Zdalna konsola typu *telnet*

Sprawozdanie z projektu na Sieci Komputerowe 2

Prowadzący: Michał Kalewski

Autor: Grzegorz Bryk 136686 i3
Zajęcia Poniedziałek 15:10

1 Opis protokołu

Protokół zastosowany w zadaniu jest bardzo prosty – klient przesyła serwerowi polecenia w formie tekstu zakodowanego do bajtów kodem ASCII. Serwer odpowiada tekstem zwróconym przez wykonanie polecania na standardowe wyjście, również zakodowanym ASCII, a transmisję kończy znakiem nowej linii (aby przy pustym wyjściu z programu mimo wszystko wykonać transmisję zwrotną do klienta, co da mu znak do wyczyszczenia pola z wyjściem).

2 Implementacja

Serwer został zaimplementowany w języku C, w zgodzie ze standardem C11. Współbieżność jest realizowana z pomocą osobnych procesów tworzonych dla każdego połączenia. Autor wyszedł z założenia, że zdalna konsola nie jest zastosowaniem które typowo przyjmuje wiele połączeń, a w standardowej implementacji i tak dla każdego połączenia tworzony jest proces powłoki, więc narzut spowodowany przez tę decyzję nie stanowi istotnego problemu, a w specyficznym przypadku w którym narzut połączenia stanowiłby większe obciążenie niż uruchamiane w zdalnej konsoli programy, stanowi to problem administracyjny, a nie implementacyjny.

Implementacja serwera znajduje się w katalogu *server*, w pliku *main.c*. kompilacja może być wykonana przez polecenia *make* lub *gcc main.c -o server*.

Klient został zrealizowany na system Android, w języku Kotlin. Język Kotlin zapewnia pełną interoperacyjność z Javą, w tym w razie potrzeby można wykonać automatyczne tłumaczenie kodu pomiędzy tymi dwoma językami.

Interfejs klienta dzieli się na dwa ekrany. Ekran powitalny z polami do wpisania adresu IP serwera i portu, oraz przyciskiem *Connect*, jest zrealizowany w plikach *Telnet/app/src/main/res/layout/content_main.xml* i *activity_main.xml* (layout) oraz *Telnet/app/src/main/java/com/gbryk/telnet/MainActivity.kt* (logika).

Naciśnięcie przycisku connect przenosi do drugiego ekranu, który pozwala na wysyłanie komend i wyświetlanie ich wyniku. Połączenie jest tworzone w tle (zgodnie z wymogami Androida). Ten ekran zrealizowany jest w plikach *Telnet/app/src/main/res/layout/activity_terminal.xml* (layout) i

Telnet/app/src/main/java/com/gbryk/telnet/TerminalActivity.kt (logika ekranu). Klasa odpowiedzialna za komunikację z serwerem i działająca jako osobny wątek w tle znajduje się w pliku *Telnet/app/src/main/java/com/gbryk/telnet/Telnet.kt.*

Interfejs drugiego ekranu dzieli się na umieszczone na górze pole do wpisywania komend wraz z przyciskiem *SEND*, oraz umieszczone poniżej pole tekstowe w którym pojawia się odpowiedź serwera.

Klasa *Telnet* wystawia interfejs do wysyłania poleceń, które są w obrębie tej akcji kolejkowane do wysłania przez główną pętlę wątku, oraz do odbioru ostatniego zwróconego tekstu odpowiedzi. Interfejs odświeża pole odpowiedzi za każdym razem kiedy otrzyma nową odpowiedź od klasy Telnet.

Operacje na kolejce poleceń oraz polu String zawierającego treść odpowiedzi serwera nie wymagają synchronizacji *explicite*, gdyż Kotlin realizuje ją wewnętrznie.

Aby skompilować i uruchomić klienta, należy otworzyć projekt *Telnet* w Android Studio, i uruchomić domyślny build Gradle.