

# PAMIĘĆ WSPÓŁDZIELONA I IPC

Kolejnym elementem *InterProcess Communication, IPC* wprowadzonym w ramach *UNIX System V* w postaci (obecnie także POSIX) jest

Współbieżność procesów (czy wątków) wiąże się często z potrzebą okresowej (asynchronicznej) wymiany komunikatów. Żaden z omawianych wcześniej mechanizmów nie spełniłby się najlepiej w tej roli. Po raz pierwszy problem ten znalazł rozwiązanie w

*UNIX System V*

w postaci

*pamięć (współ-)dzielona, ang. shared memory*

Stanowi ona najszybszy zrealizowany w ramach IPC element komunikacji międzyprocesowej, kiedy różne procesy uzyskują dostęp do wspólnego obszaru pamięci.

Przeznaczeniem jest sytuacja taka kiedy kiedy kilka współpracujących procesów do pewnego zestawu danych, który podlega współprzetwarzaniu.

# PAMIĘĆ WSPÓŁDZIELONA IPC

Podobnie jak i w przypadku pozostałych obiektów *IPC*, z poziomu interface'u użytkownika dostępne są polecenia:

- wyświetlające informacje aktualnie zdefiniowanych obiektach *IPC*

```
$ ipcs -m
```

----- Shared Memory Segments -----

key	shmid	owner	perms	bytes	nattch	status
0x00005d8b	32768	root	777	368	1	
0x0056a4d5	688129	kmirota	660	488	1	
0x0056a4d6	720898	kmirota	660	131072	1	

- ponieważ obiekty *IPC* nie są automatycznie usuwane z pamięci operacyjnej, nawet w sytuacji kiedy procesy je tworzące czy wykorzystujące przestały już istnieć – stąd i polecenie

```
$ ipcrm [ -M key | -m id ]
```

umożliwiające usunięcie obszaru pamięci współdzielonej określonego kluczem **key** albo identyfikatorem **id**, w pierwszym lub drugim wariantie składni polecenia.

Przykładowo gdyby chcieć susunąć trzeci w obszarów shared (w listingu) czyli obszar identyfikowany przez key i id , można użyć polecenia

```
$ ipcrm -M 0x0056a4d6
```

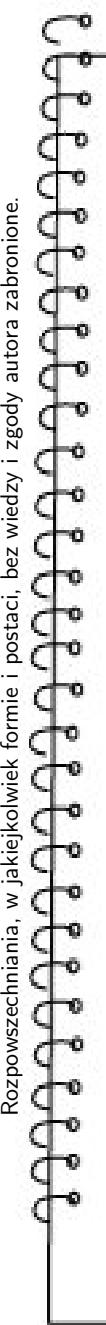
lub

```
$ ipcrm -m 720898
```

z identycznym skutkiem.

**OBLCZENIA RÓWNOLEGLE I SYSTEMY ROZPROSZONE**

KRYSPIN MIROTA, KMIROTA@ATH.BIELSKO.PL



# PAMIĘĆ WSPÓŁDZIELONA I IPC

W momencie tworzenia segmentu dzielnego jądro systemowe przydziela dla jego obsługi, dedykowany obiekt danych

```
struct shmid_ds
{
    struct ipc_perm shm_perm;          /* Uprawnienia do segmentu */
    size_t          shm_segsz;         /* Rozmiar segmentu w bajtach */
    time_t          shm_atime;        /* Ostatnie przyłączenie segmentu */
    time_t          shm_dtime;        /* Ostatnie odłączenie segmentu */
    time_t          shm_ctime;        /* Ostatnia zmiana */
    pid_t           shm_cpid;         /* PID twórcy */
    pid_t           shm_lpid;         /* PID ostatniego shmat()/shmdt() */
    shmatt_t        shm_nattch;       /* Ilość przyłączonych */
                                    /* ... i ewentualnie jeszcze inne */
};

struct ipc_perm
{
    key_t key;             /* key podany w shmget() */
    uid_t uid;              /* UID właściciela */   uid_t cuid;   /* UID twórcy */
    gid_t gid;              /* GID właściciela */   gid_t cgid;  /* GID twórcy */
    unsigned short mode; /* uprawnienia + flagi (SHM_DEST,SHM_LOCKED) */
    unsigned short seq;   /* numer sekwencyjny */
};
```

OBLCZENIA RÓWNOLEGLE I SYSTEMY ROZPROSZONE  
KRYSZPIN MIROTA, KMIROTA@ATH.BIELSKO.PL

# PAMIĘĆ WSPÓŁDZIELONA I IPC

Proces składa żądanie alokacji lub uzyskania dostępu do istniejącego segmentu pamięci współdzielonej wywołaniem funkcji `shmget()`.

## SYNOPSIS

```
#include <sys/ipc.h>
#include <sys/shm.h>
int shmget( key_t key, size_t size, int flag );
key_t key klucz identyfikujacy segment-można wykorzystać tu z ftok()
size_t size rozmiar segmentu w bajtach
int flag maska bitowa IPC_CREAT (utwórz, jeżeli nie istnieje)
IPC_EXCL (utwórz, jeżeli istnienie spowoduje błąd) oraz maska uprawnień
```

## RETURN

int identyfikator segmentu pamięci

## ERRORS

-1

Warto pamiętać, że wywołania funkcji zarządzania procesami skutkują następująco:

- `fork()` proces potomny dziedziczy przyłączone uprzednio segmenty
- `exec()` wszystkie przyłączone segmenty są zwalniane, choć nie są usuwane
- `exit()` wszystkie przyłączone segmenty są zwalniane, choć nie są usuwane

# PAMIĘĆ WSPÓŁDZIELONA IPC

Maska bitowa konstruowana jest operatorem alternatywy bitowej '|', więc bity nie ustawiane inicujemy zerami. Uprawnienia **execute** nie mają zastosowania, więc zawsze będzie '0'.

user	group	other	BIN	OCT	DEC	HEX
r w x	r w x	r w x				
1 0 ø	0 0 ø	0 0 ø	100000000	0400	256	0x100
0 1 ø	0 0 ø	0 0 ø	010000000	0200	128	0x80
1 1 ø	0 0 ø	0 0 ø	110000000	0600	384	0x180
user	group	other				
r w x	r w x	r w x				
0 0 ø	1 0 ø	0 0 ø	000100000	040	32	0x20
0 0 ø	0 1 ø	0 0 ø	000010000	020	16	0x10
0 0 ø	1 1 ø	0 0 ø	000110000	060	48	0x30
user	group	other				
r w x	r w x	r w x				
0 0 ø	0 0 ø	1 0 ø	000000100	04	4	0x4
0 0 ø	0 0 ø	0 1 ø	000000010	02	2	0x2
0 0 ø	0 0 ø	1 1 ø	000000110	06	6	0x6
user	group	other				
r w x	r w x	r w x				
1 0 ø	1 0 ø	1 0 ø	100100100	0444	292	0x124
0 1 ø	0 1 ø	0 1 ø	010010010	0222	146	0x92
1 1 ø	1 1 ø	0 1 ø	110110110	0666	438	0x1B6

# PAMIĘĆ WSPÓŁDZIELONA I IPC

Pobranie informacji o wybranym segmencie pamięci współdzielonej a także niektóre działanie sterujące dostępne są za pośrednictwem funkcji `shmctl()`.

## SYNOPSIS

```
#include <sys/ipc.h>
#include <sys/shm.h>
int shmctl( int id,int cmd,struct shmid_ds *buffer );
int id identyfikator współdzielonego segmentu, którego dotyczy odwołanie
struct shmid_ds *buffer struktura w której zwarcane są informacje
int cmd komenda, rodzaj działania
```

## RETURN

int identyfikator segmentu pamięci

## ERRORS

-1

Dopuszczalne są trzy podstawowe operacje `cmd` za pomocą `shmctl()`

`IPC_STAT`, pobiera informacje o segmencie i kopiuje

do struktury `struct shmid_ds *buffer`

`IPC_SET`, kopiuje informacje podane w parametrze `struct shmid_ds *buffer`  
do struktury systemowej

`IPC_RMID` zaznacza segment do usunięcia

(będzie usunięta po odłączeniu przez wszystkich użytkowników)

pozostałe są zależne od platformy systemowej. `LINX` dopuszcza jeszcze `IPC_INFO`, `SHM_INFO`, `SHM_STAT` (jak `IPC_STAT`) oraz `SHM_LOCK` i `SHM_UNLOCK` (blokuje i zwalnia dostęp).

**OBLCZENIA RÓWNOLEGŁE I SYSTEMY ROZPROSZONE**

KRYSPIN MIROTA, KMIROTA@ATH.BIELSKO.PL



# PAMIĘĆ WSPÓŁDZIELONA I IPC

Poniższy kod źródłowy utworzy segment pamięci współdzielonej, następnie pobierze informacje o nim i zwolni pamięć.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
```

Stała wykorzystywana w funkcji **ftok()**

```
#define PROJECTID 0xABCDEF
int main( void )  
{
```

```
    key_t key;  
    int id, size, flag;  
    struct shmid_ds buffer;
```

Inicjujemy wartości aktualne dla przyszłego wywołania **shmget()**

```
key = ftok( "/tmp",PROJECTID );  
flag = IPC_CREAT | 0x100 | 0x80;  
size = 1024*64;
```

Tworzymy segment pamięci

```
id = shmget( key, size, flag );
```

**OBLCZENIA RÓWNOLEGLE I SYSTEMY ROZPROSZONE**

KRYSPIN MIROTA, KMIROTA@ATH.BIELSKO.PL

# PAMIĘĆ WSPÓŁDZIELONA I IPC

Jeżeli próba zakończona sukcesem

```
if( id>0 )  
{
```

Wyświetlenie informacji diagnostycznych

```
printf( "SEGMENT [key=0x%x,id=%u],process=%u\n",  
       id,(unsigned)key,(unsigned)getpid() );
```

Informacje o segmencie

```
(void)shmctl( id,IPC_STAT,&buffer );  
printf( "właściciel %u\n",(unsigned)(buffer.shm_cpid) );  
printf( "rozmiar %u [B]\n",(unsigned)(buffer.shm_segsz) );  
printf( "%s\n",ctime( &(buffer.shm_ctime) ) );
```

Zwalniamy i kończymy

```
(void)shmctl( id,IPC_RMID,&buffer );
```

```
}
```

```
else{ perror( "!.!.!.shmget()..." ); exit( 1 ); }
```

```
return 0;
```

```
}
```

# PAMIĘĆ WSPÓŁDZIELONA I PC

Zanim proces będzie mógł działać na segmencie wspólnym pamięci, najpierw musi dokonać jego przyłączenia czyli odwzorowania na wskazanie adresu początkowego. Uzyskuje się to za pośrednictwem **shmat()** (zwolnienie następuje przez wywołanie **shmdt()**), efekt – w sensie operacyjnym – jest analogiczny do wywołanie funkcji przydzielających pamięć w obrębie procesu czyli **malloc()** czy **calloc()**.

## SYNOPSIS

```
#include <sys/types.h>
#include <sys/shm.h>
void *shmat( int id,const void *address,int flag );
int shmdt( const void *address );
int id identyfikator obszaru wspólnego pamięci
const void *address proponowany adres bądź NULL (wybiera system)
int flag maska bitowa określająca sposób tworzenia i użycia
(SHM_RDONLY, tylko do odczytu)
```

## RETURN

void\* wskazanie do obszaru pamięci wspólnej  
int - w przypadku shmdt() - 0 jeżeli sukces

## ERRORS

-1



# PAMIĘĆ WSPÓŁDZIELONA I IPC

Przygotujemy teraz kod dwóch programów – pierwszy utworzy segment pamięci współdzielonej **IPC** i zainicjuje go ciągiem liczb Davida Hilberta, czyli liczb postaci

$$4n + 1, \text{ gdzie } n=0,1,2,3,\dots$$

```
#include <stdio.h>      #include <stdlib.h>
#include <time.h>        #include <unistd.h>
#include <sys/ipc.h> #include <sys/shm.h>
#define PROJECTID 0xFF
#define n 100
int main( void )
{
    key_t key;
    int id, size, flag;
    unsigned i;
```

I oczywiście zmienna za pośrednictwem,  
której będziemy się odwoływać do segmentu wspólnego

```
unsigned *array;
printf( "GENERATOR [%u]\n", (unsigned) getpid() );
key = ftok( "/tmp", PROJECTID );
flag = IPC_CREAT | 0x1B6;
size = n*sizeof( unsigned );
id = shmget( key, size, flag );
```

*OBLCZENIA RÓWNOLEGŁE I SYSTEMY ROZPROSZONE  
KRYSZPIN MIROTA, KMIROTA@ATH.BIELSKO.PL*

# PAMIĘĆ WSPÓŁDZIELONA I IPC

Jeżeli udało się zainicjować segment współdzielony, to

```
if( id>0 )
```

```
{
```

```
    printf( "...utworzono segment wspólny [%u][0x%x]\n\t",id,key );
```

Sposób dostępu do obszaru współdzielonego pamięci

```
array = (unsigned*)shmat( id,NULL,0 );
```

...po przyłączeniu, odwołanie w sposób tradycyjny, jak do wskazania określonego typu

```
for( i=0;i<n;i++ )
```

```
{
```

```
    *(array+i)=(4*i+1);
```

```
    printf( "." ); fflush( stdout );
```

```
}
```

ponownie odłączamy segment współdzielony

```
shmdt( (void*)array );
```

```
printf( "\n...zakończono inicjację segmentu wspólnego\n" );
```

```
}
```

```
else{ perror( "!.!.!.shmget()..." ); exit( 1 ); }
```

```
return 0;
```

```
}
```

Zauważmy, że generator pozostawia zainicjowany segment wspólny w pamięci.



# PAMIĘĆ WSPÓŁDZIELONA I IPC

Kod użytkownika segmentu wspólnego będzie przedstawiał się podobnie. Nie ma powodu aby we fragmencie początkowym różnił się istotnie.

```
#include <stdio.h>      #include <stdlib.h>
#include <time.h>        #include <unistd.h>
#include <sys/ipc.h> #include <sys/shm.h>

#define PROJECTID 0xFF
#define n 100

int main( void )
{
    key_t key;
    int id, size, flag;
    struct shmid_ds buffer;
    unsigned i, *array;

    key = ftok( "/tmp", PROJECTID );
    flag = IPC_CREAT | 0x1B6;
    size = n*sizeof( unsigned );
    id = shmget( key, size, flag );
```

# PAMIĘĆ WSPÓŁDZIELONA I IPC

```
if( id>0 )
{
    Poprostu przyłączmy
    array = (unsigned*)shmat( id,NULL,0 );
    i korzystamy w zwyczajowy sposób
    printf( "100 liczb D.HILBERTa odczytanych z [%u] [0x%x]\n",id,key );
    for( i=0;i<n;i++ )
    {
        printf( "%10u",*(array+i) );
        ... jeszcze tylko drobna korekta listingu
        printf( "%c",!((i+1)%5)?('\n'):(' ') );
    }
    printf( "%c",'\'n' );
    Po wykorzystaniu odłączamy
    shmdt( (void*)array );
    i usuwamy z pamięci
    (void)shmctl( id,IPC_RMID,&buffer );
}
else{ perror( "!.!.!.shmget()..." ); exit( 1 ); }

return 0;
}
```

OBLCZENIA RÓWNOLEGŁE I SYSTEMY ROZPROSZONE  
KRYSZPIN MIROTA, KMIROTA@ATH.BIELSKO.PL

# PAMIĘĆ WSPÓŁDZIELONA I IPC

Sprawdźmy przed uruchomieniem generatora liczb Hilberta

```
$ ipcs -m
```

```
----- Shared Memory Segments -----
```

key	shmid	owner	perms	bytes	nattch	status
0x00005d8b	32768	root	777	368	1	

a więc brak obszarów współdzielonych pamięci użytkownika (tutaj: **kmirota**)

Jeżeli teraz uruchomimy proces generator

```
$ ./generator
```

```
GENERATOR [9631]
```

```
...utworzono segment wspólny [9273345] [0xff060481]
```

```
.....
```

```
.....
```

```
...zakończono inicjację segmentu wspólnego
```

Jeżeli ponownie sprawdzimy dostępne obiekty *IPC* typu shared memory

```
$ ipcs -m
```

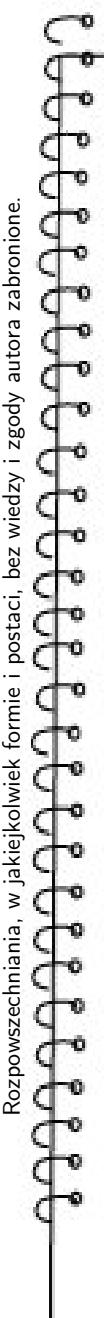
```
----- Shared Memory Segments -----
```

key	shmid	owner	perms	bytes	nattch	status
0x00005d8b	32768	root	777	368	1	
0xff060481	9273345	<b>kmirota</b>	666	400	0	

Zatem zgodnie z informacją jaką uzyskaliśmy z procesu, powstał nowy segment współdzielony o identyfikatorze 9273345 i kluczu 0xff060481.

*OBLCZENIA RÓWNOLEGLE I SYSTEMY ROZPROSZONE*

KRYSPIN MIROTA, KMIROTA@ATH.BIELSKO.PL



# PAMIĘĆ WSPÓŁDZIELONA IPC

Uruchamiając teraz proces kilenta, wyłącznie czytającego z segmentu otrzymujemy.

```
$ ./client
```

```
100 liczb D.HILBERTa odczytanych z [9273345] [0xff060481]
```

1	5	9	13	17
21	25	29	33	37
41	45	49	53	57
61	65	69	73	77
81	85	89	93	97
101	105	109	113	117
121	125	129	133	137
141	145	149	153	157
161	165	169	173	177
181	185	189	193	197
201	205	209	213	217
221	225	229	233	237
241	245	249	253	257
261	265	269	273	277
281	285	289	293	297
301	305	309	313	317
321	325	329	333	337
341	345	349	353	357
361	365	369	373	377
381	385	389	393	397

OBLICZENIA RÓWNOLEGŁE I SYSTEMY ROZPROSZONE  
KRYSZPIN MIRETA, KMIROTA@ATH.BIELSKO.PL

# PAMIĘĆ WSPÓŁDZIELONA I IPC

Gdyby sprawdzić ponownie

```
$ ipcs -m
```

----- Shared Memory Segments -----

key	shmid	owner	perms	bytes	nattch	status
0x00005d8b	32768	root	777	368	1	

czyli brak. Uruchamiając ponownie klienta, nie otrzymamy błędu ale

```
$ ./client
```

100 liczb D.HILBERTa odczytanych z [11403265] [0xff060481]

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

...itd., itd., itd.

ponieważ obszar utworzony przez generator został usunięty wcześniejszym wywołaniem klienta. W efekcie powstał nowy, nie zainicjowany.