Systemy operacyjne

Laboratorium 7

Mateusz Małek
28 kwietnia 2017

Laboratorium 7

Semafory i pamięć wspólna (System V i POSIX)

Semafory

Tworzenie zbioru semaforów (System V)

- int semget(key_t key, int nsems, int semflag) - man 2 semget
  - key to wartość zwrócona z ftok lub stała IPC_PRIVATE
  - W nsems podajemy ilość semaforów która ma być dostępna w ramach tworzonego zbioru (semafory w zbiorze są numerowane od 0)
  - W semflag możemy podać uprawnienia do zbioru semaforów (jak w open) - jeśli jeszcze nie istniał i zostanie w tym momencie utworzony, to zostaną mu ustawione właśnie takie uprawnienia
  - Dostępne są też flagi IPC_CREAT i IPC_EXCL, analogiczne do flag O_CREAT i O_EXCL
  - Funkcja zwraca liczbowy identyfikator, jednoznacznie identyfikujący dany zbiór semaforów w całym systemie
    - Jeśli użyliśmy IPC_PRIVATE, to wartość zwróconą przez semget zwykle w jakiś sposób przekazujemy innemu procesowi (np. kolejka komunikatów, pamięć wspólna, plik na dysku…)

Operacje na zbiorze semaforów (System V)

- int semop(int semid, struct sembuf *sops, size_t nsops) - man 2 semop
  - semid to wartość zwrócona z semget
  - sops wskazuje na tablicę składającą się z nsops struktur sembuf, opisujących operacje do wykonania na zbiorze semaforów:

```
struct sembuf {
    unsigned short sem_num;  /* semaphore number */
    short          sem_op;   /* semaphore operation */
    short          sem_flg;  /* operation flags */
}
```

    - sem_num - numer semafora (w zbiorze) na którym chcemy wykonać operację
    - sem_op - zmniejszenie (sem_op < 0), zwiększenie (sem_op > 0) lub oczekiwanie na zerową wartość semafora (sem_op == 0)
    - sem_flg - IPC_NOWAIT (zwraca błąd całego semop, jeśli danej operacji nie da się wykonać natychmiast) i/lub SEM_UNDO (po zakończeniu procesu cofnie tą operację)
  - Operacje podane w tablicy zostaną wykonane atomowo - albo wszystkie, albo żadna (funkcja będzie oczekiwała aż ich wykonanie będzie możliwe lub natychmiast zwróci błąd i nie wykona żadnej z nich)

Operacje na zbiorze semaforów cd. (System V)

- int semtimedop(int semid, struct sembuf *sops, size_t nsops, const struct timespec *timeout) - man 2 semtimedop
  - Sposób użycia identyczny jak w przypadku semop
  - Możemy dodatkowo określić timeout dla operacji blokujących - jeśli zostanie przekroczony, żadna z żądanych operacji nie zostanie wykonana

Zarządzanie zbiorem semaforów (System V)

- int semctl(int semid, int semnum, int cmd[, union semun arg]) - man 2 semctl
  - semid to wartość zwrócona z semget
  - semnum to numer semafora (w zbiorze) na którym chcemy wykonać operację (ten argument jest ignorowany jeśli podane cmd nie dotyczy pojedynczego semafora)
  - cmd to pożądana akcja:
    - **IPC_RMID - natychmiastowe usunięcie zbioru semaforów i wybudzenie wszystkich czekających na nim procesów z errno EIDRM**
    - GETVAL - zwraca jako wynik semctl aktualną wartość semafora numer semnum
    - SETVAL - ustawia arg.val jako aktualną wartość semafora numer semnum
    - … i jeszcze kilka innych
  - Argument arg to unia wykorzystywana przy niektórych akcjach (nie trzeba go podawać, jeśli nie jest potrzebny):

```
union semun {
    int   val;               /* Value for SETVAL */
    struct semid_ds *buf;    /* Buffer for IPC_STAT, IPC_SET */
    unsigned short  *array;  /* Array for GETALL, SETALL */
    struct seminfo  *__buf;  /* Buffer for IPC_INFO
                                (Linux-specific) */
};
```

Semafory (POSIX)

Do korzystania z POSIXowych semaforów konieczne jest zlinkowanie programu z biblioteką pthread.

Mówiąc prościej - w pliku Makefile umieszczamy:

```
LDLIBS = -lpthread
```

Tworzenie semafora (POSIX)

- sem_t sem_open(const char *name, int oflag)
- sem_t sem_open(const char *name, int oflag, mode_t mode, unsigned int value)
  - Jako name podajemy ścieżkę zgodną z omówionymi wcześniej wymaganiami
  - W przypadku tworzenia semafora można (i należy) podać flagę O_CREAT, dostępna jest też flaga O_EXCL
  - W mode podajemy uprawnienia do semafora (np. 0644)
  - W value podajemy początkową wartość semafora (jeśli jest w tym momencie tworzony)
- Oba warianty opisane w man 3 sem_open

Operacje na semaforze (POSIX)

- int sem_post(sem_t *sem) - man 3 sem_post
  - Podnosi wskazany przez sem semafor (inkrementuje jego wartość o 1)
- int sem_wait(sem_t *sem)
  - Opuszcza wskazany przez sem semafor (dekrementuje jego wartość o 1), chyba że wartość semafora wynosiła zero - wówczas zasypia
- int sem_trywait(sem_t *sem)
  - Opuszcza wskazany przez sem semafor (dekrementuje jego wartość o 1), chyba że wartość semafora wynosiła zero - wówczas natychmiast zwraca błąd
- int sem_timedwait(sem_t *sem, const struct timespec *abs_timeout)
  - Opuszcza wskazany przez sem semafor (dekrementuje jego wartość o 1), chyba że wartość semafora wynosiła zero - wówczas zasypia na czas nie dłuższy niż abs_timeout, po którym zwraca błąd
- Powyższe warianty sem_wait opisane są w man 3 sem_wait

Operacje na semaforze cd. (POSIX)

- int sem_getvalue(sem_t *sem, int *sval) - man 3 sem_getvalue
  - Umieszcza bieżącą wartość semafora wskazywanego przez sem w zmiennej wskazywanej przez sval
  - W przypadku kiedy na danym semaforze oczekuje jeden lub więcej procesów/wątków, zwracana wartość nie jest sprecyzowana - standard POSIX dopuszcza dwa warianty:
    - Zwrócenie zera (to zachowanie stosowane jest w Linuksie)
    - Zwrócenie wartości ujemnej, odpowiadającej ilości procesów/wątków oczekujących na danym semaforze

Zamykanie i usuwanie semaforów (POSIX)

- int sem_close(sem_t *sem) - man 3 sem_close
  - Zwalnia zasoby zaalokowane w danym procesie przez sem_open dla semafora wskazywanego przez sem
  - Nieobowiązkowe (wywoływane automatycznie przy zakończeniu procesu lub wywołaniu execve)
- int sem_unlink(const char *name) - man 3 sem_unlink
  - Oznacza nazwany semafor do usunięcia
  - Faktyczne usunięcie semafora następuje w momencie kiedy zostanie zamknięty przez wszystkie korzystające z niego procesy

Pamięć wspólna

Tworzenie pamięci wspólnej (System V)

- int shmget(key_t key, size_t size, int shmflg) - man 2 shmget
  - key to wartość zwrócona z ftok lub stała IPC_PRIVATE
  - size to rozmiar segmentu pamięci wspólnej do utworzenia
  - W shmflag możemy podać uprawnienia do segmentu pamięci wspólnej (jak w open) - jeśli jeszcze nie istniał i zostanie w tym momencie utworzony, to zostaną mu ustawione właśnie takie uprawnienia
  - Dostępne są też flagi IPC_CREAT i IPC_EXCL, analogiczne do flag O_CREAT i O_EXCL
  - Funkcja zwraca liczbowy identyfikator, jednoznacznie identyfikujący segment pamięci wspólnej w całym systemie
  - Jeśli użyliśmy IPC_PRIVATE, to wartość zwróconą przez shmget zwykle w jakiś sposób przekazujemy innemu procesowi (np. kolejka komunikatów, segment pamięci wspólnej utworzony w oparciu o z góry ustaloną ścieżkę i klucz, plik na dysku…)

Podłączanie/odłączanie pamięci wspólnej (System V)

- void *shmat(int shmid, const void *shmaddr, int shmflg)
  - shmid to wartość zwrócona z shmget
  - shmaddr wskazuje w jakim miejscu przestrzeni adresowej procesu chcemy dołączyć segment pamięci wspólnej (typowo podajemy tutaj NULL, co oznacza że odpowiednie miejsce ma wybrać system operacyjny)
  - shmflg może zawierać np. SHM_RDONLY (dołącz tylko do odczytu) lub SHM_EXEC (pozwala na umieszczenie w pamięci kodu wykonywalnego)
  - Funkcja zwraca wskaźnik na dołączony segment pamięci wspólnej (shmaddr z ewentualnym “zaokrągleniem” do wielokrotności SHMLBA lub adres wybrany przez system)
- int shmdt(const void *shmaddr)
  - Odłącza od przestrzeni adresowej procesu segment pamięci wspólnej, który został uprzednio dołączony pod adresem shmaddr (czy to w wyniku wskazania go jako argumentu do shmat, czy też w wyniku automatycznego przypisania go w wywołaniu shmat)
- Obie funkcje opisane w man 2 shmop

Zarządzanie pamięcią wspólną (System V)

- int shmctl(int shmid, int cmd, struct shmid_ds *buf) - man 2 shmctl
  - shmid to wartość zwrócona z shmget
  - cmd to pożądana akcja
    - **IPC_RMID - oznaczenie segmentu pamięci wspólnej do usunięcia w odłączenia się od niego ostatniego procesu - inaczej niż przy kolejkach i semaforach! (argument buf jest ignorowany, można podać np. NULL)**
    - IPC_STAT - wypełnia strukturę shmid_ds wskazywaną przez buf aktualnymi informacjami na temat segmentu pamięci wspólnej
    - IPC_SET - zmienia właściciela i uprawnienia zgodnie z wartościami ustawionymi w strukturze shmid_ds wskazywanej przez buf

Struktury shmid_ds i ipc_perm (System V)

```
struct shmid_ds {
    struct ipc_perm shm_perm;    /* Ownership and permissions */
    size_t          shm_segsz;   /* Size of segment (bytes) */
    time_t          shm_atime;   /* Last attach time */
    time_t          shm_dtime;   /* Last detach time */
    time_t          shm_ctime;   /* Last change time */
    pid_t           shm_cpid;    /* PID of creator */
    pid_t           shm_lpid;    /* PID of last shmat(2)/shmdt(2) */
    shmatt_t        shm_nattch;  /* No. of current attaches */
    ...
};
```

Pogrubieniem zaznaczono wartości, które można modyfikować z wykorzystaniem komendy IPC_SET

```
struct ipc_perm {
    key_t          __key;    /* Key supplied to shmget(2) */
    uid_t          uid;      /* Effective UID of owner */
    gid_t          gid;      /* Effective GID of owner */
    uid_t          cuid;     /* Effective UID of creator */
    gid_t          cgid;     /* Effective GID of creator */
    unsigned short mode;     /* Permissions + SHM_DEST and
                                SHM_LOCKED flags */
    unsigned short __seq;    /* Sequence number */
};
```

Pamięć wspólna (POSIX)

Do korzystania z POSIXowej pamięci wspólnej konieczne jest zlinkowanie programu z biblioteką rt.

Mówiąc prościej - w pliku Makefile umieszczamy:

```
LDLIBS = -lrt
```

Tworzenie pamięci wspólnej (POSIX)

- int shm_open(const char *name, int oflag, mode_t mode) - man 3 shm_open
  - Jako name podajemy ścieżkę zgodną z omówionymi wcześniej wymaganiami
  - W oflag podajemy tryb pracy z segmentem pamięci wspólnej (O_RDONLY/O_WRONLY/O_RDWR), w przypadku tworzenia nowego segmentu pamięci wspólnej można (i należy) podać flagę O_CREAT, dostępna są też flaga O_EXCL, O_NONBLOCK lub O_TRUNC
  - W mode podajemy uprawnienia do tworzonego segmentu pamięci (np. 0644)
  - Zostaje zwrócony numer deskryptora pliku reprezentującego segment pamięci
- int ftruncate(int fd, off_t length) - man 2 ftruncate
  - Ponieważ segment pamięci wspólnej jest reprezentowany przez deskryptor pliku, to do kontrolowania wielkości takiego segmentu możemy wykorzystać znaną nam już funkcję ftruncate
  - Jako fd podajemy numer zwrócony przez shm_open, a jako length - żądaną wielkość segmentu pamięci wspólnej

Podłączanie/odłączanie pamięci wspólnej (POSIX)

- void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset)
  - addr określa gdzie w przestrzeni adresowej procesu podłączyć segment pamięci wspólnej (typowo podajemy NULL, aby odpowiednie miejsce wybrał system operacyjny)
  - prot to maska bitowa określająca “zabezpieczenia” segmentu pamięci:
    PROT_READ (możliwość odczytu), PROT_WRITE (możliwość zapisu), PROT_EXEC (możliwość uruchamiania kodu umieszczonego w pamięci);
    PROT_NONE to brak uprawnień do korzystania ze zmapowanej pamięci w jakikolwiek sposób
  - Jako flags w przypadku pamięci wspólnej podajemy typowo MAP_SHARED
  - Jako fd podajemy numer deskryptora zwrócony przez shm_open
  - Argumenty length i offset pozwalają nam podłączyć fragment pamięci wspólnej; typowo jako length podajemy wielkość całego segmentu, a jako offset podajemy zero
- int munmap(void *addr, size_t length)
  - Odłącza wskazany obszar pamięci (fragment lub całość) od przestrzeni adresowej procesu
- Obie funkcje opisane w man 2 mmap

Zamykanie i usuwanie pamięci wspólnej (POSIX)

- int close(int fd) - man 2 close
  - Jako fd podajemy numer deskryptora zwrócony przez shm_open
  - Zwalnia zasoby zaalokowane przez shm_open w danym procesie
  - Nieobowiązkowe (co więcej, nie powoduje automatycznego wywołania munmap dla segmentów które podłączono do przestrzeni adresowej procesu - to następuje dopiero przy zakończeniu procesu!)
- int shm_unlink(const char *name) - man 3 shm_unlink
  - Oznacza segment pamięci wspólnej związany z daną nazwą do usunięcia
  - Zwolnienie segmentu pamięci oznaczonego do usunięcia przy użyciu tej funkcji następuje dopiero kiedy ostatni proces odmapuje pamięć wspólną ze swojej przestrzeni adresowej
  - Po wywołaniu shm_unlink wywołania shm_open dla tej nazwy będą kończyły się błędem (chyba że użyto flagi O_CREAT)

Dziękuję za uwagę