Zaawansowane programowanie w C++

Michał Tanaś, PhD

Adam Mickiewicz University, Faculty of Physics

http://www.amu.edu.pl/~mtanas

Michal.Tanas@amu.edu.pl

Programy wielowątkowe – IPC

IPC – System V Inter-Process Communication

- Standard komunikacji pomiędzy procesami/wątkami stworzony przez AT&T w 1983
- Umożliwia komunikację pomiędzy wątkami lub niezależnymi procesami.
- Dostępny dla większości Unixów (może wymagać driverów w jądrze).

Programy wielowątkowe – IPC

IPC – System V Inter-Process Communication

- Standard komunikacji pomiędzy procesami/wątkami stworzony przez AT&T w 1983
- Umożliwia komunikację pomiędzy wątkami lub niezależnymi procesami.
- Dostępny dla większości Unixów (może wymagać driverów w jądrze).

Programy wielowątkowe – IPC

IPC – System V Inter-Process Communication

- Standard komunikacji pomiędzy procesami/wątkami stworzony przez AT&T w 1983
- Umożliwia komunikację pomiędzy wątkami lub niezależnymi procesami.
- Dostępny dla większości Unixów (może wymagać driverów w jądrze).

Programy wielowątkowe – IPC

IPC – podstawowe zasady

- Obiekty IPC istnieją **niezależnie** od procesów.
- Obiekty IPC są identyfikowane przez unikalny klucz.
- Obiekty IPC posiadają prawa dostępu identyczne jak pliki.
- Możliwe jest dokonywanie operacji na obiektach IPC przy pomocy programów narzędziowych z poziomu shella (ipcs, ipcrm).
- IPC określa standard komunikacji pomiędzy wątkami, same wątki trzeba utworzyć przy pomocy fork lub clone.

Programy wielowątkowe – IPC

IPC – podstawowe zasady

- Obiekty IPC istnieją **niezależnie** od procesów.
- Obiekty IPC są identyfikowane przez unikalny klucz.
- Obiekty IPC posiadają prawa dostępu identyczne jak pliki.
- Możliwe jest dokonywanie operacji na obiektach IPC przy pomocy programów narzędziowych z poziomu shella (ipcs, ipcrm).
- IPC określa standard komunikacji pomiędzy wątkami, same wątki trzeba utworzyć przy pomocy fork lub clone.

Programy wielowątkowe – IPC

IPC – podstawowe zasady

- Obiekty IPC istnieją **niezależnie** od procesów.
- Obiekty IPC są identyfikowane przez unikalny klucz.
- Obiekty IPC posiadają prawa dostępu identyczne jak pliki.
- Możliwe jest dokonywanie operacji na obiektach IPC przy pomocy programów narzędziowych z poziomu shella (ipcs, ipcrm).
- IPC określa standard komunikacji pomiędzy wątkami, same wątki trzeba utworzyć przy pomocy fork lub clone.

Programy wielowątkowe – IPC

IPC – podstawowe zasady

- Obiekty IPC istnieją **niezależnie** od procesów.
- Obiekty IPC są identyfikowane przez unikalny klucz.
- Obiekty IPC posiadają prawa dostępu identyczne jak pliki.
- Możliwe jest dokonywanie operacji na obiektach IPC przy pomocy programów narzędziowych z poziomu shella (ipcs, ipcrm).
- IPC określa standard komunikacji pomiędzy wątkami, same wątki trzeba utworzyć przy pomocy fork lub clone.

Programy wielowątkowe – IPC

IPC – podstawowe zasady

- Obiekty IPC istnieją **niezależnie** od procesów.
- Obiekty IPC są identyfikowane przez unikalny klucz.
- Obiekty IPC posiadają prawa dostępu identyczne jak pliki.
- Możliwe jest dokonywanie operacji na obiektach IPC przy pomocy programów narzędziowych z poziomu shella (ipcs, ipcrm).
- IPC określa standard komunikacji pomiędzy wątkami, same wątki trzeba utworzyć przy pomocy fork lub clone.

Programy wielowątkowe – IPC

IPC – udostępniane mechanizmy:

- msg – kolejki komunikatów (POSIX Message Queues)
- shm – współdzielone segmenty pamięci (Shared Memory Segments)
- sem – semafory (Semaphores)

Programy wielowątkowe – IPC

IPC – udostępniane mechanizmy:

- msg – kolejki komunikatów (POSIX Message Queues)
- shm – współdzielone segmenty pamięci (Shared Memory Segments)
- sem – semafory (Semaphores)

Programy wielowątkowe – IPC

IPC – udostępniane mechanizmy:

- msg – kolejki komunikatów (POSIX Message Queues)
- shm – współdzielone segmenty pamięci (Shared Memory Segments)
- sem – semafory (Semaphores)

Programy wielowątkowe – IPC/msg

Kolejki komunikatów:

- Umożliwiają przesyłanie komunikatów pomiędzy wątkami lub procesami.
- Udostępniają funkcjonalność podobną do Unix Domain Sockets.

Programy wielowątkowe – IPC/msg

Kolejki komunikatów:

- Umożliwiają przesyłanie komunikatów pomiędzy wątkami lub procesami.
- Udostępniają funkcjonalność podobną do Unix Domain Sockets.

Programy wielowątkowe – IPC/msg

```
int msgget(key_t key, int fl);
```

Tworzy lub udostępnia istniejącą kolejkę komunikatów.

- Zwraca lokalny identyfikator kolejki komunikatów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowej kolejki (domyślnie: udostępnienie istniejącej)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/msg

```
int msgget(key_t key, int fl);
```

Tworzy lub udostępnia istniejącą kolejkę komunikatów.

- Zwraca lokalny identyfikator kolejki komunikatów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowej kolejki (domyślnie: udostępnienie istniejącej)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/msg

```
int msgget(key_t key, int fl);
```

Tworzy lub udostępnia istniejącą kolejkę komunikatów.

- Zwraca lokalny identyfikator kolejki komunikatów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowej kolejki (domyślnie: udostępnienie istniejącej)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/msg

```
int msgget(key_t key, int fl);
```

Tworzy lub udostępnia istniejącą kolejkę komunikatów.

- Zwraca lokalny identyfikator kolejki komunikatów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowej kolejki (domyślnie: udostępnienie istniejącej)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/msg

```
struct msgbuf {
   long mtype;
   char mtext[max_dl_komunikatu];
 };
```

Struktura msgbuf służy do przechowywania komunikatu

- mtype – typ komunikatu, liczba całkowita dodatnia
- mtext – treść komunikatu
- max_dl_komunikatu – dowolna, ale zdefiniowana z góry.

Programy wielowątkowe – IPC/msg

```
struct msgbuf {
   long mtype;
   char mtext[max_dl_komunikatu];
 };
```

Struktura msgbuf służy do przechowywania komunikatu

- mtype – typ komunikatu, liczba całkowita dodatnia
- mtext – treść komunikatu
- max_dl_komunikatu – dowolna, ale zdefiniowana z góry.

Programy wielowątkowe – IPC/msg

```
struct msgbuf {
   long mtype;
   char mtext[max_dl_komunikatu];
 };
```

Struktura msgbuf służy do przechowywania komunikatu

- mtype – typ komunikatu, liczba całkowita dodatnia
- mtext – treść komunikatu
- max_dl_komunikatu – dowolna, ale zdefiniowana z góry.

Programy wielowątkowe – IPC/msg

```
int msgsnd(int id, msgbuf *m,
      size_t s, int fl);
```

Funkcja msgsnd wysyła komunikat.

- id – lokalny identyfikator kolejki
- m – komunikat
- s – wielkość pola mtext w zmiennej m

Programy wielowątkowe – IPC/msg

```
int msgsnd(int id, msgbuf *m,
      size_t s, int fl);
```

Funkcja msgsnd wysyła komunikat.

- id – lokalny identyfikator kolejki
- m – komunikat
- s – wielkość pola mtext w zmiennej m

Programy wielowątkowe – IPC/msg

```
int msgsnd(int id, msgbuf *m,
      size_t s, int fl);
```

Funkcja msgsnd wysyła komunikat.

- id – lokalny identyfikator kolejki
- m – komunikat
- s – wielkość pola mtext w zmiennej m

Programy wielowątkowe – IPC/msg

```
int msgsnd(int id, msgbuf *m,
      size_t s, int fl);
```

Maski bitowe parametru fl:

- IPC_NOWAIT – w przypadku braku wolnego miejsca w kolejce natychmiastowy powrót z errno==EAGAIN
- zachowanie domyślne – zawieszenie wykonywania wątku do czasu aż w kolejce zwolni się miejsce dla komunikatu

Programy wielowątkowe – IPC/msg

```
int msgsnd(int id, msgbuf *m,
      size_t s, int fl);
```

Maski bitowe parametru fl:

- IPC_NOWAIT – w przypadku braku wolnego miejsca w kolejce natychmiastowy powrót z errno==EAGAIN
- zachowanie domyślne – zawieszenie wykonywania wątku do czasu aż w kolejce zwolni się miejsce dla komunikatu

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Funkcja msgrcv odbiera komunikat.

- id – lokalny identyfikator kolejki
- m – bufor w którym zostanie zapisany komunikat
- s – wielkość pola mtext w zmiennej m

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Funkcja msgrcv odbiera komunikat.

- id – lokalny identyfikator kolejki
- m – bufor w którym zostanie zapisany komunikat
- s – wielkość pola mtext w zmiennej m

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Funkcja msgrcv odbiera komunikat.

- id – lokalny identyfikator kolejki
- m – bufor w którym zostanie zapisany komunikat
- s – wielkość pola mtext w zmiennej m

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Argument typ – typ oczekiwanego komunikatu

- 0 – pierwszy komunikat w kolejce
- x>0 – pierwszy komunikat typu x
- x>0 i ustawiony bit MSG_EXCEPT w argumencie fl – pierwszy komunikat typu różnego od x.
- x<0 – pierwszy komunikat typu |x| lub niższego

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Argument typ – typ oczekiwanego komunikatu

- 0 – pierwszy komunikat w kolejce
- x>0 – pierwszy komunikat typu x
- x>0 i ustawiony bit MSG_EXCEPT w argumencie fl – pierwszy komunikat typu różnego od x.
- x<0 – pierwszy komunikat typu |x| lub niższego

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Argument typ – typ oczekiwanego komunikatu

- 0 – pierwszy komunikat w kolejce
- x>0 – pierwszy komunikat typu x
- x>0 i ustawiony bit MSG_EXCEPT w argumencie fl – pierwszy komunikat typu różnego od x.
- x<0 – pierwszy komunikat typu |x| lub niższego

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Argument typ – typ oczekiwanego komunikatu

- 0 – pierwszy komunikat w kolejce
- x>0 – pierwszy komunikat typu x
- x>0 i ustawiony bit MSG_EXCEPT w argumencie fl – pierwszy komunikat typu różnego od x.
- x<0 – pierwszy komunikat typu |x| lub niższego

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Maski bitowe parametru fl:

- IPC_NOWAIT – w przypadku braku odpowiedniego komunikatu natychmiastowy powrót z errno==ENOMSG
- zachowanie domyślne – zawieszenie wykonywania wątku do czasu nadejścia odpowiedniego komunikatu (lub usunięcia kolejki z systemu).

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Maski bitowe parametru fl:

- IPC_NOWAIT – w przypadku braku odpowiedniego komunikatu natychmiastowy powrót z errno==ENOMSG
- zachowanie domyślne – zawieszenie wykonywania wątku do czasu nadejścia odpowiedniego komunikatu (lub usunięcia kolejki z systemu).

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Maski bitowe parametru fl:

- MSG_NOERROR – w przypadku nadejścia zbyt dużego komunikatu obcięcie go do s bajtów.
- zachowanie domyślne – usunięcie zbyt dużego komunikatu z kolejki i powrót z errno==E2BIG

Programy wielowątkowe – IPC/msg

```
int msgrcv(int id, msgbuf *m,
      size_t s, int typ, int fl);
```

Maski bitowe parametru fl:

- MSG_NOERROR – w przypadku nadejścia zbyt dużego komunikatu obcięcie go do s bajtów.
- zachowanie domyślne – usunięcie zbyt dużego komunikatu z kolejki i powrót z errno==E2BIG

Programy wielowątkowe – IPC/msg

int msgctl(int id, int cmd, msqid_ds *b

msgctl – inne operacje na kolejce komunikatów

- id – lokalny identyfikator kolejki
- b – dane dla IPC_SET i IPC_STAT

Programy wielowątkowe – IPC/msg

int msgctl(int id, int cmd, msqid_ds *b

msgctl – inne operacje na kolejce komunikatów

- id – lokalny identyfikator kolejki
- b – dane dla IPC_SET i IPC_STAT

Programy wielowątkowe – IPC/msg

```
int msgctl(int id, int cmd, msqid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o kolejce (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów kolejki (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie kolejki z systemu

Programy wielowątkowe – IPC/msg

```
int msgctl(int id, int cmd, msqid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o kolejce (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów kolejki (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie kolejki z systemu

Programy wielowątkowe – IPC/msg

```
int msgctl(int id, int cmd, msqid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o kolejce (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów kolejki (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie kolejki z systemu

Programy wielowątkowe – IPC/shm

Współdzielone segmenty pamięci:

- Umożliwiają równoczesny dostęp wielu aplikacji do wspólnych obszarów pamięci
- Z punktu widzenia systemu operacyjnego są plikami
- Z punktu widzenia aplikacji nie różnią się od normalnych bloków przydzialanych przez malloc

Programy wielowątkowe – IPC/shm

Współdzielone segmenty pamięci:

- Umożliwiają równoczesny dostęp wielu aplikacji do wspólnych obszarów pamięci
- Z punktu widzenia systemu operacyjnego są plikami
- Z punktu widzenia aplikacji nie różnią się od normalnych bloków przydzialanych przez malloc

Programy wielowątkowe – IPC/shm

Współdzielone segmenty pamięci:

- Umożliwiają równoczesny dostęp wielu aplikacji do wspólnych obszarów pamięci
- Z punktu widzenia systemu operacyjnego są plikami
- Z punktu widzenia aplikacji nie różnią się od normalnych bloków przydzialanych przez malloc

Programy wielowątkowe – IPC/shm

```
int shmget(key_t key, size_t s, int fl);
```

Tworzy lub udostępnia istniejący wspólny blok pamięci

- Zwraca lokalny identyfikator wspólnego bloku pamięci o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego bloku o wielkości s bajtów, zaokrąglonej w górę do pełnej strony (domyślnie: udostępnienie istniejącego bliku pamięci)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/shm

```
int shmget(key_t key, size_t s, int fl);
```

Tworzy lub udostępnia istniejący wspólny blok pamięci

- Zwraca lokalny identyfikator wspólnego bloku pamięci o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego bloku o wielkości s bajtów, zaokrąglonej w górę do pełnej strony (domyślnie: udostępnienie istniejącego bliku pamięci)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/shm

```
int shmget(key_t key, size_t s, int fl);
```

Tworzy lub udostępnia istniejący wspólny blok pamięci

- Zwraca lokalny identyfikator wspólnego bloku pamięci o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego bloku o wielkości s bajtów, zaokrąglonej w górę do pełnej strony (domyślnie: udostępnienie istniejącego bliku pamięci)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/shm

```
int shmget(key_t key, size_t s, int fl);
```

Tworzy lub udostępnia istniejący wspólny blok pamięci

- Zwraca lokalny identyfikator wspólnego bloku pamięci o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego bloku o wielkości s bajtów, zaokrąglonej w górę do pełnej strony (domyślnie: udostępnienie istniejącego bliku pamięci)
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/shm

```
void * shmat(int id, void * addr, int fl);
```

Funkcja shmat:

- Dołącza współdzielony blok pamięci do przestrzeni adresowej procesu.
- Nie rozszerza lokalnej przestrzeni adresowej (nie wywołuje brk)!
- Zwraca wskaźnik na wspólny blok pamięci analogiczny do wskaźnika zwracanego przez malloc

Programy wielowątkowe – IPC/shm

```
void * shmat(int id, void * addr, int fl);
```

Funkcja shmat:

- Dołącza współdzielony blok pamięci do przestrzeni adresowej procesu.
- Nie rozszerza lokalnej przestrzeni adresowej (nie wywołuje brk)!
- Zwraca wskaźnik na wspólny blok pamięci analogiczny do wskaźnika zwracanego przez malloc

Programy wielowątkowe – IPC/shm

```
void * shmat(int id, void * addr, int fl);
```

Funkcja shmat:

- Dołącza współdzielony blok pamięci do przestrzeni adresowej procesu.
- Nie rozszerza lokalnej przestrzeni adresowej (nie wywołuje brk)!

- Zwraca wskaźnik na wspólny blok pamięci analogiczny do wskaźnika zwracanego przez malloc

Programy wielowątkowe – IPC/shm

```
void * shmat(int id, void * addr, int fl);
```

Funkcja shmat:

- id – lokalny identyfikator wspólnego bloku pamięci
- Jeżeli addr==NULL blok jest mapowany pod pierwszy wolny adres (decyduje system).
- Jeżeli addr!=NULL blok jest mapowany pod adres addr. Adres musi być wolny i mieć odpowiedni alingment.

Programy wielowątkowe – IPC/shm

```
void * shmat(int id, void * addr, int fl);
```

Funkcja shmat:

- id – lokalny identyfikator wspólnego bloku pamięci
- Jeżeli addr==NULL blok jest mapowany pod pierwszy wolny adres (decyduje system).
- Jeżeli addr!=NULL blok jest mapowany pod adres addr. Adres musi być wolny i mieć odpowiedni alingment.

Programy wielowątkowe – IPC/shm

```
void * shmat(int id, void * addr, int fl);
```

Funkcja shmat:

- id – lokalny identyfikator wspólnego bloku pamięci
- Jeżeli addr==NULL blok jest mapowany pod pierwszy wolny adres (decyduje system).
- Jeżeli addr!=NULL blok jest mapowany pod adres addr. Adres musi być wolny i mieć odpowiedni alingment.

Programy wielowątkowe – IPC/shm

void * shmat(int id, void * addr, int fl);

Maski bitowe argumentu fl

- SHM_RDONLY – blok pamięci tylko do odczytu
- zachowanie domyślne – blok pamięci do odczytu i zapisu

Programy wielowątkowe – IPC/shm

```
void * shmat(int id, void * addr, int fl);
```

Maski bitowe argumentu fl

- SHM_RDONLY – blok pamięci tylko do odczytu
- zachowanie domyślne – blok pamięci do odczytu i zapisu

Programy wielowątkowe – IPC/shm

```
int shmdt(void * addr);
```

Funkcja shmdt

- Usuwa współdzielony blok pamięci z przestrzeni adresowej procesu.
- Nie kasuje samego bloku pamięci!
- Odpowiednik funkcji free

Programy wielowątkowe – IPC/shm

```
int shmdt(void * addr);
```

Funkcja shmdt

- Usuwa współdzielony blok pamięci z przestrzeni adresowej procesu.
- Nie kasuje samego bloku pamięci!
- Odpowiednik funkcji free

Programy wielowątkowe – IPC/shm

int shmdt(void * addr);

Funkcja shmdt

- Usuwa współdzielony blok pamięci z przestrzeni adresowej procesu.
- Nie kasuje samego bloku pamięci!
- Odpowiednik funkcji free

Programy wielowątkowe – IPC/shm

```
int shmctl(int id, int cmd, shmid_ds *b);
```

shmctl – inne operacje na współdzielonym bloku pamięci

- id – lokalny identyfikator bloku
- b – dane dla IPC_SET i IPC_STAT

Programy wielowątkowe – IPC/shm

int shmctl(int id, int cmd, shmid_ds *b);

shmctl – inne operacje na współdzielonym bloku pamięci

- id – lokalny identyfikator bloku
- b – dane dla IPC_SET i IPC_STAT

Programy wielowątkowe – IPC/shm

```
int shmctl(int id, int cmd, shmid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o bloku (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów bloku (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie bloku z systemu

Programy wielowątkowe – IPC/shm

```
int shmctl(int id, int cmd, shmid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o bloku (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów bloku (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie bloku z systemu

Programy wielowątkowe – IPC/shm

```
int shmctl(int id, int cmd, shmid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o bloku (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów bloku (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie bloku z systemu

Programy wielowątkowe – IPC/sem

Semafory:

- Umożliwiają synchronizację pomiędzy procesam.
- Zabezpieczają współdzielone zasoby przed jednoczesnym dostępem
- Operacje na semaforach są atomowe.

Programy wielowątkowe – IPC/sem

Semafory:

- Umożliwiają synchronizację pomiędzy procesam.
- Zabezpieczają współdzielone zasoby przed jednoczesnym dostępem
- Operacje na semaforach są atomowe.

Programy wielowątkowe – IPC/sem

Semafory:

- Umożliwiają synchronizację pomiędzy procesam.
- Zabezpieczają współdzielone zasoby przed jednoczesnym dostępem
- Operacje na semaforach są atomowe.

Programy wielowątkowe – IPC/sem

int semget(key_t key, int n, int fl);

Tworzy lub udostępnia istniejący zbiór semaforów

- Zwraca lokalny identyfikator zbioru semaforów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego zbioru n semaforów.
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/sem

int semget(key_t key, int n, int fl);

Tworzy lub udostępnia istniejący zbiór semaforów

- Zwraca lokalny identyfikator zbioru semaforów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego zbioru n semaforów.
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/sem

int semget(key_t key, int n, int fl);

Tworzy lub udostępnia istniejący zbiór semaforów

- Zwraca lokalny identyfikator zbioru semaforów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego zbioru n semaforów.
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/sem

int semget(key_t key, int n, int fl);

Tworzy lub udostępnia istniejący zbiór semaforów

- Zwraca lokalny identyfikator zbioru semaforów o globalnym kluczu key.
- Maski bitowe parametru fl:
  - IPC_CREAT – utworzenie nowego zbioru n semaforów.
  - można podać maskę praw dostępu w postaci liczby ósemkowej (np. 0660).

Programy wielowątkowe – IPC/sem

```
int semop(int id, sembuf *op, int n);
```

Funkcja semop wykonuje **atomowo** zbiór operacji na semaforach.

- id – lokalny identyfikator zbioru semaforów
- op – tablica operacji
- n – ilość operacji (długość tablicy op.

Programy wielowątkowe – IPC/sem

```
int semop(int id, sembuf *op, int n);
```

Funkcja semop wykonuje **atomowo** zbiór operacji na semaforach.

- id – lokalny identyfikator zbioru semaforów
- op – tablica operacji
- n – ilość operacji (długość tablicy op.

Programy wielowątkowe – IPC/sem

```
int semop(int id, sembuf *op, int n);
```

Funkcja semop wykonuje **atomowo** zbiór operacji na semaforach.

- id – lokalny identyfikator zbioru semaforów
- op – tablica operacji
- n – ilość operacji (długość tablicy op.

Programy wielowątkowe – IPC/sem

```
struct sembuf {
   unsigned short sem_num;
   short sem_op;
   short sem_flg;
};
```

Struktura sembuf definiuje operację na semaforze

- sem_num – numer semafora ze zbioru

Programy wielowątkowe – IPC/sem

```
struct sembuf {
   unsigned short sem_num;
   short sem_op;
   short sem_flg;
};
```

Pole sem_op – typ operacji na semaforze

- sem_op==0 – „czekanie na zero”. Wykonywanie wątku zostaje zawieszone do czasu aż semafor będzie miał wartość 0.
- sem_op>0 – „zamknięcie semafora”. Wartość sem_op zostaje dodana do bieżącej wartości semafora.
- sem_op<0 – „otwarcie semafora”. Wartość |sem_op| zostaje odjęta do bieżącej wartości semafora.

Programy wielowątkowe – IPC/sem

```
struct sembuf {
   unsigned short sem_num;
   short sem_op;
   short sem_flg;
};
```

Pole sem_op – typ operacji na semaforze

- sem_op==0 – „czekanie na zero”. Wykonywanie wątku zostaje zawieszone do czasu aż semafor będzie miał wartość 0.
- sem_op>0 – „zamknięcie semafora”. Wartość sem_op zostaje dodana do bieżącej wartości semafora.
- sem_op<0 – „otwarcie semafora”. Wartość |sem_op| zostaje odjęta do bieżącej wartości semafora.

Programy wielowątkowe – IPC/sem

```
struct sembuf {
   unsigned short sem_num;
   short sem_op;
   short sem_flg;
};
```

Pole sem_op – typ operacji na semaforze

- sem_op==0 – „czekanie na zero”. Wykonywanie wątku zostaje zawieszone do czasu aż semafor będzie miał wartość 0.
- sem_op>0 – „zamknięcie semafora”. Wartość sem_op zostaje dodana do bieżącej wartości semafora.
- sem_op<0 – „otwarcie semafora”. Wartość |sem_op| zostaje odjęta do bieżącej wartości semafora.

Programy wielowątkowe – IPC/sem

```
struct sembuf {
   unsigned short sem_num;
   short sem_op;
   short sem_flg;
};
```

Maski bitowe pola sem_fl:

- IPC_NOWAIT – wykonywanie wątku nigdy nie zostanie zawieszone. W przypadku gdy wątek normalnie zostałby zawieszony zwrócenie błedu errno==EAGAIN
- SEM_UNDO – operacja zostanie automatycznie cofnięta w przypadku zakończenia wykonywania wątku (z jakiejkolwiek przyczyny).

Programy wielowątkowe – IPC/sem

```
struct sembuf {
   unsigned short sem_num;
   short sem_op;
   short sem_flg;
};
```

Maski bitowe pola sem_fl:

- IPC_NOWAIT – wykonywanie wątku nigdy nie zostanie zawieszone. W przypadku gdy wątek normalnie zostałby zawieszony zwrócenie błedu errno==EAGAIN
- SEM_UNDO – operacja zostanie automatycznie cofnięta w przypadku zakończenia wykonywania wątku (z jakiejkolwiek przyczyny).

Programy wielowątkowe – IPC/sem

int semctl(int id, int cmd, shmid_ds *b);

shmctl – inne operacje na zbiorze semaforów

- id – lokalny identyfikator zbioru semaforów
- b – dane dla IPC_SET i IPC_STAT

Programy wielowątkowe – IPC/sem

int semctl(int id, int cmd, shmid_ds *b);

shmctl – inne operacje na zbiorze semaforów

- id – lokalny identyfikator zbioru semaforów
- b – dane dla IPC_SET i IPC_STAT

Programy wielowątkowe – IPC/sem

```
int semctl(int id, int cmd, shmid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o zbiorze semaforów (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów zbioru semaforów (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie zbioru semaforów z systemu

Programy wielowątkowe – IPC/sem

int semctl(int id, int cmd, shmid_ds *b);

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o zbiorze semaforów (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów zbioru semaforów (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie zbioru semaforów z systemu

Programy wielowątkowe – IPC/sem

```
int semctl(int id, int cmd, shmid_ds *b);
```

Polecenia w argumencie cmd

- IPC_STAT – odczytanie informacji o zbiorze semaforów (m.in. UID, GID, czas ostatniej zmiany, itp.)
- IPC_SET – zmiana parametrów zbioru semaforów (m.in. UID, GID, prawa dostępu, wielkość, itp.)
- IPC_RMID – usunięcie zbioru semaforów z systemu