Warszawa, 17.12.2019

student: Patryk Jan Sozański
grupa: 2I5
nr indeksu: 300258

SYSTEMY OPERACYJNE: LABORATORIUM NR 4
RAPORT Z WYKONANEGO ĆWICZENIA

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 zaimplementowana w pliku „lista.h”. W celu implementacji monitora korzystam z biblioteki „monitor.h” pobranej ze strony prowadzącego.

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 w buforze WIELKIMI LITERAMI.

Każda wiadomość jest sygnowana znacznikiem czasu, który odpowiada czasowi jej wstawienia do bufora.

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. Treść wiadomości jest wyświetlana wraz z znacznikiem czasu.

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.

Uwaga: więcej szczegółów w plikach „main.cpp”, „lista.cpp” oraz „monitor.h”

Testowanie

a) test działania ogólnego:

funkcja void tDzialanieOgolne();

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.

Prezentacja działania:

Menu wyboru testu oraz wybór odpowiedniego testu (1).

Wynik:

```
# WYBIERZ TEST:
## 1 TEST OGOLNEGO DZIALANIA
## 2 TEST PRZEPELNIENIA BUFORA
## 3 TEST PUSTEGO BUFORA
## 4 TEST PRIORYTETOW
## 4 TEST PRIORYTETOW
## 5 TEST DLUGOSCI WIADOMOSCI
## 0 WYJDZ
1
Co chcesz uruchomic?:
(p) - producent
(k) - konsument
(w) - wyswielt liste wiadomosci
(q) - quit
```

Wprowadzona sekwencja poleceń: „p”, „p”, „p”, „w” (trzykrotne uruchomienie wątków „pisarzy” i wyświetlenie zawartości bufora).

Wynik:

```
p

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: marchewka o priorytecie: 0 znacznik czasu: 1576662358
p

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: kalarepa o priorytecie: 0 znacznik czasu: 1576662359
p

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: ziemniaki o priorytecie: 1 znacznik czasu: 1576662360
w

### LISTA WIADOMOSCI
## WIADOMOSC:                                                    ZNACZNIK CZASU:
# ZIEMNIAKI                                                     1576662360
# marchewka                                                     1576662358
# kalarepa                                                      1576662359
```

Wprowadzona sekwencja poleceń: „k”, „k”, „w” (dwukrotne uruchomienie wątków „konsumentów” i wyświetlenie zawartości bufora).

Wynik:

```
k

### CZYTELNIK ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: ZIEMNIAKI znacznik czasu: 1576662360
k

### CZYTELNIK ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: marchewka znacznik czasu: 1576662358
w

### LISTA WIADOMOSCI
## WIADOMOSC:                                          ZNACZNIK CZASU:
# kalarepa                                            1576662359
```

b) test przepełnienia bufora:

funkcja void tPrzepelnienieBufora();

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 odpowiednimi komunikatami w konsoli.

Prezentacja działania:

Wybór odpowiedniego testu (2) z menu wyboru.

Wynik:

Poniższy obraz przedstawia fragment testu odpowiadający sytuacji dodawania przez wątki „pisarzy” wiadomości do bufora, gdy ten jeszcze nie jest zapełniony (5-krotne dodanie wiadomości).

```
### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: pietruszka o priorytecie: 0 znacznik czasu: 1576662823

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: brukselka o priorytecie: 1 znacznik czasu: 1576662824

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: kalarepa o priorytecie: 0 znacznik czasu: 1576662825

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: rzodkiewka o priorytecie: 0 znacznik czasu: 1576662825

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: slonecznik o priorytecie: 0 znacznik czasu: 1576662826
```

Poniższy obraz przedstawia fragment testu odpowiadający sytuacji dodawania przez wątki „pisarzy” wiadomości do bufora, gdy ten jest już zapełniony.

```
### PISARZ ROZPOCZYNA PRACE

###BUFOR PELNY - NIE MOZNA DODAC WIADOMOSCI


### PISARZ ROZPOCZYNA PRACE

###BUFOR PELNY - NIE MOZNA DODAC WIADOMOSCI


### PISARZ ROZPOCZYNA PRACE

###BUFOR PELNY - NIE MOZNA DODAC WIADOMOSCI
```

Poniższy obraz przedstawia fragment testu odpowiadający sytuacji odczytywania przez wątki „czytelników” wiadomości z bufora. Widoczne jest budzenie się wątków „pisarzy”, gdy „czytelnik” odczytując wiadomość, zwalnia miejsce w buforze.

```
### CZYTELNIK ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: ziemniaki o priorytecie: 0 znacznik czasu: 1576662829

## Czytelnik pobrał z bufora wiadomosc: BRUKSELKA znacznik czasu: 1576662824


### CZYTELNIK ROZPOCZYNA PRACE

### WIADOMOSC NIE SPELNIA WARUNKOW DLUGOSCI

## Producent dodal do bufora wiadomosc: soi o priorytecie: 1 znacznik czasu: 1576663306

## Czytelnik pobrał z bufora wiadomosc: pietruszka znacznik czasu: 1576662823


### CZYTELNIK ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: kalarepa znacznik czasu: 1576662825


### CZYTELNIK ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: rzodkiewka znacznik czasu: 1576662825


### CZYTELNIK ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: slonecznik znacznik czasu: 1576662826
```

c) test pustego bufora:

funkcja void tPustegoBufora();

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 odpowiednimi komunikatami w konsoli.

Prezentacja działania:

Wybór odpowiedniego testu (3) z menu wyboru.

Wynik:

Poniższy obraz przedstawia fragment testu odpowiadający sytuacji odczytywania przez wątki „czytelników” wiadomości z bufora, gdy ten jest pusty.

```
### CZYTELNIK ROZPOCZYNA PRACE

###BUFOR PUSTY - NIE MOZNA POBRAC WIADOMOSCI


### CZYTELNIK ROZPOCZYNA PRACE

###BUFOR PUSTY - NIE MOZNA POBRAC WIADOMOSCI


### CZYTELNIK ROZPOCZYNA PRACE

###BUFOR PUSTY - NIE MOZNA POBRAC WIADOMOSCI


### CZYTELNIK ROZPOCZYNA PRACE

###BUFOR PUSTY - NIE MOZNA POBRAC WIADOMOSCI


### CZYTELNIK ROZPOCZYNA PRACE

###BUFOR PUSTY - NIE MOZNA POBRAC WIADOMOSCI
```

Poniższy obraz przedstawia fragment testu odpowiadający sytuacji dodawania przez wątki „pisarzy” wiadomości do bufora. Widoczne jest budzenie się wątków „czytelników”, gdy „pisarz” dodając wiadomość, zapełnia miejsce w buforze.

```
### PISARZ ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: SLONECZNIK znacznik czasu: 1576663747

## Producent dodal do bufora wiadomosc: slonecznik o priorytecie: 1 znacznik czasu: 1576663747


### PISARZ ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: slonecznik znacznik czasu: 1576663754

## Producent dodal do bufora wiadomosc: slonecznik o priorytecie: 0 znacznik czasu: 1576663754


### PISARZ ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: slonecznik znacznik czasu: 1576663755

## Producent dodal do bufora wiadomosc: slonecznik o priorytecie: 0 znacznik czasu: 1576663755


### PISARZ ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: kalafior znacznik czasu: 1576663756

## Producent dodal do bufora wiadomosc: kalafior o priorytecie: 0 znacznik czasu: 1576663756


### PISARZ ROZPOCZYNA PRACE

## Czytelnik pobrał z bufora wiadomosc: SLONECZNIK znacznik czasu: 1576663757

## Producent dodal do bufora wiadomosc: slonecznik o priorytecie: 1 znacznik czasu: 1576663757
```

d) <u>test priorytetów:</u>

funkcja void tPriorytetow();

W celu przetestowania poprawności umieszczania wiadomości w buforze względem ich priorytetów, utworzone zostają wątki „pisarzy” wykonujące różne sekwencje dodawania wiadomości (sekwencję ustala użytkownik poprzez wybieranie odpowiednich opcji z menu wyboru) np. same wiadomości o priorytecie 0, same wiadomości o priorytecie 1, wiadomości o losowych priorytetach. Użytkownik ma możliwość śledzenia zmian zapełnienia bufora w konsoli.

Prezentacja działania:

Wybór odpowiedniego testu (4) z menu wyboru.

Wynik:

```
Co chcesz uruchomic?:
(0) - dodaj wiadomosc o niskim priorytecie
(1) - dodaj wiadomosc o wysokim priorytecie
(k) - konsument
(w) - wyswielt liste wiadomosci
(q) - quit
```

Wprowadzona sekwencja poleceń: „0”, „0”, „0”, „w” (trzykrotne dodanie wiadomości o priorytecie niskim przez wątki „pisarzy” i wyświetlenie zawartości bufora).

Wynik:

```
### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: marchewka o priorytecie: 0 znacznik czasu: 1576664299
0

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: roszponka o priorytecie: 0 znacznik czasu: 1576664301
0

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: ziemniaki o priorytecie: 0 znacznik czasu: 1576664302
w

### LISTA WIADOMOSCI
## WIADOMOSC:                                                    ZNACZNIK CZASU:
# marchewka                                                     1576664299
# roszponka                                                     1576664301
# ziemniaki                                                     1576664302
```

Wprowadzona sekwencja poleceń: „1”, „1”, „1”, „w” (trzykrotne dodanie wiadomości o priorytecie wysokim przez wątki „pisarzy” i wyświetlenie zawartości bufora).
Po wpisaniu polecenia „1” za drugim razem, wiadomość nie została dodana do bufora, ponieważ wylosowana wartość nie spełniała warunków długości.

Wynik:

```
### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: rzodkiewka o priorytecie: 1 znacznik czasu: 1576664511
1

### PISARZ ROZPOCZYNA PRACE

### WIADOMOSC NIE SPELNIA WARUNKOW DLUGOSCI

## Producent dodal do bufora wiadomosc: soi o priorytecie: 1 znacznik czasu: 1576664513
1

### PISARZ ROZPOCZYNA PRACE

## Producent dodal do bufora wiadomosc: rzodkiewka o priorytecie: 1 znacznik czasu: 1576664518
w

### LISTA WIADOMOSCI
## WIADOMOSC:                                                    ZNACZNIK CZASU:
# RZODKIEWKA                                                    1576664511
# RZODKIEWKA                                                    1576664518
# kalafior                                                      1576664480
# roszponka                                                     1576664482
# marchewka                                                     1576664483
```

e) test długości wiadomości

funkcja void tDlugosci();

W celu przetestowania poprawności działania programu w sytuacji, gdy wątki „pisarzy” próbują dodać do bufora wiadomości nie spełniające założeń (tj. o długości nie mieszczącej się w przedziale [8; 64]), utworzone zostaje menu wyboru, dzięki któremu użytkownik może dodawać wiadomości za krótkie lub za długie i obserwować wyniki programu w konsoli.

Prezentacja działania:

Wybór odpowiedniego testu (5) z menu wyboru.

Wynik:

```
Co chcesz uruchomic?:
(0) - dodaj wiadomosc za krotka
(1) - dodaj wiadomosc za dluga
(k) - konsument
(w) - wyswielt liste wiadomosci
(q) - quit
```

Wprowadzona sekwencja poleceń: „0”, „1”, „w” (dodanie wiadomości za krótkiej, dodanie wiadomości za długiej i wyświetlenie zawartości bufora).

Wynik:

```
### PISARZ ROZPOCZYNA PRACE

### WIADOMOSC NIE SPELNIA WARUNKOW DLUGOSCI

## Producent dodal do bufora wiadomosc: soi o priorytecie: 0 znacznik czasu: 1576665244
1

### PISARZ ROZPOCZYNA PRACE

### WIADOMOSC NIE SPELNIA WARUNKOW DLUGOSCI

## Producent dodal do bufora wiadomosc: dlugosc wieksza niz szescdziesiat cztery znaki dlugosc wie
ksza niz szescdziesiat cztery znaki o priorytecie: 0 znacznik czasu: 1576665245
w

### LISTA WIADOMOSCI
## WIADOMOSC:                                              ZNACZNIK CZASU:
```