.

Поділ пам'яті забезпечує найбільш швидкий обмін даними між процесами.

1.2. Робота з пам'яттю, що розділяється починається з того, що процес за допомогою системного виклику `shmget(2)` створює поділюваний сегмент з унікальним ідентифікатором і асоційовану з ним структуру даних. Унікальний ідентифікатор називається ідентифікатором пам‘яті що розділяється (`shmid`); він використовується для звертань до асоційованої структури даних, що визначається в такий спосіб:

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


struct shmid_ds
{
   // Структура прав на виконання операцій
   struct ipc_perm shm_perm;
   // Розмір сегмента
   int shm_segsz;
   //   Покажчик   на   структуру   області
пам'яті
   struct region* shm_reg;
   // Інформація для підкачки
```

```
        char pad[4]
};
```

Існує два варіанти його використання для створення нової області поділюваної пам'яті.

1.3. Стандартний спосіб. Ключу для системного виклику вказується значення сформоване функцією `ftok()` для деякого імені файлу і номера екземпляра області пам'яті, що розділяється. У якості прапорців вказується комбінація прав доступу до створюваного сегмента і прапора `IPC_CREAT`. Якщо сегмент для даного ключа ще не існує, то система буде намагатися створити його з зазначеними правами доступу. Якщо ж він вже існував, то ми просто одержимо його дескриптор. Можливе додавання до цієї комбінації прапорів прапора`IPC_EXCL`. Цей прапор гарантує нормальне завершення системного виклику тільки в тому випадку, якщо сегмент дійсно був створений (тобто раніше він не існував), якщо ж сегмент існував, системний виклик завершиться з помилкою, і значення системної змінної `errno`, описаної у файлі `errno.h`, буде встановлене `EEXIST`.

1.4. Нестандартний спосіб. Значенням ключа вказується спеціальне значення `IPC_PRIVATE`. Використання значення `IPC_PRIVATE` завжди призводить до спроби створення нового сегмента пам'яті, що розділяється з заданими правами доступу і з ключем, що не збігається із значенням ключа жодного з вже існуючих сегментів і який не може бути отриманий за допомогою функції `ftok()`. Наявність прапорів `IPC_CREAT` і `IPC_EXCL` у цьому випадку ігнорується.

1.5. Щоб потім одержати доступ до поділюваного сегмента, його потрібно приєднати за допомогою системного виклику `shmat()`, що розмістить сегмент у віртуальному просторі процесу. Після приєднання, відповідно до прав доступу, процеси можуть читати дані із сегмента і записувати їх.

1.6. Коли поділюваний сегмент стає непотрібним, його треба від‘єднати, скориставшись системним викликом `shmdt()`.

1.7. Для виконання керуючих дій над пам’яттю, що розділяється служить системний виклик `shmctl(2)`. У число керуючих дій входить розпорядження утримувати сегмент в оперативній пам'яті і зворотне розпорядження про зняття утримання. Після того, як останній процес від‘єднав поділюваний сегмент, потрібно виконати керуючу дію по видаленню сегмента із системи.

2. Синтаксис та призначення системних викликів

2.1. Системний виклик `shmget`.

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

int shmget (key, size, shmflg)
            key_t key;
            int size;
            int shmflg;
```

Системний виклик `shmget` призначений для виконання операції доступу до сегмента пам'яті, що розділяється і, у випадку його успішного завершення, повертає дескриптор System V IPC для цього сегмента.

Параметр `key` є ключем System V IPC для сегмента, тобто фактично його ім'ям із простору імен System V IPC. Значенням цього параметра може бути значення ключа, отримане за допомогою функції `ftok()`, чи спеціальне значення `IPC_PRIVATE`. Використання значення `IPC_PRIVATE` завжди приводить до спроби створення нового сегмента пам'яті, що розділяється з ключем, що не збігається із значенням ключа жодного з вже існуючих сегментів і який не може бути отриманий за допомогою функції `ftok()` при жодній комбінації її параметрів.

Параметр `size` визначає розмір створюваного чи вже існуючого сегмента в байтах. Якщо сегмент із зазначеним ключем вже існує, але його розмір не збігається з зазначеним у параметрі `size`, констатується виникнення помилки.

Параметр `shmflg` – прапорці (відіграє роль тільки при створенні нового сегмента пам'яті, що розділяється) визначає права різних користувачів при доступі до сегмента, а також необхідність створення нового сегмента. Він є деякою комбінацією (за допомогою операції побітове чи - "|") прав доступу (див. Лабораторну роботу №3).

При успішному завершенні системного виклику повертається ідентифікатор сегмента пам'яті, що розділяється. У випадку помилки повертається -1, а змінній `errno` присвоюється код помилки.

(Примітка: Необхідно явно видаляти поділюваний сегмент пам'яті після того, як видаляється останнє посилання на нього.)

2.2. Системний виклик `shmat`.

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

char* shmat (shmid, shmaddr, shmflg)
             int shmid;
             char* shmaddr;
             int shmflg;
```

Системний виклик `shmat` приєднує сегмент пам'яті, що розділяється, асоційований з ідентифікатором `shmid`, до сегмента даних процесу. Якщо значення аргументу `shmaddr` дорівнює нулю, то сегмент приєднується за адресою, обраною системою.

Аргумент `shmflg` використовується для передачі системному виклику `shmat()` прапорців `SHM_RND` і `SHM_RDONLY`. Наявність першого з них означає, що адресу `shmaddr` варто заокруглити до деякої системно-залежної величини. Другий прапорець наказує приєднати сегмент тільки для читання; якщо він не встановлений, приєднаний сегмент буде доступний і на читання, і на запис (якщо процес має відповідні права).

(Примітка: У деяких реалізаціях результат системного виклику shmat має тип int, а не char *)

При успішному завершенні системного виклику `shmat` повертається початкова адреса приєднаного сегмента. У випадку помилки повертається -1, а змінній `errno` присвоюється код помилки.

2.3. Системний виклик `shmdt`.

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

int shmdt (shmaddr)
            char* shmaddr;
```

Системний виклик `shmdt` від'єднує поділюваний сегмент пам'яті, розташований за адресою `shmaddr`, від сегмента даних процесу.

При успішному завершенні системного виклику `shmdt` результат дорівнює 0. У випадку помилки повертається -1, а змінній `errno` присвоюється код помилки.

2.4. Системний виклик `shmctl`.

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

int shmctl (shmid, cmd, buf)
            int shmid;
            int cmd;
            struct shmid_ds* buf;
```

Системний виклик shmctl дозволяє виконувати операції керування сегментами пам'яті, що розділяється. Як аргумент shmid повинен виступати ідентифікатор сегмента пам'яті, що розділяється, попередньо отриманий за допомогою системного виклику shmget(2).

Операція визначається значенням аргументу cmd, що повинне бути одним з наступних:

IPC_STAT - Помістити поточні значення полів структури даних, асоційованої з ідентифікатором сегмента пам‘яті shmid, у структуру, на яку вказує аргумент buf.

IPC_SET - У структурі даних, асоційованій з ідентифікатором shmid, перевстановити значення діючих ідентифікаторів користувача (shm_perm.uid) і групи (shm_perm.gid), прав на операції (shm_perm.mode). Потрібні значення витягаються зі структури даних, на яку вказує аргумент buf.

IPC_RMI - Видалити із системи ідентифікатор shmid, ліквідувати сегмент пам'яті, що розділяється і асоційовану з ним структуру даних.

SHM_LOCK - Утримати в пам'яті поділюваний сегмент, заданий ідентифікатором shmid.

SHM_UNLOCK - Звільнити сегмент пам'яті, що розділяється, заданий ідентифікатором shmid.

Щоб виконати керуючі дії IPC_SET чи IPC_RMI, процес повинен мати діючий ідентифікатор користувача, рівний ідентифікаторам творця чи власника сегмента пам‘яті, або ідентифікатору суперкористувача. Керуючі дії SHM_LOCK і SHM_UNLOCK може виконати тільки суперкористувач. Для виконання керуючого дії IPC_STAT процесу потрібно право на читання.

При успішному завершенні результат дорівнює 0; у випадку помилки повертається -1, а змінній errno присвоюється код помилки.

2.5. Пам'ять, що розділяється і системні виклики fork(), exec() і функція exit().

Важливим питанням є поведінка сегментів пам'яті, що розділяється при виконанні процесом системних викликів fork(), exec() і функції exit().

При виконанні системного виклику fork() всі області пам'яті, що розділяється розміщені в адресному просторі процесу, успадковуються породженим процесом.

При виконанні системних викликів exec() і функції exit() всі області поділюваної пам'яті, розміщені в адресному просторі процесу, виключаються з його адресного простору, але продовжують існувати в операційній системі.