Systemy operacyjne

Laboratorium 4 - synchronizacja procesów przy użyciu monitorów

Prowadzący: mgr. inż. Aleksander Pruszkowski

Zdający: Jakub Sikora

Termin: wtorek 14:15 - 16:00

1. Treść zadania

Napisz w C++ dla środowiska Linux, system kolekcjonowania krótkich wiadomości tekstowych (maks. 128 znaków, ale nie mniej niż 16 znaków). Dla przypomnienia system ma bazować na synchronizacji dostępu do zasobów wykorzystujący mechanizm monitorów.

Zadaniem budowanego systemu ma być niezawodne zbieranie od klientów wiadomości, liczba klientów może być duża, ale system musi być gotowy do obsłużenia minimum 5 klientów.

Klienci - pojedynczy pod-proces lub wątek - „wrzucają" wiadomości do systemu, oprócz samej treści wiadomości wybierają priorytet wrzucanej wiadomość (np.: 0-zwykły, 1->priorytetowy).

System może zbierać wiadomości tylko w jednym pojemnym buforze. Mechanizm wkładania nowych wiadomości do tego bufora musi uwzględniać priorytety. Wszelkie operacje na buforze powinny być optymalizowane w taki sposób by nie kopiować niepotrzebnie wiadomości, oraz czas wkładania wiadomości oraz czas wyjmowania były możliwie jak najkrótsze.

Dodatkowo dla systemu utworzony ma być pod-proces lub wątek „czytelnik" zebranych wiadomości. Jego zadaniem jest pobieranie z bufora i przedstawianie wiadomości tekstowych na konsoli tekstowej. Zakłada się, że „czytelnik" będzie pobierał wiadomości z bufora, a w buforze wiadomości będą już poukładane zarówno względem priorytetów jaki i czasu ich włożenia.

Przemyśl bardzo dokładnie metodę testowania powstałego systemu, w szczególności zwróć uwagę na pokazanie równoczesnego działania wielu procesów umieszczających wiadomości w tym z różnymi priorytetami oraz współdziałanie w tym czasie „czytelnika".

Załóż, że program testowy będą działały automatycznie generując przez klientów fikcyjne wiadomości wyłącznie tekstowe, a „czytelnik" pokazywał je na konsoli.

2. Propozycja rozwiązania

Bufor - monitor

W pierwszej kolejności zaimplementuje klasę bufora `MessageBuffer` dziedziczącą/agregujący funkcjonalności klasy `Monitor` zaimportowanej z pliku `monitor.h` dostarczonego przez wykładowcę. Klasa będzie oferowała na zewnątrz trzy metody:

- `pop` - pobranie pierwszej w kolejności (z uwzględniem priorytetów) wiadomości
- `push` - wstawienie wiadomości zwykłej
- `push_pri` - wstawienie wiadomości priorytetowej

Każda z udostępnianych metod będzie wykorzystywać API klasy `Monitor` tak aby zapewnić synchronizowany dostęp - tylko jeden użytkownik może pracować na buforze.

Bufor komunikacyjny zostanie zrealizowany jako pojedyncza struktura przechowująca wiadomości. W pierwszej wersji projektu jako wewnętrzną strukturę danych zastosuję binarną stertę lub listę łączoną (jedno- lub dwukierunkową).

Użytkownicy

Użytkownikami systemu będą wątki procesu głównego. Założenie to znacząco ułatwia implementację struktury danych, która w tym przypadku może znaleźć się w pamięci procesu a nie w pamięci współdzielonej. Oddelegowany zostanie jeden wątek czytelnika, który z określoną częstotliwością będzie pobierał wiadomości ze struktury. Kolejne wątki (minimum 5) będą wpisywać wiadomości tekstowe. Wątki nie będą przeprowadzały żadnych czynności synchronizacyjncyh, nad wszystkim będzie czuwać sama struktura z monitorem.

3. Zbiór testów rozwiązania

Każdy nowy użytkownik, będzie wysyłał nową wiadomość co określony kwant czasu za pomocą procesu uruchomionego w tle. Każdy z użytkowników będzie wysyłał wiadomości z różnymi częstotliowściami.

W ramach testów przygotuje scenariusze testowe, podobne do tych z zadania trzeciego. Testy będą pokrywały następujące scenariusze:

- pisanie do pełnego bufora wiadomościami zwykłymi
- czytanie z pustego bufora wiadomościami zwykłymi
- zachowanie kolejności wiadomości
- poprawna obsługa wiadomości priorytetowych
- głodzenie procesów
- pisanie do pełnego bufora wiadomościami priorytetowymi
- czytanie z pustego bufora wiadomościami priorytetowymi

4. Realizacja rozwiązania

Wiadomość

Wiadomości przesylane w ramach chatu są reprezentowane przez obiekt typu `Message`. Typ ten implementuje listę jednokierunkową, jego polami są;

- bufor na wiadomość tekstową do 128 znaków
- wskaźnik na kolejny element typu `Message`

Monitor

Zgodnie z przedstawioną koncepcją rozwiązania, zaimplementowałem klasę `MessageBuffer` dziedziczącą po klasie `Monitor` obsługującej wzajemne wykluczanie. Instancja `MessageBuffer` posiada dwie zmienne warunkowe `prod` i `cons` typu `Condition`, które pilnują aby wątki producenckie nie pisały do pełnego bufora a wątki konsumenckie nie czytały z bufora pustego.

Do obsługi wybranej struktury danych jaką jest lista jednokierunkowa, jako pola klasy `MessageBuffer` dodałem trzy wskaźniki `head`, `pri_tail`, `tail` na typ `Message` która reprezentuje wiadomość chatu. Wskaźniiki te odpowiednio wskazują na początek listy, ostatni element priorytetowy oraz ostatni element listy.

Klasa `MessageBuffer` wystawia trzy metody publiczne `pop()`, `push(Message *m)`, `pu``sh_pri(Message *m)`. Metoda `pop()` wypisuje na ekran pierwszy element listy.

Metoda `push(Message *m)` wkłada na koniec bufora nowy element przekazywany przez wskaźnik `m`. Ostatnia metoda `push_pri(Message *m)` wkłada do bufora nowy element za ostatnim elementem priorytetowym a przed pierwszym zwykłym.

Konsument

Wątek konsumencki w pętli wywołuje metodę `pop()` na obiekcie typu `MessageBuffer`. Skutkuje to wypisaniem nowej wiadomości na terminal. W przypadku pustego bufora, wątek zatrzyma się do momentu aż w buforze pojawi się jakaś wiadomość i wątkowi zostanie zasygnalizowana możliwość odczytu.

Producent

Wątek zwykłego producenta w pętli tworzy nowy obiekt typu `Message`, której treścią jest informacja o numerze wątku i o kolejności wiadomości. Tworzenie nowego obiektu wiadomości jest wykonywane za pomocą operatora `new`, dlatego też należy zadbać o zabezpieczenie przed błędem alokacji.

```
try{
 new_msg = new Message(msg_content);
 } catch(std::bad_alloc& ba){
 std::cout << "Bad alloc from producer 0" << std::endl;
 exit(1);
 }
```

Po poprawnym utworzeniu wiadomości, wątek wywołuje na buforze metodę `push()`, z argumentem będącym wskaźnikiem na nowy obiekt `mb->push(new_msg);`.

Producent priorytetowy

Wątek producenta priorytetowego działa dokładnie tak samo jak zwykły producent z tą różnicą że zamiast metody zwykłego wkładania do bufora, wywoływana jest metoda `puhs_pri()`.

Program wynikowy

Program `monitor` testujący zaimplementowaną strukturę danych należy najpierw skompilować za pomocą polecenia `make`. Po uruchomieniu programu bez argumentów lub z argumentem `h` pojawi się okno z listą dostępnych testów. Odpowiedni test uruchamiamy podając jego numer jako pierwszy argument wywołania.

Przykład: Aby uruchomić test trzeci należy wywołać program w następujący sposób: `./monitor 3`.

Wysłanie wiadomości przez zwykłego użytkownika jest odpowednio sygnalizowane przez wypisanie zawartości komunikatu opatrzonego symbolem `TX` i timestampem. Użytkownik priorytetowy wysyła w identyczny sposób tylko jego wiadomość jest opatrzona symbolem `PX`. Odczyt komunikatu odbywa się w podobny sposób, jednak jest opisany symbolem `RX` i dodatkowo wyróżniony kolorem czerwonym.

5. Wyniki testów

Test zachowania kolejności wiadomości zwykłych

Wynik wywołania testu:

./monitor 0

RX 1545155611: Nie znalazlem wiadomosci

TX 1545155611: Message number 0 from 0 thread

RX 1545155611: Message number 0 from 0 thread

TX 1545155611: Message number 0 from 1 thread

TX 1545155611: Message number 0 from 2 thread

TX 1545155612: Message number 1 from 0 thread

TX 1545155612: Message number 1 from 1 thread

TX 1545155612: Message number 1 from 2 thread

TX 1545155613: Message number 2 from 0 thread

TX 1545155613: Message number 2 from 1 thread

TX 1545155613: Message number 2 from 2 thread

RX 1545155614: Message number 0 from 1 thread

TX 1545155614: Message number 3 from 0 thread

TX 1545155614: Message number 3 from 1 thread

TX 1545155614: Message number 3 from 2 thread

TX 1545155615: Message number 4 from 0 thread

TX 1545155615: Message number 4 from 1 thread

TX 1545155615: Message number 4 from 2 thread

TX 1545155616: Message number 5 from 0 thread

TX 1545155616: Message number 5 from 1 thread

TX 1545155616: Message number 5 from 2 thread

RX 1545155617: Message number 0 from 2 thread

TX 1545155617: Message number 6 from 0 thread

TX 1545155617: Message number 6 from 1 thread

TX 1545155617: Message number 6 from 2 thread

Wysłanie wiadomości przez zwykłego użytkownika jest odpowednio sygnalizowane przez wypisanie zawartości komunikatu opatrzonego symbolem `TX` i timestampem. Na samym początku konsument sygnalizuje brak wiadomosci w buforze. W momencie gdy w buforze znajdzie się pierwsza wiadomość, wybudzony zostanie konsument który odczyta nową wiadomość. W dalszej kolejności, konsument wypisuje wiadomości zgodnie z kolejnoścIa ich przychodzenia.

Test pisania do bufora zapełnionego przez wiadomości zwykłe

Wynik wywołania testu:

./monitor 1

TX 1545155948: Message number 0 from 0 thread

TX 1545155948: Message number 0 from 1 thread

TX 1545155948: Message number 0 from 2 thread

TX 1545155948: Message number 0 from 3 thread

TX 1545155948: Message number 0 from 4 thread

TX 1545155949: Message number 1 from 0 thread

TX 1545155949: Message number 1 from 1 thread

TX 1545155949: Message number 1 from 2 thread

TX 1545155949: Message number 1 from 3 thread

TX 1545155949: Message number 1 from 4 thread

TX 1545155950: Message number 2 from 0 thread

TX 1545155950: Message number 2 from 1 thread

TX 1545155950: Message number 2 from 2 thread

TX 1545155950: Message number 2 from 3 thread

TX 1545155950: Message number 2 from 4 thread

TX 1545155951: Message number 3 from 0 thread

TX 1545155951: Message number 3 from 1 thread

TX 1545155951: Message number 3 from 2 thread

TX 1545155951: Message number 3 from 3 thread

TX 1545155951: Message number 3 from 4 thread

TX 1545155952: BRAK MIEJSCA W BUFORZE

TX 1545155952: BRAK MIEJSCA W BUFORZE

TX 1545155952: BRAK MIEJSCA W BUFORZE

TX 1545155952: BRAK MIEJSCA W BUFORZE

TX 1545155952: BRAK MIEJSCA W BUFORZE

W tym teście uruchomione zostało 5 wątków producenckich i żaden konsumencki. Po zapełnieniu bufora w całości, każdy z wątków sygnalizuje brak miejsca w buforze a następnie wpada w stan zawieszenia.

Test czytania z pustego bufora

Wynik wywołania testu:

./monitor 2

RX 1545156023: Nie znalazlem wiadomosci

RX 1545156023: Nie znalazlem wiadomosci

W następnym teście uruchamiam dwa wątki próbujące czytać z pustego bufora. Oba sygnalizują brak wiadomości w buforze a następnie wpadają w stan zawieszenia.

Test obsługi wiadomości priorytetowych

Wynik testu:

./monitor 3

TX 1545156123: Message number 0 from 0 thread

TX 1545156123: Message number 0 from 1 thread

TX 1545156123: Message number 0 from 2 thread

PX 1545156123: Priority message number 0 from producer 1000

TX 1545156124: Message number 1 from 0 thread

TX 1545156124: Message number 1 from 1 thread

TX 1545156124: Message number 1 from 2 thread

RX 1545156124: Priority message number 0 from producer 1000

TX 1545156125: Message number 2 from 0 thread

TX 1545156125: Message number 2 from 1 thread

TX 1545156125: Message number 2 from 2 thread

PX 1545156125: Priority message number 1 from producer 1000

TX 1545156126: Message number 3 from 0 thread

TX 1545156126: Message number 3 from 1 thread

TX 1545156126: Message number 3 from 2 thread

TX 1545156127: Message number 4 from 0 thread

TX 1545156127: Message number 4 from 1 thread

TX 1545156127: Message number 4 from 2 thread

PX 1545156127: Priority message number 2 from producer 1000

RX 1545156127: Priority message number 1 from producer 1000

TX 1545156128: Message number 5 from 0 thread

TX 1545156128: Message number 5 from 1 thread

TX 1545156128: Message number 5 from 2 thread

TX 1545156129: Message number 6 from 0 thread

TX 1545156129: BRAK MIEJSCA W BUFORZE

TX 1545156129: BRAK MIEJSCA W BUFORZE

PX 1545156129: BRAK MIEJSCA W BUFORZE

TX 1545156130: BRAK MIEJSCA W BUFORZE

RX 1545156130: Priority message number 2 from producer 1000

TX 1545156130: Message number 6 from 1 thread

TX 1545156131: BRAK MIEJSCA W BUFORZE

RX 1545156133: Message number 0 from 0 thread

TX 1545156133: Message number 6 from 2 thread

W kolejnym teście uruchamiam trzy wątki zwykłych proucentów, jeden wątek producenta priorytetowego oraz jeden wątek konsumenta. Na listingu wyniku można zauważyć że wiadomości priorytetowe wypisywane są w pierwszej kolejności nawet gdy są wiadomości zwykłe oczekujące na wypisanie.

Problem zapełnienia bufora przez wiadomości priorytetowe

Wynik testu:

./monitor 5

PX 1545156276: Priority message number 0 from producer 0

PX 1545156276: Priority message number 0 from producer 1

PX 1545156276: Priority message number 0 from producer 2

PX 1545156276: Priority message number 0 from producer 3

PX 1545156277: Priority message number 0 from producer 4

PX 1545156278: Priority message number 1 from producer 0

PX 1545156278: Priority message number 1 from producer 1

PX 1545156278: Priority message number 1 from producer 2

PX 1545156278: Priority message number 1 from producer 3

PX 1545156279: Priority message number 1 from producer 4

PX 1545156280: Priority message number 2 from producer 0

PX 1545156280: Priority message number 2 from producer 1

PX 1545156280: Priority message number 2 from producer 2

PX 1545156280: Priority message number 2 from producer 3

PX 1545156281: Priority message number 2 from producer 4

PX 1545156282: Priority message number 3 from producer 0

PX 1545156282: Priority message number 3 from producer 1

PX 1545156282: Priority message number 3 from producer 2

PX 1545156282: Priority message number 3 from producer 3

PX 1545156283: Priority message number 3 from producer 4

PX 1545156284: BRAK MIEJSCA W BUFORZE

PX 1545156284: BRAK MIEJSCA W BUFORZE

PX 1545156284: BRAK MIEJSCA W BUFORZE

PX 1545156284: BRAK MIEJSCA W BUFORZE

PX 1545156285: BRAK MIEJSCA W BUFORZE

W tym teście uruchomiłem 5 wątków producentów priorytetowych oraz żadnego konsumenta. Po kilku iteracjach, bufor zapycha się i uniemożliwa wstawianie nowych wiadomości. Każdy z wątków po wypisaniu stosownego komunikatu o braku miejsca, zawiesza się i oczekuje na miejsce.

Zapełnienie bufora po połowie wiadomościami obu typów

Wynik testu:

./monitor 7

TX 1545156445: Message number 0 from producer 0

TX 1545156445: Message number 1 from producer 0

TX 1545156445: Message number 2 from producer 0

TX 1545156445: Message number 3 from producer 0

TX 1545156445: Message number 4 from producer 0

TX 1545156445: Message number 5 from producer 0

TX 1545156445: Message number 6 from producer 0

TX 1545156445: Message number 7 from producer 0

TX 1545156445: Message number 8 from producer 0

TX 1545156445: Message number 9 from producer 0

PX 1545156445: Priority message number 0 from producer 0

PX 1545156445: Priority message number 1 from producer 0

PX 1545156445: Priority message number 2 from producer 0

PX 1545156445: Priority message number 3 from producer 0

PX 1545156445: Priority message number 4 from producer 0

PX 1545156445: Priority message number 5 from producer 0

PX 1545156445: Priority message number 6 from producer 0

PX 1545156445: Priority message number 7 from producer 0

PX 1545156445: Priority message number 8 from producer 0

PX 1545156445: Priority message number 9 from producer 0

RX 1545156447: Priority message number 0 from producer 0

PX 1545156448: Priority message number 0 from producer 1000

TX 1545156448: BRAK MIEJSCA W BUFORZE

PX 1545156450: BRAK MIEJSCA W BUFORZE

RX 1545156450: Priority message number 1 from producer 0

TX 1545156450: Message number 0 from 1 thread

TX 1545156451: BRAK MIEJSCA W BUFORZE

RX 1545156453: Priority message number 2 from producer 0

PX 1545156453: Priority message number 1 from producer 1000

PX 1545156455: BRAK MIEJSCA W BUFORZE

RX 1545156456: Priority message number 3 from producer 0

TX 1545156456: Message number 1 from 1 thread

TX 1545156457: BRAK MIEJSCA W BUFORZE

RX 1545156459: Priority message number 4 from producer 0

PX 1545156459: Priority message number 2 from producer 1000

PX 1545156461: BRAK MIEJSCA W BUFORZE

RX 1545156462: Priority message number 5 from producer 0

TX 1545156462: Message number 2 from 1 thread

W ostatnim teście, na samym początku bufor jest synchronicznie wypełniany wiadomościami obu typów po połowie. Następnie uruchamiane są trzy wątki: jeden producenta zwykłego, jeden producenta priorytetowego oraz jeden wątek konsumenta. Na powyższym listingu widać dwie ważne rzeczy. Po pierwsze, do wypisywania pobierane są wiadomości priorytetowe. Po drugie, elementy w miarę możliwości są wpisywane po jednym do bufora zamiennie raz przez wątek specjalny a raz przez wątek zwykły. Przedstawiona implementacja skutecznie chroni wątki zwykłe przez zagłodzeniem ze strony wątków priorytetowych.