SOI.B Zadanie na laboratorium 3 – grupa CS500 pon. 10-12

1. Zaimplementować mechanizm priorytetowych kolejek wiadomości dla komunikacji międzyprocesowej.
2. Wykorzystując zaimplementowany mechanizm kolejkowy przygotować części składowe systemu przetwarzającego wiadomości w kolejkach.
3. Przeprowadzić prezentację systemu, odpowiedzieć na pytania.

Uwagi szczegółowe do punktu 1

- Kolejki należy zaimplementować w systemie Linux przy użyciu semaforów i pamięci wspólnej.
  - Zalecam zapoznanie się ze stronami manuala linuksowego dla funkcji semget, semop, semctl, shmget, shmctl, shmat, shmdt.
  - Należy zadbać o zwalnianie wykorzystywanych zasobów systemowych (semaforów i pamięci dzielonej) – pozostawianie niezwolnionych zasobów będzie surowo karane w punktacji!
- Kolejki powinny zapewniać:
  - ograniczenie pojemności kolejki,
  - mechanizm producent-konsument,
  - dostarczanie w kolejności priorytetu, a w ramach każdego priorytetu według zasady FIFO (first in – first out),
  - możliwość nieblokującego sprawdzenia, czy kolejka jest pełna/pusta i/lub stopnia jej zapełnienia.
- Zalecana jest implementacja osobnej biblioteki realizującej funkcje mechanizmu kolejkowego, wykorzystywanej przez pozostałe programy implementowane w ramach zadania.

Uwagi szczegółowe do punktu 2

- System zawiera trzy kolejki (A, B i C) o zadanej pojemności każda. Komunikat zawiera trzy znaki z zakresu „A”, „B”, „C” (może też zawierać inne dane, jeśli z projektu wynika taka potrzeba). Komunikaty mogą mieć priorytet normalny lub wysoki.
- W systemie występują trzy rodzaje procesów: producenci (P), konsumenci (K) (tak naprawdę prosumenci, bo też mogą wstawiać do kolejek) oraz producenci specjalni (S).
- Proces producenta jest powiązany z jedną z kolejek i z zadaną częstością wrzuca do niej komunikat zawierający trzy losowe znaki z podanego zakresu.
- Proces konsumenta jest powiązany z jedną z kolejek i działa w pętli w następującym cyklu:
  - Pobierz komunikat z kolejki (blokująco – jeśli jest pusta, czekaj aż coś się pojawi).
  - Odczekaj pół sekundy.
  - Jeśli pobrany komunikat był pusty, odrzuć go (zakończenie przetwarzania) i przejdź do następnej iteracji.
  - W przeciwnym wypadku odczytaj i usuń z niego pierwszą literę*, a na koniec komunikatu z prawdopodobieństwem* ***pr*** *dodaj nową, wylosowaną*. Wrzuć zmodyfikowany komunikat do kolejki o identyfikatorze zgodnym z odczytanym znakiem.
- Proces producenta specjalnego wrzuca z zadaną częstością komunikaty wysokiego priorytetu, zawierające trzy losowe znaki z zakresu ABC, do losowo wybranej kolejki.

Uwagi szczegółowe do punktu 3

W systemie znajdują się trzy procesy producentów (P) oraz trzy procesy konsumentów (K), przyłączone po jednym każdego typu do każdej z kolejek. Ponadto istnieje jeden proces producenta specjalnego (S).
Każdy producent, zarówno zwykły (P), jak i specjalny (S), wytwarza jeden komunikat co 3 sekundy.

Pytania:

1. Niech ***pr*** = 0, czyli dodatkowe litery nie są dodawane.
   a) Jaka minimalna pojemność kolejek jest wymagana, żeby system na sensownym (parominutowym) horyzoncie działał całkowicie stabilnie, bez blokad?
   b) Jaki jest przeciętny czas obsługi[1] komunikatu o priorytecie zwykłym, a jaki o priorytecie wysokim? Długość kolejki powinna być wystarczająca, aby system działał całkowicie stabilnie.
2. Niech pojemność kolejek wynosi 20 (każda, nie łącznie).
   a) Przy jakiej wartości ***pr*** na sensownym (parominutowym) horyzoncie pojawiają się pierwsze przepełnienia kolejki?
   b) Przy jakiej wartości ***pr*** na sensownym (parominutowym) horyzoncie regularnie występują zakleszczenia?

Wyniki mogą być wizualizowane w dowolny czytelny sposób, umożliwiający prezentację działania systemu i udzielenie odpowiedzi na pytania. Zwracam uwagę, że w przypadku wypisywania komunikatów przez kilka procesów na jedną konsolę tekstową istnieje ryzyko mieszania się poszczególnych strumieni znaków. Można tego uniknąć przez użycie innego podejścia (osobne konsole tekstowe, wizualizacja graficzna 2D, 3D, telepatia, cokolwiek) lub poprzez synchronizację – w końcu i tak używamy semaforów...

Uwagi teoretyczne

Przede wszystkim: jeśli nie podano inaczej (a nie podano), to przez słowo „losowy" rozumiemy „z rozkładem jednostajnym". To powinno być oczywiste, ale skoro już wchodzimy w teorię, to zaznaczmy, że nie ma tu celowych pułapek. Żadnych dziwnych rozkładów, itp. Interpretacja jak najbardziej intuicyjna.

Co trzy sekundy producenci wrzucają do systemu łącznie 4 wiadomości, z których każda już w momencie wrzucenia wymaga łącznie co najmniej 4-krotnego obsłużenia – raz przez konsumenta kolejki, do której została wrzucona, i po razie przez konsumenta każdej z kolejek wskazanych literami w wiadomości. 4 *4 = 16.

Konsumenci w 3 sekundy dokonują 18 (3 konsumentów, 3 sekundy, 2x na sekundę) obsłużeń.

Teoretycznie przy ***pr*** = 0 jest zatem zapas. Jeśli mamy jednak doskonały generator liczb losowych, to gwarancji braku zatorów czy choćby przepełnień nie uzyskamy dla **żadnej** długości kolejek. Wystarczy bowiem skończona sekwencja zdarzeń (np. wszyscy konsumenci generują ciągle literę B), która ma przecież niezerowe prawdopodobieństwo. Bez specjalnego mechanizmu rozwiązywania takich sytuacji system zawsze może się zatkać. Widać jednak, że zapas jest spory, ponad 10%, a więc można oczekiwać, że typowo dla ***pr*** = 0 kolejki będą jednak typowo trzymać najwyżej kilka wiadomości na raz. Stąd wymaganie sensownego horyzontu czasowego – na nieskończonym zadanie jest niewykonalne, na umiarkowanie krótkim pewnie nawet nie trzeba dużych kolejek.

Z drugiej strony, dla ***pr*** = 1 żadna wiadomość nigdy nie opuści systemu, a zator wystąpi zawsze, i to dość szybko (zależność od długości kolejek jest zasadniczo liniowa, acz losowość znowu nam bruździ). Musi więc istnieć większa od 0 wartość ***pr*** pomiędzy tymi skrajnościami, dla której system zasadniczo nadal działa w typowej sytuacji, w sensownym czasie eksperymentu.

To tylko zarys analizy. Jak widać, chwila z kartką papieru i prostą matematyką może znacznie skrócić eksperymenty, dając dobre oszacowanie sensownych wartości.

Trzeba jednak też pamiętać, że doskonałym generatorem liczb losowych nie dysponujemy. Mamy tylko generatory pseudolosowe (z których domyślny w C rand() jest raczej kiepski). Ich własności mogą nam mocno zaburzyć te oszacowania... Bez eksperymentu się nie obędzie.

---

1 Realizacja tego punktu może wymagać wprowadzenia do komunikatu dodatkowych danych, np. identyfikatora albo stempla czasowego.