Warszawa, 10.12.2019

student: Patryk Jan Sozański
grupa: 2I5

SYSTEMY OPERACYJNE: LABORATORIUM NR 4
KONCEPCJA

Treść zadania

Napisz w C++ dla środowiska Linux, system kolekcjonowania krótkich wiadomości tekstowych (maks. 64 znaków, ale nie mniej niż 8 znaków). 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 watek - 'wrzucają' wiadomości do systemu, oprócz samej treści wiadomości wybierają priorytet wrzucanej wiadomość (np.: 0 to zwykły, 1 to 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 zapewnić by czas wkładania wiadomości oraz czas wyjmowania były możliwie jak najkrótsze.

Dodatkowo dla systemu utworzony ma być pod-proces lub watek '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 automatycznego testowania powstałego systemu. W szczególności zwróć uwagę na pokazanie równoczesnego działania wielu procesów (czy wątków) umieszczających wiadomości, w tym także z rożnymi priorytetami oraz współdziałanie w tym czasie 'czytelnika'.

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

Sposób rozwiązania problemu

Przy rozwiązywaniu problemu wykorzystywany jest model producent-konsument. Wątki funkcji „klientów/pisarzy” to producenci zapełniający bufor wiadomościami. Wątek funkcji „czytelnika” to konsument pobierający wiadomości z bufora i wyświetlający je w konsoli.

Bufor stanowi pamięć współdzieloną programu, do której dostęp jest regulowany przez monitor. Buforem jest kolejka FIFO. W celu implementacji monitora korzystam z biblioteki monitor.h.

Funkcje „klientów” zapełniają pojedynczo bufor losowo generowanymi wiadomościami tekstowymi spośród bazy dostępnych wiadomości typu string. Wiadomości dzielą się na dwie również przydzielane losowo klasy: wiadomości priorytetowe i zwykłe, za co odpowiada odpowiednie pole klasy „lista” przyjmujące jedną z dwóch wartości: 0 lub 1. W zależności od priorytetu wiadomości są one wstawiane w kolejkę w odpowiednim miejscu. Wiadomości o priorytecie niskim (wartość 0) są umieszczane zawsze na końcu kolejki, a wiadomości o priorytecie wysokim (wartość 1) w przedniej części. Oba typy wiadomości są posortowane w kolejności chronologicznej. Dla odróżnienia wiadomości priorytetowych od zwykłych, te pierwsze będą dla większej przejrzystości zapisywane WIELKIMI LITERAMI.

Wiadomości dotyczy ograniczenie długości. Jeżeli są krótsze od 8 znaków lub dłuższe od 64 znaków, to są zapominane i nie są wprowadzane do bufora.

Funkcje „czytelników” pobierają z bufora pojedynczo wiadomości zgodnie z reżimem kolejki FIFO. Wiadomości są już odpowiednio posortowane w buforze, a więc działanie funkcji ogranicza się do pobrania pierwszej wiadomości w kolejce i jej wyświetlenia do konsoli.

Bufor cechuje się maksymalną długością (maksymalna ilość zapamiętanych wiadomości). W przypadku osiągnięcia długości maksymalnej nie jest możliwe dalsze wypełnianie bufora. Wtedy wątek funkcji „pisarza” zostaje uśpiony i oczekuje, aż w buforze zwolni się miejsce (wiadomość nie zostaje utracona). Gdy bufor jest pusty, a wątek funkcji „czytelnika” próbuje pobrać wiadomość z bufora, również następuje uśpienie wątku i oczekiwanie, aż bufor się wypełni.

Testowanie

a) działanie ogólne:

W celu przetestowania ogólnego działania programu utworzone zostaje menu wyboru, dzięki któremu użytkownik może sam i dowolnie kierować wątkami „pisarzy" i „czytelników" oraz obserwować działanie programu. Użytkownik poza możliwością utworzenia odpowiednich wątków, ma również wgląd bufora, mogąc tym samym kontrolować poprawność jego wypełnienia (kolejność wiadomości, długość wiadomości, długość bufora). Użytkownik może także doprowadzić do powstania warunków brzegowych (np. maksymalne wypełnienie bufora lub próba pobierania wiadomości z bufora pustego) oraz obserwować działanie programu.

b) przepełnienie bufora:

W celu przetestowania działania programu w sytuacji osiągnięcia maksymalnej długości bufora, zostaje nadana mała wartość maksymalnej długości bufora (np. 5). Następnie zostają uruchomione wątki „pisarzy" w liczbie przewyższającej maksymalną długość bufora. Działanie programu jest możliwe do obserwacji, dzięki odpowiednim komunikatom wyświetlanym w konsoli oraz wglądowi do bufora. Następnie uruchomione zostają wątki „czytelników", które pobierają kolejno wiadomości z bufora, zwalniając w nim miejsce i dając tym samym możliwość wątkom „pisarzy" do obudzenia się, co odzwierciedlane jest w konsoli.

c) pusty bufor:

W celu przetestowania działania programu w sytuacji, gdy bufor jest pusty, uruchamiane zostają wątki „czytelników" w dużej ilości. Działanie programu jest możliwe do obserwacji, dzięki odpowiednim komunikatom wyświetlanym w konsoli oraz wglądowi do bufora. Następnie uruchomione zostają wątki „pisarzy", które dodają kolejno wiadomości do bufora i dając tym samym możliwość wątkom „czytelników" do obudzenia się, co odzwierciedlane jest w konsoli.

d) priorytet:

W celu przetestowania poprawności umieszczania wiadomości w buforze względem ich priorytetów utworzone zostają wątki „pisarzy" wykonujące różne sekwencje dodawania wiadomości np. same wiadomości o priorytecie 0, same wiadomości o priorytecie 1, wiadomości o losowych priorytetach. Użytkownik ma możliwość śledzenia zmian zapełnienia bufora w konsoli. W celu uzyskania przejrzystości programu testowego, wiadomości otrzymują konkretne treści odpowiadające chronologii ich nadania oraz priorytetowi.