

Maryna Lukachyk (308294)

Laboratorium 8

Zaawansowana komunikacja międzyprocesowa - semafony i pamięć wspólna

Semafony

- Wykorzystanie semaforów zapobiega niedozwolonemu wykonaniu operacji na określonych danych jednocześnie przez większą liczbę procesów
- Przez odpowiednie wykorzystywanie semaforów można zapobiec sytuacji w której wystąpi zakleszczenie lub zagłócenie.
- Operacje wykonywane na semaforze są atomowe
- Semafor można traktować jako licznik, który jest zmniejszany o 1 gdy jest zajmowany i zwiększany o 1 gdy jest zwalniany
- Semafor trzeba zainicjować wywołując operację podniesienia
- Struktura semafora wygląda następująco:

```
struct sem{
    ushort semval    // wartość semafora
    pid_t  sempid    // identyfikator procesu ostatnio wykonującego operację na semaforze
    ushort semncnt   // liczba procesów, które czekają aż wartość semafora będzie zwiększona
    ushort semzcnt   // liczba procesów które czekają aż semafor osiągnie wartość 0
}
```

- Niektóre funkcje:

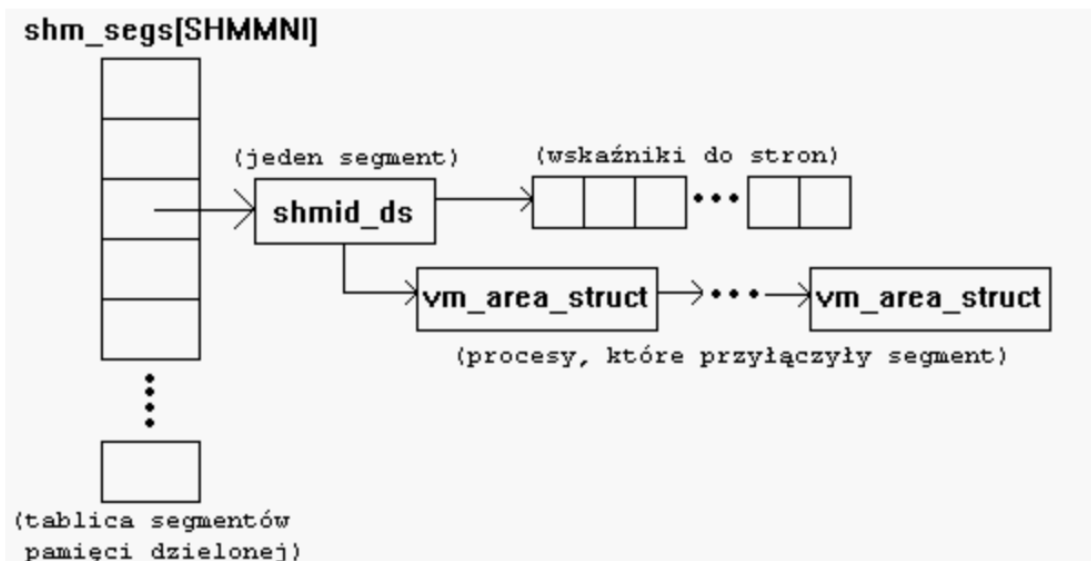
int semget (key_t key, int nsems, int semflgs) - tworzenie semafora

int semctl (int semid, int semnum, int cmd, union semun ctl_arg) - funkcje kontrolujące pojedynczy semafor lub ich zestaw

int semop (int semid, struct sembuf *sops, unsigned nsops) - zajmowanie i zwalnianie semafora

Pamięć współdzielona

- najszybsza metoda komunikacji międzyprocesowej.
- Komunikacja w systemie pamięci wspólnej jest symetryczna i dwukierunkowa.



- **shmget()** - służy do uzyskania identyfikatora segmentu używanego później przez pozostałe funkcje jako pierwszy parametr,
- **shmctl()** - umożliwia ustawianie i pobieranie wartości parametrów związanych z segmentami pamięci dzielonej oraz zaznaczanie segmentów do skasowania,
- **shmat()** - dołącza segment pamięci dzielonej do przestrzeni adresowej procesu,
- **shmdt()** - odłącza segment.

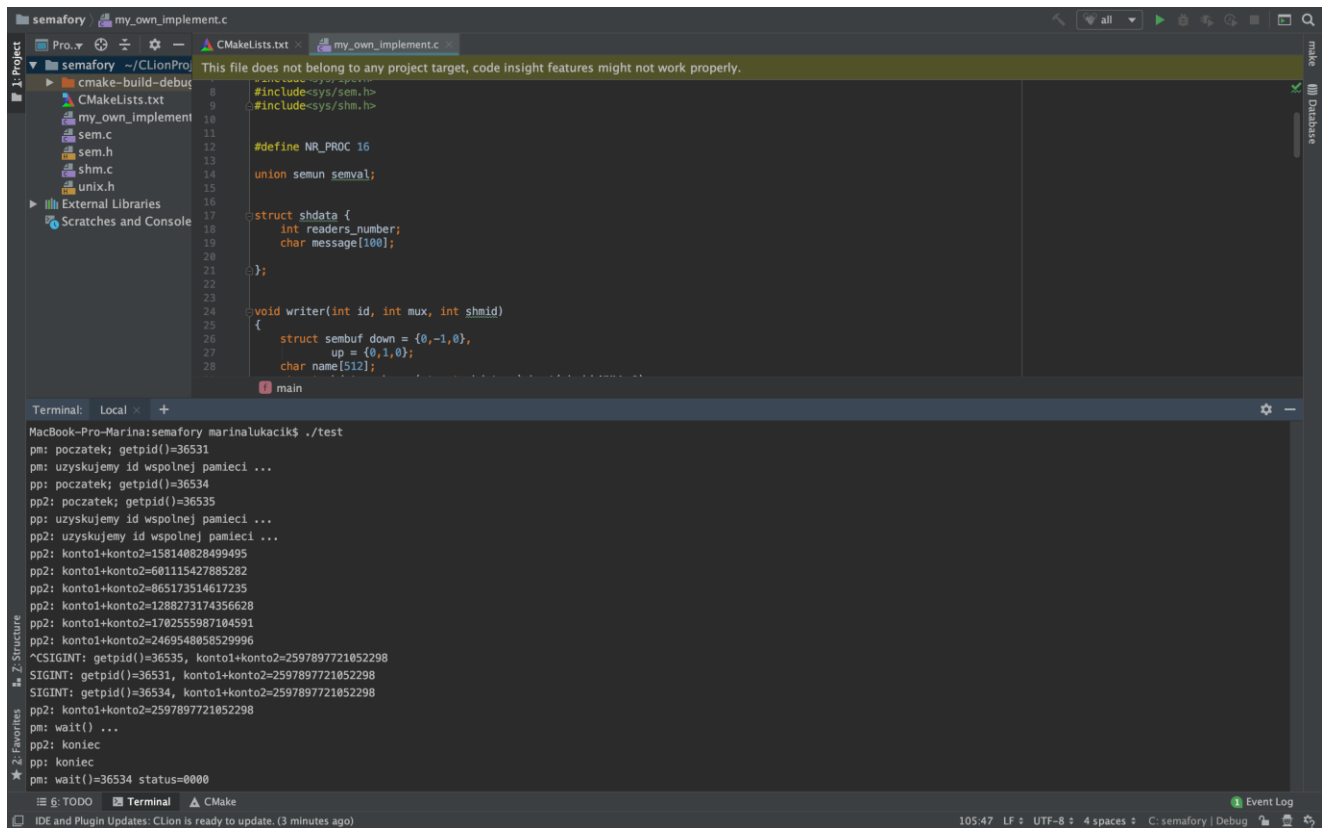
Problem czytelników i pisarzy: Dowolna liczba procesów czyta lub pisze do jednego pliku. Dowolne z nich mogą jednocześnie czytać. Jeśli jakiś pisze to inne nie piszą ani nie czytają. W rozwiązaniu nie wolno zagłodzić żadnego wątku.

Program rozwiązuje problem czytelników i pisarzy, gdzie czytelnicy są uprzywilejowani. Jeśli na zasobie współdzielonym jest wykonywana wyłącznie operacja odczytu, to jest to działanie bezpieczne, a więc pamięć dzieloną może współbieżnie odczytywać wiele procesów, ale zapisywać może w danym czasie tylko jeden proces i dodatkowo w czasie tej modyfikacji nie może być wykonywany żaden odczyt.

W funkcji `main()` programu tworzony jest prywatny obszar pamięci dzielonej oraz zbiór semaforów, które będą go chronić. Następnie program tworzy 16 procesów potomnych, przy czym losowo wybiera rolę dla każdego z nich - czytelnik lub pisarz. Proces macierzysty czeka na ich zakończenie, a następnie usuwa wszystkie wykorzystywane przez siebie zasoby ipc i kończy swoje działanie.

Proszę zwrócić uwagę, że kolejność wchodzenia procesów do sekcji krytycznej nie jest wymuszana w tym programie, więc może się zdarzyć, że czytelnicy

otrzymają dostęp do pamięci dzielonej przed wszystkimi pisarzami i odczytają pustą wiadomość lub, że pisarz nadpisze wiadomość zostawioną przez innego pisarza, zanim odczyta ją którykolwiek z czytelników.



The screenshot displays the CLion IDE interface. The top pane shows the source code for `my_own_implementation.c`. The code includes headers for `sem.h` and `shm.h`, defines `NR_PROD` as 16, and declares a union `semun semval`. It defines a `shdata` struct with `readers_number` and a `message` array. A `writer` function is implemented, which uses `sembuf` to manage the semaphore. The `main` function is partially visible.

The bottom pane shows a terminal window with the following output:

```
MacBook-Pro-Marina:semafory marinalukaci@.$ ./test
pm: poczatek; getpid()=36531
pm: uzyskujemy id wspolnej pamieci ...
pp: poczatek; getpid()=36534
pp2: poczatek; getpid()=36535
pp: uzyskujemy id wspolnej pamieci ...
pp2: uzyskujemy id wspolnej pamieci ...
pp2: konto1+konto2=158140828499495
pp2: konto1+konto2=601115427885282
pp2: konto1+konto2=865173514617235
pp2: konto1+konto2=1288273174356628
pp2: konto1+konto2=1702555987104591
pp2: konto1+konto2=2469548058529996
^CSIGINT: getpid()=36535, konto1+konto2=2597897721052298
SIGINT: getpid()=36531, konto1+konto2=2597897721052298
SIGINT: getpid()=36534, konto1+konto2=2597897721052298
pp2: konto1+konto2=2597897721052298
pm: wait() ...
pp2: koniec
pp: koniec
pm: wait()=36534 status=0000
```

The status bar at the bottom indicates the file encoding is UTF-8 with 4 spaces, and the IDE is ready to update.