Systemy operacyjne – studia stacjonarne 2022/23

Lab 13. Rozwiązywanie klasycznych problemów synchronizacji procesów przy pomocy mechanizmów IPC: implementacja problemu czytelników - pisarzy. Projekt nr 6.

Problem Czytelników i Pisarzy charakteryzuje procesy korzystające ze wspólnej bazy danych. Występują dwa typy procesów: procesy czytające i piszące, które współpracują ze sobą, ale równocześnie rywalizują o zasoby. Wiele procesów czytających może jednocześnie korzystać z zasobu, jednak pod warunkiem, że nie korzysta z niego w tym czasie żaden proces piszący. Procesy piszące wymagają wzajemnego wykluczania zarówno względem procesów czytających, jak i innych procesów piszących.

Priorytet czytelnika.

W rozwiązaniu zastosowano dwa binarne semafory **sp** i **w** :

- **sp** – dla zapewnienia wykluczenia wzajemnego procesu piszącego względem wszystkich innych procesów, inicjowany wartością 1 (sp = 1)
- **w** – dla zapewnienia wykluczenia wzajemnego procesowi czytającemu w chwilach rozpoczynania i kończenia korzystania z zasobu, inicjowany wartością 1 (w = 1)

W tym rozwiązaniu większy priorytet mają procesy czytające - uzyskują one bezzwłoczny dostęp do zasobu, gdy tylko nie korzysta z niego żaden proces piszący. Zmienna **lc** to licznik procesów czytających. Pseudokody procesu czytającego i piszącego:

```
          czytanie-1
begin
repeat
   wait(w)
   lc=lc+1
   if lc=1 then wait(sp)
   signal(w)
   czytanie
   wait(w)
   lc=lc-1
   if lc=0 then signal(sp)
   signal(w)
end
```

```
          pisanie-1
begin
repeat
   wait(sp)
   pisanie
   signal(sp)
end
```

Priorytet pisarza.

Aby korzystając z prostych semaforów binarnych, zapewnić większy priorytet dla procesów piszących - umożliwić pisanie procesowi piszącemu możliwie jak najszybciej - należy zastosować dwie zmienne licznikowe i pięć semaforów:

- **w1** – dla wykluczenie wzajemnego procesów czytających w chwili rozpoczynania i kończenia korzystania z zasobu
- **sp** – dla wykluczenia wzajemnego procesu piszącego względem wszystkich innych procesów
- **sc** – do ochrony wejścia do sekcji krytycznej procesu czytającego

- **w2** – dla wykluczenia wzajemnego procesów piszących w chwili rozpoczynania i kończenia korzystania z zasobu
- **w3** – dla zapewnienia priorytetu pisania nad czytaniem

Wszystkie semafory inicjowane są wartością 1:

w1=w2=w3=sc=sp=1

Zmienna **lc** to licznik procesów czytających, **lp** to licznik procesów piszących.

Pseudokody dla procesów czytających i piszących:

czytanie-2

```
begin
repeat
   wait(w3)
   wait (sc)
   wait(w1)
   lc=lc+1
   if lc=1 then wait (sp)
   signal (w1)
   signal(sc)
   signal(w3)
   czytanie
   wait (w1)
   lc=lc-1
   if lc=0 then signal(sp)
   signal (w1)
end
```

pisanie-2

```
begin
repeat
   wait(w2)
   lp=lp+1
   if lp=1 then wait(sc)
   signal (w2)
   wait(sp)
   pisanie
   signal(sp)
   wait(w2)
   lp=lp-1;
   if lp=0 then signal(sc)
   signal(w2)
end
```

Opis funkcji systemowych i przykłady ich implementacji do tworzenia, ustawiania, sterowania i usuwania semaforów podano w notatniku do Lab. 8.

Projekt nr 6.

Zadanie zsynchronizowania dwóch grup procesów: czytelników i pisarzy, konkurujących o dostęp do wspólnej czytelni. Proces czytelnik: co jakiś czas odczytuje informację zgromadzoną w czytelni i może to robić razem z innymi czytelnikami; Proces pisarz: co jakiś czas zapisuje swoją informację i musi wówczas przebywać sam w czytelni; W zadaniu tym może wystąpić tzw. zjawisko zagłodzenia jednej z grup procesów. Występuje ono wówczas, gdy proces nie zostaje wznowiony, mimo że zdarzenie, na które czeka występuje dowolną ilość razy. Za każdym razem, gdy proces ten mógłby być wznowiony, wybierany jest jakiś inny oczekujący proces.

Rozważ 2 rozwiązania problemu czytelników i pisarzy:

- Rozwiązanie z możliwością zagłodzenia pisarzy – priorytet czytelnika;
- Rozwiązanie z możliwością zagłodzenia czytelników – priorytet pisarza;

Rozwiązać zadanie w 2 wersjach przy pomocy semaforów. Przyjąć, że liczba miejsc w czytelni jest ograniczona i ustalona na pewną wartość M. Liczba czytelników oraz pisarzy może być zarówno większa jak i mniejsza od M. Napisać program do generowania procesów czytelników i pisarzy w oparciu o funkcje fork() i exec(). Zademonstrować zjawisko zagłodzenia procesów pisarza i procesów czytelnika. Do symulacji różnych prędkości działania programów użyć np. funkcji sleep() z losową liczbą sekund. Zaimplementuj obsługę przerwania wywołania funkcji systemowej semop(). Dla funkcji systemowych zaprogramować obsługę błędów w oparciu o funkcję perror() i zmienną errno.

Program główny sprawdza dopuszczalny limit procesów, które może użytkownik uruchomić w danym momencie i w przypadku, kiedy limit jest większy lub równy liczbie tworzonych procesów, uruchamia zadanie.

Przykładowe wywołanie „powielacza”.

./program_główny liczba_p liczba_c liczba_m

gdzie, liczba_p - liczba pisarzy,
liczba_c - liczba czytelników,
liczba_m - liczba miejsc w czytelni.

```
[]$ ./prog 2 5 3
```