Temat zajęć: Mechanizmy IPC: pamięć współdzielona.

Czas realizacji zajęć:	*90 min.*
Zakres materiału, jaki zostanie zrealizowany podczas zajęć:	Tworzenie i obsługa pamięci współdzielonej, przyłączanie/odłączanie pamięci współdzielonej do/od procesu, parametry pamięci współdzielonej, implementacja przykładowych programów obsługi pamięci współdzielonej

I. Pamięć współdzielona.

Pamięć współdzielona jest specjalnie utworzonym segmentem wirtualnej przestrzeni adresowej, do którego dostęp może mieć wiele procesów. Jest to najszybszy sposób komunikacji pomiędzy procesami. Podstawowy schemat korzystania z pamięci współdzielonej wygląda następująco: jeden z procesów tworzy segment pamięci współdzielonej, dowiązuje go powodując jego odwzorowanie w bieżący obszar danych procesu, opcjonalnie zapisuje w stworzonym segmencie dane. Następnie, w zależności od praw dostępu inne procesy mogą odczytywać i/lub zapisywać wartości w pamięci współdzielonej. Każdy proces uzyskuje dostęp do pamięci współdzielonej względem miejsca wyznaczonego przez jego adres dowiązania, stąd każdy proces korzystając z tych samych danych używa innego adresu dowiązania. W przypadku współbieżnie działających procesów konieczne jest najczęściej synchronizowanie dostępu np. za pomocą semaforów. Kończąc korzystanie z segmentu pamięci proces może ten segment odwiązać, czyli usunąć jego dowiązanie. Kiedy wszystkie procesy zakończą korzystanie z segmentu pamięci współdzielonej, za jego usunięcie najczęściej odpowiedzialny jest proces, który segment utworzył.

Podczas tworzenia segmentu pamięci współdzielonej tworzona jest systemowa struktura danych o nazwie `shmid_ds`. Definicję tej obsługiwanej przez system struktury można znaleźć w pliku nagłówkowym `<sys/shm.h>`.

Funkcje operujące na pamięci współdzielonej zdefiniowane są w plikach: `<sys/ipc.h>` i `<sys/shm.h>`.

II. Funkcje systemowe obsługujące pamięć współdzieloną i ich argumenty.

- **`int shmget (key_t key, size_t size, int shmflags)`**

Wartości zwracane:
poprawne wykonanie funkcji: identyfikator segmentu pamięci współdzielonej
zakończenie błędne: -1

Możliwe kody błędów (*errno*) w przypadku błędnego zakończenie funkcji:
EACCES – brak brak praw dostępu
ENOENT – segment pamięci nie istnieje
EIDRM – segment pamięci został usunięty
EINVAL – nieprawidłowy rozmiar segmentu pamięci
ENOMEM – nie ma wystarczająco dużo miejsca by stworzyć segment pamięci współdzielonej
EEXIST – segment pamięci współdzielonej istnieje

Argumenty funkcji:
key – wartość klucza, który identyfikuje segment pamięci współdzielonej (podobnie jak w przypadku kolejek komunikatów może to być dowolna liczba lub stała **IPC_PRIVATE**)
size – wielkość segmentu pamięci współdzielonej (w bajtach)

shmflags – prawa dostępu do pamięci współdzielonej (z prawami dostępu mogą zostać użyte znaczniki IPC_CREAT, IPC_EXCL, ich działanie jest analogiczne jak w przypadku funkcji **msgget**)

UWAGI:
Funkcja **shmget** służy do tworzenia segmentu pamięci współdzielonej i do uzyskiwania dostępu fo już istniejących segmentów pamięci. W drugim przypadku wartością parametru *size* może być 0, ponieważ rozmiar segmentu został już wcześniej zadeklarowany przez proces, który go utworzył.

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

Wartości zwracane:
poprawne wykonanie funkcji: 0
zakończenie błędne: -1

Argumenty funkcji:
shmid – identyfikator pamięci współdzielonej
cmd – stała specyfikująca rodzaj operacji
- *cmd* = IPC_STAT – pozwala uzyskać informację o stanie pamięci współdzielonej
- *cmd* = IPC_SET – pozwala zmienić parametry segmentu pamięci
- *cmd* = IPC_RMID – pozwala usunąć segment pamięci współdzielonej z systemu

buf - wskaźnik na zmienną strukturalną przez którą przekazywane są parametry operacji

UWAGI:
Funkcja odpowiada funkcji **msgctl**. Przy próbie usunięcia segmentu odwzorowanego na przestrzeń adresową procesu system odpowiada komunikatem o błędzie.
Jeśli w wywołaniu funkcji użyje się stałej IPC_RMID, to wartość argumentu *buf* należy wyzerować, rzutując 0 na typ (`shmid_ds *`).

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

Wartości zwracane:
poprawne wykonanie funkcji: wskaźnik do segmentu danych, do którego jest dowiązana pamięć współdzielona.
zakończenie błędne: -1

Argumenty funkcji:
shmid – identyfikator pamięci współdzielonej zwracany przez funkcję **shmget**
shmaddr – adres dla tworzonego segmentu pamięci współdzielonej lub wartość NULL, która powoduje, że segment dołączany jest w miejscu wybranym przez system (użytkownik nie musi znać rozmieszczenia programu w pamięci)
shmflg – określa uprawnienia do segmentu pamięci współdzielonej i specjalne warunki dowiązania

UWAGI:
Ponieważ pamięć jest alokowana przy wywołaniu funkcji **shmat** nie ma potrzeby używania funkcji **malloc** przy umieszczaniu danych w segmencie.
Domyślnie dowiązane segmenty są dostępne w trybie do zapisu i odczytu. W przypadku gdy segment ma segmentem tylko do odczytu, argument *shmflg* można połączyć operatorem **OR** ze znacznikiem SHM_RDONLY. Natomiast gdy dla *shmflg* jest ustawiony znacznik SHM_RND, to przy wywołaniu funkcji adres *shmaddr* jest zaokrąglany w dół do granicy strony w pamięci, a w przeciwnym razie pobierana jest wartość podana jako argument wejściowy.

- **`char* shmdt (char* shmaddr)`**

  Wartości zwracane:
  poprawne wykonanie funkcji: 0
  zakończenie błędne: -1

  Argumenty funkcji:
  shmaddr – adres stworzonego segmentu pamięci współdzielonej

  UWAGI:
  Odłączenie segmentu pamięci współdzielonej. Odłączenie to powinno nastąpić po zakończeniu pracy z danym segmentem. Po wywołaniu funkcji **shmdt** licznik dołączeń do segmentu jest zmniejszany o 1. Przykład usunięcia segmentu pamięci:

```
struct shmid_ds shm_desc;
shmctl(shm_id, IPC_RMID, shm_desc)
```

III. Przykładowe programy zapisujące/odczytujące dane z segmentów pamięci współdzielonej.

Listing 1 przedstawia program, w którym następuje cykliczny zapis bufora umieszczonego we współdzielonym obszarze pamięci. Listing przedstawia program, w którym jest analogiczny odczyt bufora cyklicznego.

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

   #define MAX 10
6
   main(){
      int shmid, i;
9     int *buf;
      shmid = shmget(45281, MAX*sizeof(int), IPC_CREAT|0600);
12    if (shmid == -1){
            perror("Utworzenie segmentu pamieci wspoldzielonej");
            exit(1);
15    }
      buf = (int*)shmat(shmid, NULL, 0);
18    if (buf == NULL){
            perror("Przylaczenie segmentu pamieci wspoldzielonej");
            exit(1);
21     }
      for (i=0; i<10000; i++)
24    buf[i%MAX] = i;
   }
```

Listing 1: Zapis bufora cyklicznego

Opis programu: W linii 11 tworzony jest segment współdzielonej pamięci o kluczu 45281, o rozmiarze MAX*sizeof(int) i prawach do zapisu i odczytu przez właściciela. Jeśli obszar o takim kluczu już istnieje, zwracany jest jego identyfikator, czyli nie jest tworzony nowy obszar i tym samym rozmiar podany w drugim parametrze oraz prawa dostępu są ignorowane. W linii 17 utworzony segment włączony zostaje do segmentów danego procesu i zwracany jest adres tego segmentu. Zwrócony adres podstawiany jest pod zmienną *buf*. *Buf* jest zatem adresem tablicy o rozmiarze MAX i typie składowym int. Pętla w liniach 23–24 oznacza cykliczny zapis tego bufora,

tzn. indeks pozycji, na której zapisujemy jest równy *i*%MAX, czyli zmienia się cyklicznie od 0 do MAX-1.

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

   #define MAX 10
6
   main(){
      int shmid, i;
9     int *buf;
      shmid = shmget(45281, MAX*sizeof(int), IPC_CREAT|0600);
12    if (shmid == -1){
            perror("Utworzenie segmentu pamieci wspoldzielonej");
            exit(1);
15    }
      buf = (int*)shmat(shmid, NULL, 0);
18    if (buf == NULL){
            perror("Przylaczenie segmentu pamieci wspoldzielonej");
            exit(1);
21    }
      for (i=0; i<10000; i++)
24          printf("Numer: %5d Wartosc: %5d\n", i, buf[i%MAX]);
   }
```

Listing 2: Odczyt bufora cyklicznego

Opis programu: Powyższy program jest analogiczny, jak program na listingu 2, przy czym w pętli w liniach 23–24 następuje cykliczny odczyt, czyli odczyt z pozycji w buforze, zmieniającej się cyklicznie od 0 do MAX-1.

IV. Zadania do samodzielnego wykonania.

1) Napisać dwa programy komunikujące się poprzez pamięć współdzieloną:
   - W nieskończonej pętli wpisuje do współdzielonej pamięci na przemian napisy: "haaaa" i "hoooooo" (w to samo miejsce).
   - W nieskończonej pętli odczytuje z współdzielonej pamięci napis i sprawdza czy jest to "haaaa" lub "hoooooo". Jeżeli napis jest różny od tych napisów, to powinien pojawić się komunikat o błędzie i wartość błędnego napisu.

V. Literatura.

[HGS99] Havilland K., Gray D., Salama B., *Unix - programowanie systemowe*, ReadMe, 1999

[Roch97] Rochkind M.J., *Programowanie w systemie UNIX dla zaawansowanych*, WNT, 1997

[NS99] Neil M., Stones R,*Linux. Programowanie*, ReadMe, 1999

[MOS02] Mitchell M., Oldham J., Samuel A.,*Linux. Programowanie dla zaawansowanych*, ReadMe,2002

[St02] Stevens R.W., *Programowanie w środowisku systemu UNIX*, WNT, 2002