Zadanie 1

Sieciowa rezerwacja biletów

Zaimplementuj serwer UDP obsługujący sieciową rezerwację biletów na wydarzenia. Serwer ma być jednowątkowy i powinien obsługiwać klientów symultanicznie.

Parametry serwera

Serwer akceptuje w linii poleceń następujące parametry:

- `-f file` – nazwa pliku z opisem wydarzeń poprzedzona opcjonalnie ścieżką wskazującą, gdzie szukać tego pliku, obowiązkowy;
- `-p port` – port, na którym nasłuchuje, opcjonalny, domyślnie 2022;
- `-t timeout` – limit czasu w sekundach, opcjonalny, wartość z zakresu od 1 do 86400, domyślnie 5.

Serwer powinien dokładnie sprawdzać poprawność parametrów. Błędy powinien zgłaszać, wypisując stosowny komunikat na standardowe wyjście diagnostyczne i kończąc działanie z kodem 1.

Plik `file` zawiera opis wydarzeń. Każde wydarzenie opisane jest w dwóch kolejnych liniach. Pierwsza z tych linii zawiera dowolny niepusty tekst o długości co najwyżej 80 znaków (nie licząc znaku przejścia do nowej linii), niezawierający znaku o kodzie zero, będący właściwym opisem wydarzenia. Druga z tych linii zawiera liczbę dostępnych biletów. Jest to wartość z przedziału od 0 do 65535 zapisana przy podstawie dziesięć. W drugiej linii nie ma innych znaków niż cyfry. Wolno założyć, że zawartość pliku `file` jest poprawna.

Komunikaty wymieniane między klientem a serwerem

W komunikatach między klientem a serwerem mogą wystąpić następujące pola:

- `message_id` – 1 oktet, pole binarne;
- `description_length`– 1 oktet, pole binarne, liczba oktetów w polu `description`;
- `description` – opis wydarzenia, dowolny niepusty tekst, niezawierający znaku o kodzie zero ani znaku przejścia do nowej linii;
- `ticket_count` – 2 oktety, pole binarne;
- `event_id` – 4 oktety, pole binarne, unikalny identyfikator wydarzenia, generowany przez serwer, wartość z zakresu od 0 do 999999;
- `reservation_id` – 4 oktety, pole binarne, unikalny identyfikator rezerwacji, generowany przez serwer, wartość większa niż 999999;
- `cookie` – 48 oktetów, znaki ASCII o kodach z zakresu od 33 do 126, unikalny, trudny do odgadnięcia napis potwierdzający rezerwację, generowany przez serwer;
- `expiration_time` – 8 oktetów, pole binarne, liczba sekund od początku epoki uniksa;
- `ticket`– 7 oktetów, znaki ASCII, tylko cyfry i wielkie litery alfabetu angielskiego, unikalny kod biletu, generowany przez serwer.

Wartości w binarnych polach wielooktetowych zapisuje się w porządku sieciowym.

Komunikaty wysyłane przez klienta

Klient wysyła następujące komunikaty (nazwa komunikatu, lista pól, wartości pól, opis):

- `GET_EVENTS` – `message_id` = 1, prośba o przysłanie listy wydarzeń i liczb dostępnych biletów na poszczególne wydarzenia;
- `GET_RESERVATION` – `message_id` = 3, `event_id`, `ticket_count` > 0, prośba o zarezerwowanie wskazanej liczby biletów na wskazane wydarzenia;
- `GET_TICKETS` – `message_id` = 5, `reservation_id`, `cookie`, prośba o wysłanie zarezerwowanych biletów.

Komunikaty wysyłane przez serwer

Serwer wysyła następujące komunikaty (nazwa komunikatu, lista pól, wartości pól, opis):

- `EVENTS` – `message_id` = 2, powtarzająca się sekwencja pól `event_id`, `ticket_count`, `description_length`, `description`, odpowiedź na komunikat `GET_EVENTS` zawierająca listę opisów wydarzeń i liczb dostępnych biletów na każde wydarzenie;
- `RESERVATION` – `message_id` = 4, `reservation_id`, `event_id`, `ticket_count`, `cookie`, `expiration_time`, odpowiedź na komunikat `GET_RESERVATION` potwierdzająca rezerwację, zawierająca czas, do którego należy odebrać zarezerwowane bilety;
- `TICKETS` – `message_id` = 6, `reservation_id`, `ticket_count` > 0, `ticket`, …, `ticket`, odpowiedź na komunikat `GET_TICKETS` zawierająca `ticket_count` pól typu `ticket`;
- `BAD_REQUEST` – `message_id` = 255, `event_id` lub `reservation_id`, odmowa na prośbę zarezerwowania biletów `GET_RESERVATION` lub wysłania biletów `GET_TICKETS`.

Komunikat `EVENTS` musi się zmieścić w jednym datagramie UDP. Jeśli opis wszystkich wydarzeń nie mieści się, to należy wysłać komunikat `EVENTS` zawierający tyle (dowolnie wybranych) opisów, ile się zmieści.

Czas, do którego należy odebrać zarezerwowane bilety. Jest to czas w sekundach od początku epoki uniksa wyznaczony jako czas odebrania komunikatu `GET_RESERVATION` powiększony o wartość parametru `timeout`.

Serwer odpowiada komunikatem `BAD_REQUEST` z wartością `event_id` w odpowiedzi na komunikat `GET_RESERVATION`, jeśli prośba nie może być zrealizowana, klient podał nieprawidłowe `event_id`, prosi o zero biletów, liczba dostępnych biletów jest mniejsza niż żądana, bilety zostały w międzyczasie zarezerwowane przez innego klienta, żądana liczba biletów nie mieści się w komunikacie `TICKETS` (który musi się zmieścić w jednym datagramie UDP).

Serwer odpowiada komunikatem `BAD_REQUEST` z wartością `reservation_id` w odpowiedzi na komunikat `GET_TICKETS`, jeśli minął czas na odebranie biletów, klient podał nieprawidłowe `reservation_id` lub `cookie`.

Jeśli klient nie odbierze biletów w czasie wskazanym w komunikacie `RESERVATION`, rezerwacja zostaje anulowana, a bilety wracają do puli dostępnych biletów. Klient może wielokrotnie wysłać komunikat `GET_TICKET`. Jeśli serwer już co najmniej raz wysłał bilety, to przestaje obowiązywać limit czasu i serwer wysyła bilety ponownie, używając komunikatu `TICKETS`.

Serwer nie odpowiada na komunikaty o niepoprawnej długości i z błędną wartością pola `message_id`. Serwer powinien być możliwe odporny na złośliwe działanie klienta.

Dodatkowe uwagi

Wystarczy zaimplementować obsługę IP w wersji 4.

Program nie może wysyłać komunikatów niezgodnych ze specyfikacją. Program nie może naruszać ochrony pamięci. Nie wolno dopuszczać do błędu przepełnienia bufora.

Plik `events_example` zawiera przykładowy opis wydarzeń.

W celu ułatwienia testowania rozwiązań dostarczamy prostego klienta `ticket_client` w wersji binarnej skompilowanej na maszynie `students`. Przy czym nie gwarantujemy, że ten klient działa bezbłędnie.

Rozwiązanie

Rozwiązanie należy zaimplementować w języku C lub C++, korzystając z interfejsu gniazd. Rozwiązanie powinno być zawarte w pliku o nazwie `ticket_server.c` lub `ticket_server.cpp`. Plik należy złożyć w Moodle przed upływem podanego terminu. Rozwiązanie będzie kompilowane na maszynie students poleceniem:

```
gcc -Wall -Wextra -Wno-implicit-fallthrough -std=c17 -O2
```

lub

```
g++ -Wall -Wextra -Wno-implicit-fallthrough -std=c++17 -O2
```