6 Pamięć dzielona

6.1 Wprowadzenie

Pamięć dzielona, inaczej wspólna, (ang. *shared memory*) jest zasobem umożliwiającym najszybszy sposób komunikacji między procesami, z reguły wymagający jednak dodatkowego mechanizmu synchronizacji. Schemat korzystania z pamięci dzielonej jest na ogół następujący. Najpierw jeden z procesów tworzy (rezerwuje) segment pamięci, podając jego żądany rozmiar i prawa dostępu. Następnie dowiązuje go, tzn. odwzorowuje w swój obszar danych. Do tak utworzonego segmentu mogą sięgać inne procesy wykonując odpowiednie dowiązania, o ile posiadają do tego uprawnienia. Każdy proces uzyskuje dostęp do obszaru pamięci dzielonej względem miejsca wyznaczonego przez adres jego dowiązania. Dostęp do pamięci dzielonej najczęściej kontroluje się przy pomocy semaforów. Kiedy proces zakończy korzystanie z segmentu pamięci dzielonej, powinien go ,,odwiązać", tzn. usunąć jego dowiązanie. Po zakończeniu używania segmentu przez wszystkie korzystające z niego procesy powinien on zostać usunięty (zwykle za to odpowiedzialny jest proces, który dany segment utworzył).

W systemie UNIX pamięć dzielona należy, tak jak semafory, do grupy tzw. mechanizmów IPC (ang. *interprocess communications*). Podobnie jak dla semaforów, również dla pamięci dzielonej istnieją użyteczne komendy poziomu powłoki:

`ipcs -m`	podaje informację na temat utworzonych segmentów pamięci dzielonej;
`ipcs -ml`	podaje informację na temat ograniczeń systemowych dotyczących pamięci dzielonej;
`ipcrm shm` *shmid*	usuwa segment pamięci o identyfikatorze *shmid*.

6.2 Tworzenie (uzyskiwanie dostępu do) pamięci dzielonej

Do tworzenia segmentu pamięci dzielonej, albo uzyskiwania dostępu do istniejącego już segmentu, służy funkcja systemowa shmget. Podobnie jak odpowiednia funkcja do tworze-

Pliki włączane	`<sys/types.h>, <sys/ipc.h>, <sys/shm.h>`		
Prototyp	`int shmget(key_t key, int size, int shmflg);`		
Zwracana wartość	Sukces	Porażka	Czy zmienia `errno`
	Identyfikator pamięci dzielonej	-1	Tak

nia semaforów, potrzebuje ona całkowitoliczbowego klucza jako pierwszego argumentu. Klucz ten powinien jednoznacznie identyfikować dany segment pamięci dzielonej i może być uzyskiwany np. za pomocą funkcji ftok (podobnie jak dla semaforów). Parametr size określa rozmiar segmentu w bajtach (jeżeli uzyskuje się dostęp do istniejącego już segmentu, to jego wartością może być zero), natomiast parametr shmflg jest flagą określającą warunki tworzenia segmentu oraz prawa dostępu do niego (podobnie jak plików za wyjątkiem praw wykonywania). Aby utworzyć nowy segment, argument shmflg należy ustawić na znacznik `IPC_CREAT`. Jeżeli znacznik ten zostanie dodatkowo połączony (sumą bitową)

ze znacznikiem `IPC_EXCL`, to funkcja zakończy się porażką w przypadku, gdy dany segment już istnieje. Znaczniki łączy się sumą bitową z prawami dostępu, np. `IPC_CREAT|0666`. Funkcja shmget zakończona sukcesem powoduje utworzenie segmentu pamięci dzielonej, albo uzyskanie dostępu, jeżeli segment dla danego klucza już istnieje, i zwraca całkowitoliczbowy indentyfikator tego segmentu. Identyfikator ten następnie służy do odnoszenia się do danego segmentu przez funkcje systemowe operujące na pamięci dzielonej.

6.3 Sterowanie pamięcią dzieloną

Funkcja systemowa shmctl umożliwia wykonywanie pewnych operacji na segmentach pamięci dzielonej. Pierwszy parametr jest identyfikatorem segmentu pamięci (zwracanym

Pliki włączane	`<sys/types.h>, <sys/ipc.h>, <sys/shm.h>`		
Prototyp	`int shmctl(int shmid, int cmd, struct shmid_ds *buf);`		
Zwracana wartość	Sukces	Porażka	Czy zmienia `errno`
	0	−1	Tak

przez funkcję shmget), drugi określa operację wykonywaną przez funkcję, a trzeci jest wskaźnikiem na systemową strukturę shmid_ds do przechowywania informacji o segmencie (zdefiniowaną w pliku `<sys/shm.h>`).

- Wybrane polecenia `cmd`:

`IPC_STAT` zwraca bieżącą wartość struktury shmid_ds związaną z danym segmentem pamięci,

`IPC_RMID` powoduje usunięcie danego segmentu pamięci; argument buf powinien wówczas przyjąć wartość: `(struct shmid_ds *)0`.

6.4 Operacje na pamięci dzielonej

Do dowiązywania (odwzorowywania) segmentu pamięci dzielonej do przestrzeni danych procesu służy funkcja shmat. Pierwszym jej parametrem jest identyfikator segmentu

Pliki włączane	`<sys/types.h>, <sys/ipc.h>, <sys/shm.h>`		
Prototyp	`void *shmat(int shmid, void *shmaddr, int shmflg);`		
Zwracana wartość	Sukces	Porażka	Czy zmienia `errno`
	wskaźnik do dowiązanego segmentu pamięci dzielonej	−1	Tak

(zwracany przez funkcję semget). Parametr shmaddr służy do określania położenia segmentu pamięci dzielonej: jeżeli jest równy `NULL`, to system sam wybiera odpowiedni adres dowiązania (**zalecane!**), natomiast jeśli jest różny od `NULL`, to jego wartość zostanie użyta jako adres dowiązania. Trzeci parametr służy do określania warunków dowiązania oraz

uprawnień do segmentu. Dla shmflg = 0 zostaną użyte domyślne właściwości. Domyślnie dowiązany segment jest dostępny zarówno do odczytu, jak i zapisu. Ustawienie shmflg na znacznik SHM_RDONLY spowoduje, że dany segment będzie dostępny tylko do odczytu (zalecane w procesach, które mają tylko czytać dane z pamięci dzielonej). Więcej szczegółow można znaleźć w podręczniku systemowym man.

Do usuwania dowiązań segmentów pamięci dzielonej do obszaru danych procesu służy funkcja shmdt. Ma ona tylko jeden parametr, shmaddr, który powinien być adresem (wskaźnikiem) dowiązanego wcześniej segmentu.

Pliki włączane	`<sys/types.h>, <sys/ipc.h>, <sys/shm.h>`		
Prototyp	`int shmdt(void *shmaddr);`		
Zwracana wartość	Sukces	Porażka	Czy zmienia `errno`
	0	−1	Tak

ĆWICZENIE 7: Producent–Konsument: Pamięć Dzielona i Semafory

Przy pomocy **pamięci dzielonej** oraz **semaforów** systemu UNIX zaimplementować problem **„Producenta–Konsumenta"** z **ćwiczenia 4**. Zamiast potoku użyć N-elementowego bufora cyklicznego (tzn. po dojściu do końca bufora wracamy na jego początek) umieszczonego w pamięci dzielonej, gdzie elementem bufora jest pewna ustalona porcja bajtów. Dostęp do wspólnego bufora synchronizować przy pomocy semaforów.

Podobnie jak dla semaforów, stworzyć własną bibliotekę funkcji do obsługi pamięci dzielonej.

→ *Podać rozwiązanie problemu w pseudokodzie posługując się notacją języka C.*

→ *Z własnych bibliotek funkcji do obsługi semaforów i pamięci dzielonej stworzyć bibliotekę statyczną oraz bibliotekę dzieloną (ang. shared library); umieścić je w podkatalogu* `./lib` *(patrz przykład w* `StartSO`*).*

→ *Uogólnić zadanie na przypadek wielu producentów i wielu konsumentów (przynajmniej w pseudokodzie).*