Sprawozdanie z zadania na SK2

Zdalne wyłączanie komputerów

Sebastian Michoń

1 Protokół komunikacji

1. Klient porozumiewa się z serwerem z użyciem protokołu TCP - zależy mi na niezawodności przesłania uprawnień, a także przesłania na sprzęt klienta odpowiedniej komendy wyłączającej sprzęt

2. Klient po zainicjowaniu połączenia przesyła nasłuchującemu serwerowi - a ściślej wątkowi, który z tego serwera wychodzi bufor tekstu w następującym formacie:
   komenda
   uid
   gid
   wynik ls -lnH $(which shutdown)
   wynik ls -lnH $(which init)
   czas
   Gdzie:
   komenda - jedna cyfra - 0, jeśli shutdown, 1, jeśli reset
   uid, gid - user id, group id, którymi posługuje się aplikacja klienta
   2 kolejne linie są użyteczne do sprawdzenia uprawnień komend, którymi mogę wyłączyć system
   czas - za ile minut ten komputer ma zostać wyłączony? (w minutach)

3. Serwer po parsingu tych danych wysyła klientowi bufor tekstu, w którym znajduje się komenda, którą klient ma wykonać na systemie - w szczególności komentarz, jeśli klient jest pozbawiony odpowiednich praw.

2 Struktura serwera

1. Serwer tworzy socketa, manipuluje jego opcjami(setsockopt), wiąże tego socketa z portem 1234 i adresem, następnie zaczyna nasłuchiwać na tym sockecie do 5 połączeń i je akceptować - nawiązane połączenia obsługuje funkcja handleConnection.

2. Funkcja handleConnection przekazuje nowemu wątku wykonującemu funkcję ThreadBehavior deskryptor, z którym może się komunikować i wraca do maina.

3. Funkcja ThreadBehavior odbiera od klienta bufor tekstu, wysyła go do funkcji parsującej parse_wisdom, jej wynik (zamknięty w strukturze wisdom) jest przekazywany do funkcji grant_wisdom tworzącej stringa przesyłanego do klienta; następnie wychodzę z wątku.

4. Funkcja parse_wisdom operując na danych przesyłanych przez klienta zwraca informację, do jakich komend ma on uprawnienia a także jaki jest czas, za jaki komputer ma zostać wyłączony/zresetowany.

5. Funkcja grant_wisdom operując na wyniku funkcji parsującej zwraca komendę, która zostanie wysłana do klienta.

3 Struktura klienta

1. W aplikacji okienkowej można podać serwer i port, z którym chcę się skomunikować, ponadto czas, za jaki chcę zrestartować/wyłączyć komputer

2. Po kliknięciu na przyciski reset/shutdown uruchamia się trigger odpowiadający za przedsięwzięcie odpowiednich operacji po stronie klienta - zamraża on przyciski shutdown i reset, przechodząc do funkcji outer_processing. Przycisk Cancel zamyka okno i związane z nim operacje, ale

3. funkcja outer_processing wsadza dane od gui do struktury i wrzuca je do nowego wątku wykonującego funckję parse_connection, wychodząc z funkcji bez pthread_joina - dzięki temu aplikacja się nie wiesza.

4. funkcja parse_connection tworzy socketa, a następnie łączy się z podanym przez użytkownika adresem i portem. Dalej wywołuję funkcję zarządzającą połączeniem handleConnection i zamykam socketa.

5. Funkcja handleConnection zdobywa informacje w określonym formacie od funkcji attain_wisdom o użytkowniku i jego prawach, wysyła je do serwera i otrzymuje odpowiesź - polecenie do wykonania na komputerze, na którym uruchomiony jest klient.

6. Timer odczytuje co 1 sekundę informację o tym, czy thread się zakończył - jeśli tak, to wykonywane jest polecenie wysłane przez serwer, odblokowywane są przyciski.

4 Uruchomienie

1. Uruchomienie serwera:
   cd ...(miejsce, gdzie zaszyty jest projekt)
   gcc ser.c -o ser -Wall -l pthread
   ./ser

2. Uruchomienie klienta:
   cd .../QtServerClientApp (w miejscu, gdzie jest projekt)
   qmake
   make
   ./QtServerClientApp