MNM
TEAM
Munich Network Management Team

Ludwig-Maximilians-Universität München

Prof. Dr. D. Kranzlmüller
Dr. N. gentschen Felde
Jan Schmidt
Sophia Grundner-Culemann

Systempraktikum — Projektaufgabe (Teil 1 von 4)

Ihre Aufgabe für dieses Übungsblatt ist es, die erste Protokollphase („Prolog“) der Kommunikation mit dem Gameserver zu implementieren. In der Übersichtsgrafik des einleitenden Übungsblatts ist dies durch die TCP-Verbindung zum Gameserver des rechten Prozesses (*Connector*) dargestellt. Die genaue Protokolldefinition finden Sie ebenfalls in diesem Dokument.

Folgende Teilaufgaben sind unter anderem zu erledigen:

- Ihr Programm muss über `./sysprak-client` ausgeführt werden können.
- Ihr Programm muss eine 13-stellige Game-ID und die gewünschte Spielernummer als Kommandozeilenparameter (Hinweis: `argv` und `argc`) auslesen können: `./sysprak-client -g <GAME-ID> -p <{1,2}>`
- Definieren sie die drei Konstanten `GAMEKINDNAME` mit dem Wert `"Checkers"`, `PORTNUMBER` mit dem Wert `1357` und `HOSTNAME`, welche den Wert `"sysprak.priv.lab.nm.ifi.lmu.de"` erhält (Hinweis: `#define`).
- Anschließend verbinden Sie sich mit dem Gameserver (Hinweis: `socket()`, `gethostbyname()` / `getaddrinfo()` und `connect()`) und rufen die später von Ihnen zu implementierende Methode `performConnection()` auf, welche als Argument den File-Descriptor Ihres Sockets übergeben bekommt.
- Implementieren Sie nun die *Prolog*-Phase der Kommunikation in der Methode `performConnection()`. Geben Sie beim `PLAYER`-Kommando keine Werte mit und lassen Sie sich vom Gameserver einen freien Spieler zuweisen. Diese Methode sollte sich der besseren Übersichtlichkeit halber in der separaten Datei `performConnection.c` befinden. Sie können nach Belieben zusätzliche Methoden und Dateien erstellen, wenn diese Ihnen helfen.
- Geben Sie alle vom Gameserver erhaltenen Informationen wohl formatiert aus, d. h. nicht die Protokollzeile vom Gameserver, sondern z. B.: „Spieler 1 (Uli) ist noch nicht bereit“. Achten Sie hierbei darauf, dass Integer-Werte, wie z. B. die 1, auch als solche interpretiert werden.
- Achten Sie bei all Ihren Aufrufen auf eine ordentliche Fehlerbehandlung (Hinweis: `perror()` und das Kapitel „Return Value“ der Manpages), da Ihr Programm Fehler, wie z. B. ein nicht vorhandener Host oder ein nicht laufender Gameserver, erkennen und melden sollte.
- Testen Sie Ihren Client ausführlich mit dem Gameserver! Versuchen Sie einem (nicht) existierenden Spiel mit freien/besetzten Computergegnern beizutreten. Achten sie dabei darauf, dass Ihr Client die Fehlermeldungen des Gameservers richtig interpretiert und sich entsprechend verhält.
- Zum leichteren Übersetzen Ihres Programms erstellen Sie ein Makefile für Ihr Projekt. Das „default target“ soll dabei die einzelnen Quelldateien zu Objektdateien kompiliert, diese zu einer ausführbaren Datei linken und in einen direkt ausführbaren Zustand bringen. Zudem muss das Target `play` sicherstellen, dass ihr Client gebaut ist und ihn dann ausführen. Dabei soll der Client der Partie beitreten, deren Game-ID in der Umgebungsvariable `GAME_ID` zu finden ist, und dabei den Spieler aus der `PLAYER` Umgebungsvariable verwenden (Exemplarischer Aufruf: `GAME_ID=<GAME-ID> PLAYER=<{1,2}> make play`).

Achtung: Verwenden Sie für all Ihre Übersetzungen die gcc- / clang-Schalter `-Wall`, `-Wextra` und `-Werror`. Dies führt dazu, dass auch Kleinigkeiten als Warnung ausgegeben werden und der Compiler eine Warnung als einen Fehler ansieht und abbricht. Dies dient dem Zweck Ihnen eine spätere, lästige Fehlersuche zu ersparen, die wesentlich aufwändiger ist als die Warnungen frühzeitig zu beseitigen bzw. zu vermeiden.