Sieci Komputerowe 2021/22

Projekt zaliczeniowy III: Gra UDP

Wybierz ulubioną liczbę całkowitą n z zakresu $[5001, 20000]$.

Napisz, używając protokołu UDP IPv4, program umożliwiający grę w **kółko i krzyżyk** pomiędzy dwoma osobami na różnych maszynach. Ma to być jeden program (nie dwa programy typu klient-serwer)!

Specyfikacja:

(1) Komunikaty wysyłamy/odbieramy na tym samym porcie równym wybranej liczbie n.

(2) Pierwszy argument programu to adres domenowy **lub** adres IP hosta, z którym będziemy prowadzić rozgrywkę.

(3) Drugi, opcjonalny argument programu to nick użytkownika. Nick ten będzie się pojawiał lokalnie i zdalnie przy naszych komunikatach (patrz przykłady poniżej). Jeżeli argument nie zostanie podany program ma przyjąć słowo `NN` jako nick.

(4) Jednocześnie umożliwiamy użytkownikowi wpisanie/wysłanie komunikatu i nasłuchujemy komunikatów przychodzących.

(5) Wyświetlamy adres IP zdalnego komputera obok nicku rozmówcy (patrz przykłady).

(6) Wpisanie `<koniec>` informuje zdalnego gracza o zakończeniu gry i wyłącza aplikację (uprzednio „sprzątając" po sobie).

(7) Wpisanie `<wynik>` powoduje wypisanie wyniku rozgrywki pomiędzy graczami.

(8) Aplikacja informuje użytkownika o faktach dołączenia do gry i zakończenia gry przez zdalnego gracza (zgodnie z przykładami poniżej).

(9) **Interfejs oraz zasady rozgrywki są zdefiniowane poprzez przykład na końcu zadania.**

Rozwiązania będą testowane na maszynach wydziałowych: `ultra60`, `juliusz`, `aleks-2`, `vh0`, itd.

Należy zadbać o:

- obsługę błędów i inne „dobre praktyki" programistyczne opisane w sekcji *Jak przygotować dobry projekt zaliczeniowy?* na ostatniej stronie,
- **precyzyjne i estetycznie sformatowane komunikaty** informujące użytkownika na każdym etapie co się dzieje, czego program oczekuje, jakie opcje / skróty klawiaturowe są dostępne (patrz przykłady),
- obsługa polskich znaków przez program nie jest wymagana.

Film:

- należy przygotować i przesłać film z prezentacją programu, wskazówki jak przygotować film znajdują się na stronie wykładowcy,
- podkreślmy, że film powinien trwać 5-8 min. (filmy trwające więcej niż 10 min. nie będą akceptowane) i mieć formę screencastu z komentarzem głosowym,
- na filmie powinno znaleźć się:
  - omówienie kodu źródłowego (proszę nie omawiać każdej linii kodu ani rzeczy elementarnych, należy krótko skupić się na kilku wybranych najważniejszych fragmentach kodu / pomysłach na które (subiektywnie) warto zwrócić uwagę, np. fragmenty dotyczące wypełniania struktur adresowych, przesyłania i odbierania wiadomości itp.),
  - prezentacja kompilacji,
  - pokaz działania – główna część filmu.

Punktacja: (każdy z punktów powinien zostać zaprezentowany na filmie)

- podstawa: programy komunikują się ze sobą w dziedzinie internetu za pomocą protokołu UDP i umożliwia grę w kółko i krzyżyk: **4 punkty** [w tym 1,5 p. za poprawną reakcję na próbę wyboru pola poza swoją kolejką]
- kod programu: estetyka, obecność komentarzy, obsługa błędów przynajmniej w kluczowych miejscach, kompilacja obu kodów z opcją `-Wall` bez warningów: **1 punkt**,
- odpowiednia reakcja na wywołanie programu z błędnymi argumentami (na filmie przykład takiego wywołania): **1 punkt**,
- interfejs: estetyka i zgodność ze specyfikacją (w tym, poprawne wyświetlanie wyniku po komendzie `<wynik>` użytej **w dowolnym momencie działania gry** oraz odpowiednie zakończenie rozgrywki: informacja o zwycięstwie/porażce, pytanie o kolejną rozgrywkę): **1 punkt**,
- poprawne zamykanie programu (poprzez komendę `<koniec>` użytą **w dowolnym momencie działania gry**, na filmie należy zaprezentować, że po zakończeniu działania gry nie pozostają procesy sieroty ani zombie), wysyłanie (i odbieranie) odpowiedniego komunikatu do (od) współgracza: **1 punkt**,
- wypełnianie struktur adresowych za pomocą funkcji `getaddrinfo`: **2 punkty**.

Przesyłka:
Należy przesłać (w terminie, za pomocą mechanizmu Moodle poniżej) dwa pliki:

- kod gry (plik o nazwie `NazwiskoImie_gra.c`),
- film (plik o nazwie `NazwiskoImie_prezentacja.???`), gdzie w miejscu `NazwiskoImie` należy wpisać swoje nazwisko i imię.

Przykład:

- na maszynie `juliusz` uruchamiamy program: `./gra aleks-2 jasio`
- a na maszynie `aleks-2`: `./gra juliusz`

Uwaga: Na niebiesko zaznaczone są dane podawane przez użytkownika.

Przykładowe wykonanie (na maszynie `juliusz`):

```
[kanies@juliusz gra-FreeBSDbuild]> ./gra aleks-2 jasio
Rozpoczynam gre z 158.75.2.10. Napisz <koniec> by zakonczyc lub <wynik> by
wyswietlic aktualny wynik gry.
[Propozycja gry wyslana]
[NN (158.75.2.10) dolaczyl do gry]
a|b|c
d|e|f
g|h|i
[wybierz pole] e
<wynik>
Ty 0 : 0 NN
[NN (158.75.2.10) wybral pole a]
X|b|c
d|O|f
g|h|i
[wybierz pole] d
[NN (158.75.2.10) wybral pole h]
X|b|c
O|O|f
g|X|i
[wybierz pole] a
[tego pola nie mozesz wybrac, wybierz pole] f
[Wygrana! Kolejna rozgrywka, poczekaj na swoja kolej] <wynik>
Ty 1 : 0 NN
[NN (158.75.2.10) zakonczyl gre, mozesz poczekac na kolejnego gracza]
<koniec>
```

Odpowiadające mu wykonanie (na maszynie `aleks-2`):

```
[ola@aleks-2 gra-Linuxbuild]> ./gra juliusz
Rozpoczynam gre z 158.75.112.121. Napisz <koniec> by zakonczyc lub <wynik>
by wyswietlic aktualny wynik gry.
[Propozycja gry wyslana]
[Jasio (158.75.112.121) wybral pole e]
a|b|c
d|O|f
g|h|i
[wybierz pole] a
b
[teraz kolej drugiego gracza, poczekaj na swoja kolej]
[Jasio (158.75.112.121) wybral pole d]
X|b|c
O|O|f
g|h|i
[wybierz pole] h
[Jasio (158.75.112.121) wybral pole f]
X|b|c
O|O|O
g|X|i
[Przegrana! Zagraj jeszcze raz]
a|b|c
d|e|f
g|h|i
[wybierz pole] <koniec>
```

Jak przygotować dobry projekt zaliczeniowy?

Programistyczne zadanie „zaliczeniowe" zostanie ocenione na maksymalną liczbę punktów, jeśli spełnia wszystkie poniższe warunki:

- Program działa poprawnie (w pełni zgodnie ze specyfikacją) na serwerach dostępnych w sieci wydziałowej.
- Kod kompiluje się poprawnie z opcją `-Wall` (*print all warnings*) bez wyświetlania ostrzeżeń.
- Program jest napisany samodzielnie i autor rozumie użyte techniki.
- Kod jest napisany estetycznie i czytelnie (odstępy, wcięcia, czytelne nazwy zmiennych, itp.).
- Kod jest dobrze udokumentowany (komentarze opisujące najważniejsze fragmenty kodu i najważniejsze zmienne).
- Jest zaimplementowana obsługa błędów wywołania funkcji systemowych oraz odpowiednia reakcja na niepoprawne dane od użytkownika.
- Wyjściowe komunikaty wypisywane przez program są czytelne, zrozumiałe i estetyczne (w szczególności, komunikaty podają wszystkie informacje niezbędne dla użytkownika).
- Jeżeli kod źródłowy składa się z więcej niż jednego pliku lub wymaga dodatkowych („niestandardowych") opcji kompilacji, to wówczas należy dołączyć odpowiedni plik `Makefile`.
- Do plików źródłowych dołączony jest krótki (5-8 minutowy) film zawierający prezentację projektu.