ТЕМА 6. ВЗАИМОДЕЙСТВИЕ НА ПРОЦЕСИ ПОСРЕДСТВОМ ПОДЕЛЕНА ПАМЕТ

Поделена памет в UNIX. Системни извиквания shmget(), shmat(), shmdt(). Команди ipc и ipcrm. Използване на системното извикване shmctl() за освобождаване на ресурси заети от поделена памет. Поделена памет и системни извиквания fork(), exec() и функция exit().

1. Поделена памет в UNIX

Поделената памет (shared memory) е второто средство от групата на System V IPC за взаимодействие между процеси (първото, опашки от съобщения бе разгледано в предходното упражнение). Тези средства първо се създадат, след което е възможно организиране на междупроцесното взаимодействие. За създаване на област от поделена памет, както и за достъп до вече създадена такава се използва системното извикване *shmget()*. Съществуват два варианта за неговото използване:

•Първи вариант (стандартен способ) – Системното извикване създава поделената памет, като използва ключа формиран от функцията *ftok()*. От своя страна *ftok()* използва като параметри името на съществуващ файл и номера на инстанцията на създаваната област. При този вариант параметъра *shmflg* се задава като комбинация от правата за достъп до създавания сегмент и флага *IPC_CREAT*. Последното означава, че ако сегментът, определен от ключа все още не съществува, то системата ще се опита да го създаде с указаните права за достъп. В противен случай, ако сегмента съществува, то извикването ще върне неговия дескриптор. Възможно е правата за достъп да се комбинират с флага *IPC_EXCL*, който гарантира нормално завършване на системното извикване в случай, че сегментът не съществува. В противен случай (ако сегментът вече съществува), то системното извикване завършва с грешка и стойността на системната променлива *errno*, описана във файла *errno.h*, се установява в *EEXIST*.

•Втори вариант (нестандартен способ) – За ключ се използва константата *IPC_PRIVATE*. Така се създава нов сегмент от поделена памет, чийто ключ не съвпада с нито един от ключовете на вече съществуващите

сегменти и който не може да бъде получен с помощта на функцията *ftok()* при нито една комбинация от нейните параметри. Наличието на флагове *IPC_CREAT* и *IPC_EXCL* в този случай се игнорира.

ИМЕ

shmget

ПРОТОТИП

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

ОПИСАНИЕ

Системното извикване shmget е предназначено за получаване на достъп до сегмент от поделена памет. При успешно приключване, то връща System V IPC дескриптора за този сегмент (цяло неотрицателно число, еднозначно идентифициращо сегмента в цялата изчислителна система).
Параметърът key е System V IPC ключа за сегмента, т. е. неговото име от пространството на System V IPC имената. Неговата стойност се получава или с помоща на функцията ftok() или се задава специалната константа IPC_PRIVATE. Използването на IPC_PRIVATE води до създаване на нов сегмент поделена памет с ключ, който не съвпада с ключа на нито един от вече съществуващите сегменти и който не може да бъде получен с помоща на функцията ftok() при нито една комбинация на нейните параметри.
Параметърът size определя размера на сегмента в байтове. Ако указания сегмент вече съществува и неговия размер не съвпада с указания в параметъра size, възниква грешка.
Параметъра shmflg се използва само при създаване на нов сегмент поделена памет. Той определя правата на различните потребители за достъп до сегмента. Освен това, чрез него се определя дали сегмента съществува и съответно дали има нужда от създаване на нов. Стойността на този параметър се задава като комбинация (с помощта на побитово или – "|") от следните предварително дефинирани константи и осмичните права за достъп:
IPC_CREAT – ако сегмента за указания ключ не съществува, трябва да бъде създаден;
IPC_EXCL – използва се съвместно с флага IPC_CREAT. При съвместното им използване и наличието на сегмент с указания ключ, възниква грешка, при което променливата errno, описана във файла <errno.h>, приема стойност EEXIST;
0400 — право за четене за потребителя, създал сегмента;
0200 — право за запис за потребителя, създал сегмента;
0040 — право за четене за групата на потребителя, създал сегмента;
0020 — право за запис за групата на потребителя, създал сегмента;
0004 — право за четене за всички останали потребители;
0002 — право за запис за всички останали потребители.

ВРЪЩАНИ СТОЙНОСТИ

Системното извикване връща стойнстта на System V IPC дескриптора за сегмента от поделена памет, при нормално завършване и стойност -1 при възникване на грешка.

Достъпът до създадена област от поделена памет се обезпечава от нейния дескриптор. Получаването му може да се извърши по два начина:

• При първия се използва системното извикване *shmget()*, което връща дескриптора на поделената памет, при известен ключ. В този случай не трябва да се използва флага *IPC_EXCL*, а стойността на ключа не може да бъде *IPC_PRIVATE*. Правата за достъп се игнорират, а размера на

областта трябва да съвпада с размера указан при нейното създаване.

•При втория начин се използва факта, че System V IPC дескриптора е действителен в рамките на цялата операционна система, което позволява неговата стойност да се предава от процеса създал поделената памет, към текущия процес. Трябва да се отбележи, че при създаване на поделена памет с помощта на константата *IPC_PRIVATE* – това е единствено възможния способ.

След като дескриптора е получен е необходимо включването на областта от поделена памет в адресното пространство на текущия процес. Това се осъществява с помощта на системното извикване *shmat()*. При нормално завършване, то връща адреса на поделената памет в адресното пространство на текущия процес. Последващия достъп до тази памет се осъществява с помощта на обичайните средства на езика за програмиране.

ИМЕ

shmat

ПРОТОТИП

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

ОПИСАНИЕ

Системното извикване shmat е предназначено за включване на областа от поделана памет в адреснто пространство на текущия процес. Даденото описание не е пълно и е адаптирано за настоящия курс. Повече информация може да се намери в UNIX Manual.
Параметърът shmid е System V IPC дескриптора за сегмената от поделена памет, т. е. стойността, която връща системното извикване shmget() при създаването на сегмент или при достъп до него по зададен ключ.
В настоящия курс на параметъра shmaddr се задава стойност NULL, което позволява на операционната система, само да помести поделената памет в адресното пространство на процеса.
Параметърът shmflg в настоящия курс приема само две стойности: 0 – за операциите четене и запис и SHM_RDONLY – само за четене от сегмента. При това процеса трябва да притежава съответните права за достъп до сегмента.

ВРЪЩАНИ СТОЙНОСТИ

При нормално приключване на системното извикване се връща адреса на сегмента от поделена памет в адресното пространство на процеса и -1 при възникване на грешка.

След приключване на използването на поделената памет процеса може да намали размера на своето адресно пространство, изключвайки я от него с помощта на системното извикване *shmdt()*. Като параметър на системното извикване *shmdt()* трябва да се зададе началния адрес на областта от поделената памет в адресното пространство на процеса, т. е.

стойността, която връща системното извикване *shmat()*. Ето защо тази стойност трябва да се съхранява през цялото време на нейното използване.

ИМЕ

shmdt

ПРОТОТИП

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

ОПИСАНИЕ

Системното извикване shmdt е предназначено за изключване на областа от поделана памет от адресното пространство на текущия процес.
Параметърът shmaddr е адреса на сегмента от поделена памет, т. е. стойността, която връща системното извикване shmat().

ВРЪЩАНИ СТОЙНОСТИ

Системното извикване връща стойност 0 при нормално приключване и стойност -1 при възникване на грешка.

Следващите две взаимодействащи си програми илюстрират използването на поделена памет.

```
/* Програма 1 (06_1a.c), илюстрираща работата с поделена памет */
/* В програмата се създава поделена памет, която представлява
масив от три цели числа. В първия елемент на масива се записват
броя стартирания на програма 1, във вория елемент – броя стартирания
на програма 2, а в третия – общия брой и за двтете програми */
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <errno.h>
int main()
{
    int *array;    /* Указател към поделената памет */
    int shmid;     /* IPC дескриптор за областта от поделена памет */
    int new = 1;   /* Флаг, показващ необходимостта от инициализация
                      на елементите от масива */
    char pathname[] = "06_1a.c"; /* Името на файла,
        използван за генериране на ключа. Файл с това
        име трябва да съществува в текущата директория */
    key_t key;     /* IPC ключ */
    /* Генериране на IPC ключа (използва се името на файла 06_1a.c в
            текущата директория и номера на инстанцията на областта от
        поделена памет 0 */
    if((key = ftok(pathname,0)) < 0){
        printf("Can\'t generate key\n");
        exit(-1);
    }
    /* Опит да се създаде поделена памет за генерирания ключ,
    ако за този ключ, тя вече съществува, системното извикване
    връща отрицателна стойност. Размера на паметта се определя
    от размера на масива, съдържащ три целочислени променливи,
    правата за достъп са 0666 – право за чтене и запис имат
    всички потребители */
    if((shmid = shmget(key, 3*sizeof(int),
    0666|IPC_CREAT|IPC_EXCL)) < 0){
    /* В случай на грешка се определя дали, възникването й се дължи
       на това, че сегмента вече съществува или по друга причина */
        if(errno != EEXIST){
            /* Ако е по друга причина – работата се прекратява */
            printf("Can\'t create shared memory\n");
            exit(-1);
```

```
            } else {
                /* Ако е поради това, че поделената памет вече съществува,
                   то се прави опит да се получи нейният IPC дескриптор.
                   При успех се изчиства флага за нобходимостта от
                   инициализация на елементите от масива */
                if((shmid = shmget(key, 3*sizeof(int), 0)) < 0){
                    printf("Can\'t find shared memory\n");
                    exit(-1);
                }
                new = 0;
            }
        }
        /* Поделената памет се помества в адресното пространство на текущия
           процес. Обърнете внимание на това, че за да бъде сравнението
           правилно стойността -1 се преобразува в указател към цяло число.*/
        if((array = (int *)shmat(shmid, NULL, 0)) == (int *)(-1)){
            printf("Can't attach shared memory\n");
            exit(-1);
        }
        /* В зависимост от стойността на флага new се извършва или
        инициализиране на масива, или инкрементиране на съответните броячи */
        if(new){
            array[0] = 1;
            array[1] = 0;
            array[2] = 1;
        } else {
            array[0] += 1;
            array[2] += 1;
        }
        /* Отпечатват се новите стойности на броячите, поделената памет се
           изтрива от адресното пространство на текущия процес и порграмата
           приключва */
        printf("Program 1 was spawn %d times,
            program 2 - %d times, total - %d times\n",
            array[0], array[1], array[2]);
        if(shmdt(array) < 0){
            printf("Can't detach shared memory\n");
            exit(-1);
        }
        return 0;
    }
```

Листинг 6.1а. Програма 1 (06_1a.c), илюстрираща работата с поделена памет

```
    /* Програма 2 (06_1b.c), илюстрираща работата с поделена памет */
    /* В програмата се създава поделена памет, която представлява масив от
    три цели числа. В първия елемент на масива се записват броя стартирания
    на програма 1, във вория елемент – броя стартирания на програма 2,
    а в третия – общия брой и за двтете програми */
    #include <sys/types.h>
    #include <sys/ipc.h>
    #include <sys/shm.h>
    #include <stdio.h>
    #include <errno.h>
    int main()
    {
        int *array;   /* Указател към поделената памет */
        int shmid;    /* IPC дескриптор за областта от поделена памет */
        int new = 1;  /* Флаг, показващ необходимостта от инициализация
                         на елементите от масива */
        char pathname[] = "06_1a.c"; /* Името на файла,
            използван за генериране на ключа. Файл с това
            име трябва да съществува в текущата директория */
        key_t key;    /* IPC ключ */
        /* Генериране на IPC ключа (използва се името на файла 06_1a.c в
                текущата директория и номера на инстанцията на областта от
            поделена памет 0 */
        if((key = ftok(pathname,0)) < 0){
            printf("Can\'t generate key\n");
            exit(-1);
        }
```

```
        /* Опит да се създаде поделена памет за генерирания ключ,
        ако за този ключ, тя вече съществува, системното извикване
        връща отрицателна стойност. Размера на паметта се определя
        от размера на масива, съдържащ три целочислени променливи,
        правата за достъп са 0666 – право за чтене и запис имат
        всички потребители */
        if((shmid = shmget(key, 3*sizeof(int),
            0666|IPC_CREAT|IPC_EXCL)) < 0){
        /* В случай на грешка се определя дали, възникването й се дължи
           на това, че сегмента вече съществува или по друга причина */
            if(errno != EEXIST){
                /* Ако е по друга причина – работата се прекратява */
                printf("Can\'t create shared memory\n");
                exit(-1);
            } else {
                /* Ако е поради това, че поделената памет вече съществува,
                   то се прави опит да се получи нейният IPC дескриптор.
                   При успех се изчиства флага за нобходимостта от
                   инициализация на елементите от масива */
                if((shmid = shmget(key, 3*sizeof(int), 0)) < 0){
                    printf("Can\'t find shared memory\n");
                    exit(-1);
                }
                new = 0;
            }
        }
        /* Поделената памет се помества в адресното пространство на текущия
           процес. Обърнете внимание на това, че за да бъде сравнението
           правилно стойността -1 се преобразува в указател към цяло число.*/
        if((array = (int *)shmat(shmid, NULL, 0)) ==
            (int *)(-1)){
            printf("Can't attach shared memory\n");
            exit(-1);
        }
        /* В зависимост от стойността на флага new се извършва или
        инициализиране на масива, или инкрементиране на съответните броячи */
        if(new){
            array[0] = 0;
            array[1] = 1;
            array[2] = 1;
        } else {
            array[1] += 1;
            array[2] += 1;
        }
        /* Отпечатват се новите стойности на броячите, поделената памет се
           изтрива от адресното пространство на текущия процес и порграмата
           приключва */
        printf("Program 1 was spawn %d times,
            program 2 - %d times, total - %d times\n",
        array[0], array[1], array[2]);
        if(shmdt(array) < 0){
            printf("Can't detach shared memory\n");
            exit(-1);
        }
        return 0;
    }
```

Листинг 6.1b. Програма 2 (06_1b.c), илюстрираща работата с поделена памет

Представените програми са идентични. В поделената памет е поместен масив от три цели числа. Първият елемент на масива се използва като брояч за броя стартирания на първата програма – програма 1, втория елемент – за програма 2, третия елемент – сумарно и за двете програми. Необходимо е програмите да могат да различават случая, когато самите те създават поделената памет и случая, когато тя вече е създадена, за да могат

правилно да инициализират елементите на масива. Тези различия се установяват посредством използваното системното извикване *shmget()* с установени флагове *IPC_CREAT* и *IPC_EXCL*. Ако извикването завърши нормално, то поделената памет е създадена. Ако извикването завърши с възникване на грешка и се установи, че стойността на променливата *errno* е *EEXIST*, то споделената памет вече съществува и може да се получи нейният IPC дескриптор, прилагайки същото системно извикване без флагове.

Задача 1: Въведете програмите и ги запишете със съответните им имена *06_1a.c* и *06_1b.c*. Компилирайте и стартирайте няколко пъти всяка от тях. Анализирайте получените резултати.

2. Команди ipcs и ipcrm

Предходните примери показват, че създадената област от поделена памет се съхранява в операционната система, дори тогава, когато не е включена в адресното пространство на нито един процес. От една страна това има определени предимства – не е необходимо взаимодействащите си процеси да съществуват едновременното. От друга стана води и до неудобства. Например, създадения в предходните програми сегмент от поделена памет не би могъл да се използва за преброяване на стартиранията в рамките само на една сесия, тъй като в него се запазва информацията и от предходната сесия. Така програмата изчислява общия брой стартирания за цялото време от момента на зареждане на операционната система. Възможно е за всяка следваща сесия да се създаде нов сегмент от поделена памет, но това трябва да е съобразено с ограниченото количество системни ресурси. В тази връзка може да се приложи съществуващия способ за изтриване на неизползваните System V IPC ресурси, както с помощта на команди на операционната система, така и с помощта на системни извиквания. Заетите от System V IPC ресурси могат да се изтрият от командния ред посредством две команди – *ipcs* и *ipcrm*. Първата от тях (командата *ipcs*) извежда информация за всички налични в системата System V IPC средства, за които потребителят има права за четене: опашки от съобщения, области от поделена памет и семафори. След което може да се използва командата *ipcrm*, за изтриване

от системата на онези от тях, които не се използват.

ИМЕ

```
ipcs
```

СИНТАКСИС

```
ipcs [-asmq] [-tclup]
ipcs [-smq] -i id
ipcs -h
```

ОПИСАНИЕ

```
Командата ipcs е предназначена за получаване на информация за System V IPC
средствата, до които потребителя има права за четене.
Опцията -i позволява да се укаже идентификатор на ресурси. Така се получава
информация само за ресурса притежаващ този идентификатор.
Видът на IPC ресурсите може да се зададе с помоща на следните опции:
–s за семафори; -m за сегменти от поделена памет; -q за опашки от съобщения;
-a за всички ресурси (по подразбиране).
Опции [-tclup] използват се за промяна на съдържанието на изходната
информация. По подразбиране за всяко средство се извежда ключа, IPC
идентификатора, идентификатора на собственика, правата за достъп и редица
други характеристики. Опциите позовляват допълнително да се изведе:
-t времето на приключване на последната операция с IPC средството;
-p идентификатора на процеса, създал ресурса и последния процес, работил с
него;
-c идентификаторите на потребителя създал ресурса, групата на този потребител
и потребителя-собственник;
-l системните ограничение за System V IPC средствата;
-u общото състояние на IPC ресурсите в системата.
Опция -h се използва за получаване на кратка справочна информация.
```

Командата за изтриване на сегмент от поделена памет има вида:

```
ipcrm shm <IPC идентификатор>
```

Задача 2: Изтрийте създадените от вас сегменти от поделена памет, като използвате посочената команда.

ИМЕ

```
ipcrm
```

СИНТАКСИС

```
ipcrm [shm | msg | sem] id
```

ОПИСАНИЕ

```
Командата ipcrm е предназначена за изтриване на System V IPC ресурс от
операционната система. Параметърът id задава IPC идентификатора на изтривания
ресурс, параметъра shm се използва за сегменти от поделена памет, параметъра
msg – опашки от съобщения, параметъра sem – за семафори.
```

Съвет: Ако действието на определи програми се базира на създадени от тях System V IPC средства, не забравяйте преди да ги стартирате да изтриете вече създадени от тях ресурси.

3. Освобождаване на ресурси заети от поделана памет, посредством shmctl()

За изтриване на област от поделена памет може да се използва и системното извикване *shmctl()*. Това системно извикване позволява

напълно да се премахне област от поделена памет по зададен IPC дескриптор, при условие, че потребителя има съответните пълномощия. Системното извикване *shmctl()* позволява да се извършват и други операции над сегмент от поделена памет. Повече информация за него може да се намери в [UM].

ИМЕ

shmctl

ПРОТОТИП

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

ОПИСАНИЕ

Системното извикване shmctl е предназначено за получаване на информация за област с поделена памет, промяна на нейните атрибути и изтриването ѝ от системата. Даденото описание не е пълно, а адаптирано за целите на настоящия курс. Пълно описание може да се намери в UNIX Manual.
В това упражнение системното изивикване shmctl се използва само за изтриване на област от поделена памет от системата. Параметърът shmid е System V IPC дескриптора на сегмента от поделена памет, т. е. стойността, която връща системното изивкване shmget() при създаване на сегмента или при достъп до него по зададения ключ.
В рамките на това упражнение за параметъра cmd винаги ще се задава стойност IPC_RMID – командата за изтриване на сегманта от поделена памет, зададена със своя идентификатор. Параметърът buf за тази команда не е необходим и винаги ще приема стойност NULL.

ВРЪЩАНИ СТОЙНОСТИ

Системното извикване връща стойност 0 при нормално приключване и стойност -1 при възникване на грешка.

4. Системни извиквания fork(), exec() и функция exit(). Особености при работа с поделена памет

Един важен въпрос при използване на поделена памет е въпросът за поведението на сегментите от поделена памет при използване на системните извиквания *fork()*, *exec()* и функцията *exit()*.

При изпълнение на системното извикване *fork()* всички области от поделена памет в адресното пространство на процеса се наследяват от породените процеси. При изпълнение на системно извикване от семейството на *exec()* или на функцията *exit()* всички области от поделена памет, разположени в адресното пространство на процеса се изключват от него, но продължават да съществуват в операционната система.

Задача 3: Напишете самостоятелно две програми, взаимодействащи си чрез поделена памет. Първата програма трябва да създаде сегмента от поделена памет

и да копира в него собственият си изходен текст, а втората – да извлече този текст от там, да го отпечата на екрана и да изтрие сегмента от системата.