Adam Detmer 155976
Jakub Buler 155987

Ukończony

Informacje wstępne

- Nasz projekt składa się z pary programów: serwer i klient.
- Serwer traci wszystkie dane przy zakończeniu działania programu
- Maksymalna liczba użytkowników, tematów oraz liczba wiadomości odbieranych asynchronicznie w czasie jednego odświeżenia programu klienta są ustalone w programie za pomocą stałych zdefiniowanych
- Klienci są rozróżnialni przez swoje unikatowe 3 cyfrowe ID
- Wiadomości wysyłane od klienta do serwera mają typ 1, a wiadomości wysyłane od serwera do klienta są wysyłane z typem równym ID odbiorcy

Struktury

Struktura użytkownika:

```
{
    long type;
    char name[NAME_DL];
    int id;
    struct temat subskrypcje[MAX_TEMAT];
    int liczbasub;
    int zbanowani[MAX_USER];
    int liczbazbnowanych;
};
```

- `type` - służy do określenia typu wiadomości gdy chcemy przesłać informację o danym kliencie.
- `name` - zawiera nazwę klienta.
- `id` - unikalny identyfikator klienta w postaci liczby 3 cyfrowej.
- `subskrypcje` - tablice subskrypcji klienta.
- `liczbasub` - informacja o tym ile subskrypcji posiada dany klient.
- `zbanowani` - tablica klientów zawierająca id klientów od których nie chcemy otrzymywać wiadomości.
- `liczbazbnowanych` - informacja o tym ilu użytkowników zbanował klient

Struktura tematu:

```
{
    long type;
    int id_topic;
    int topic_type;
    int ilejeszcze;
};
```

- `type` - służy do określenia typu wiadomości.

- `id_topic` - unikalny identyfikator tematu podawany przez stwórcę tematu.
- `topic_type` - wartość liczbowa określająca operację: 2 - tworzenie tematu 1 - subskrypcja permanentna, 0 - czasowa.
- `ilejeszcze` - wartość liczbowa określająca w przypadku subskrypcji czasowej ile wiadomości należy wysłać użytkownikowi.

Struktura msglogin

```
{
    long type;
    char name[NAME_DL];
    int id;
};
```

- `type` - służy do określenia typu wiadomości.
- `name[NAME_DL]` - zawiera informacje o nazwie klienta, który chce się zalogować.
- `id` - zawiera unikalne id klienta, który chce się zalogować

Struktura signal

```
{
     long type;
     int odp;
};
```

- `type` - służy do określenia typu wiadomości
- `odp` - służy do określenia treści wiadomości w postaci pojedynczej liczby

Struktura wiadomosc

```
{
    long type;
    char tekst[1024];
};
```

- `type` - służy do określenia typu wiadomości
- `tekst` - służy do określenia treści wiadomości w postaci tekstu

Struktura wiadomosc_tematyczna

```
{
    long type;
    char tekst[1024];
    int id_wysylanego;
    int id_subskrypcji;
    int priorytet; //1-10
};
```

- `type` - służy do określenia typu wiadomości
- `tekst` - służy do określenia treści wiadomości w postaci tekstu
- `id_wysylanego` - unikalny identyfikator klienta, który wysyła wiadomość.
- `id_subskrypcji` - unikalny identyfikator tematu, na który ma zostać wysłana wiadomość.
- `priorytet` - wartość liczbowa 1-10 określająca priorytet wiadomości (1-najszybciej, 10-najwolniej).

Funkcje użytkownika

1. **Logowanie(realizowana przy uruchomieniu programu klienta)**
   Klient wysyła wiadomość `signal` o `odp` równej 0, do serwera by poinformować go, że chcę się zalogować. Następnie klient wpisuje swoją nazwę oraz id do konsoli. Kolejno jest to wysyłane do serwera w kolejce `kolejka_logowanie` w strukturze `msg logowanie`, natomiast serwer analizuje, czy użytkownik o podanym identyfikatorze już istnieje. Jeżeli nie to stwarza konto w postaci struktury `newUser` i przechowuje o nim informacje w tablicy `users`, w przeciwnym razie program klienta jest wyłączany. Dalej serwer wysyła wiadomość zwrotną zależną od wyniku powyższych działań.
2. **Subskrypcja tematu**
   Klient wysyła wiadomość `signal` o `odp` równej 1.
   Klient wysyła wiadomość `nadawca` (`struktura signal`) do serwera z jego id. Następnie serwer dostarcza użytkownikowi aktualne tematy jakie już istnieją(`wysyłana jest struktura wiadomość, nowa_wiadomosc`). Klient wybiera, który temat chciałby zasubskrybować, czy ma to być subskrypcja trwała czy czasowa. W drugim przypadku podaję również ile wiadomości chciałby z danego tematu odebrać. Następnie klient wysyła poprzez kolejkę `subskrypcja`, struktury `konto` oraz `nowa_subskrypcja` (`struktura temat`). Serwer dodaje do struktury `user` należącej do subskrybenta, dodając `nowa_subskrypcja` do jego tablicy `subskrypcje` oraz powiększa `liczbasub` tego klienta. W odpowiedzi serwer wysyła poprzez tę samą kolejkę w strukturze `wiadomosc_serwera` (`struktura wiadomosc`) wynik realizacji tej operacji, czy udała się czy też nie oraz z jakiego powodu.
3. **Tworzenie nowego tematu**
   Klient wysyła wiadomość `signal` o `odp` równej 2.
   Klient podaje identyfikator tematu, który chciały stworzyć i jest on wysyłany do serwera w kolejce `subskrypcja`. Serwer otrzymuje wyżej podany identyfikator oraz sprawdza czy inny temat o danym id istnieje. Jeżeli nie to owy temat zostaje dodany do tablicy tematów (`tematy`), zwiększa się liczba tematów (`numtemats`) oraz zostaje wysłana wiadomość do wszystkich zalogowanych użytkowników mówiąca, że powstał nowy temat wraz z jego id,odbywa się to poprzez kolejkę `wszyscy` przesyłając strukturę `nowa_wiadomosc`. Twórca tematu dostaje od serwera informacje poprzez kolejkę `subskrypcja` w strukturze `nowa_wiadomosc` o rezultacie operacji jakiej dokonał.
4. **Wysyłanie wiadomości**
   Klient wysyła wiadomość `signal` o `odp` równej 3.
   Klient wysyła wiadomość `nadawca` (`struktura signal`) do serwera z jego id. Klient od serwera otrzymuje wszystkie istniejące tematy (ich id). Klient podaje, na który temat chciałby napisać wiadomość, jej treść oraz priorytet. Wszystkie te informacje są wysyłane do serwera w kolejce `wysylanie_wiadomosci` w strukturze `wyslana_wiadomosc` (`wiadomosc_tematyczna`). Serwer sprawdza czy dane zostały podane w odpowiedni sposób: czy istnieje dany temat oraz czy priorytet mieści się w zakresie. Jeżeli tak to serwer szuka użytkowników, którzy

subskrybują dany temat, sprawdza czy klient wysyłający wiadomośc nie jest zbanowanym u tego użytkownika. Jeżeli nie to w zależności od rodzaju subskrypcji w dany sposób jest wysyłana wiadomość (jeżeli typ jest czasowy to od `ilejeszcze` jest odejmowane 1). Jeżeli dany klient jest zbanowany to wiadomość nie zostanie wysłana do określonego użytkownika. Klient wysyłający otrzymuje wiadomość w kolejce `wysylanie_wiadomosc` w strukturze `sukces_wyslania` (`wiadomosc`) o rezultacie wykonanej przez niego operacji.

5. **Odbieranie synchroniczne**
   Klient wysyła wiadomość `signal` o `odp` równej 4.
   W przypadku wybrania tej opcji, serwer nie podejmuje żadnych działań.Klient wówczas jednak odbiera wszystkie wysłane wiadomości tematyczne z tematów z kolejki `wszyscy`, które subskrybuje i zapisuje do tablicy `wiadomosci_odebrane`. Następnie wyświetla odebrane wiadomości na ekran w kolejności zgodnej z ich priorytetem(1- najszybciej, 10- najwolniej). Po wyświetleniu wiadomości, w przypadku wykonania kolejnego odbierania synchronicznego klient zaczyna od pustej tablicy `wiadomosci_odebrane`.
6. **Banowanie użytkownika**
   Klient wysyła wiadomość `signal` o `odp` równej 5.
   Klient przesyła informacje o swoim id do serwera (kolejka `do_banow`), na co w odpowiedzi dostaje wiadomość zawierająca nazwy oraz id pozostałych klientów. Następnie klient przesyła id użytkownika, którego chce zbanować, a serwer dodaje otrzymane id do tablicy `zbanowani` w strukturze `user` naszego banującego oraz zwiększa jego `liczbazbnowanych` o 1.
7. **Odświeżanie**
   Klient wysyła wiadomość `signal` o `odp` równej 9.
   Pozwala to klientowi przejść przez całą pętlę, co pozwala odbierać mu wiadomości w sposób asynchroniczny w ilość `MAX_ODEBRANE`, jeżeli owe istnieją.