Portfolioprüfung – Werkstück A – Alternative 5

Aufgabe 1 Aufgabe

Entwickeln und implementieren Sie ein Sudoku-Spiel mit Interprozesskommunikation, mit dem Benutzer in der Shell gegeneinander Sudoku spielen können.

Die Benutzer (mindestens zwei Spieler) sollen gegeneinander spielen können und der Spieler, der als erster das Sudoku gelöst hat, hat gewonnen.

Wikipedia enthält zahlreiche Informationen zu Sudoku[1] und den mathematischen Grundlagen[2].

Entwickeln und implementieren Sie Ihre Lösung als Bash-Skript oder als C-Programm als freie Software (Open Source) und verwenden Sie hierfür ein Code-Repository, z.B. bei GitHub.

Bearbeiten Sie die Aufgabe in Teams zu **5 Personen**.

Schreiben Sie eine aussagekräftige und ansehnliche Dokumentation (Umfang: **8-10 Seiten**) über Ihre Lösung.

Bereiten Sie einen Vortrag mit Präsentationsfolien und eine Live-Demonstration (Umfang: **15-20 Minuten**) vor. Demonstrieren Sie die Funktionalität der Lösung in der Übung.

Aufgabe 2 Anforderungen

- Das fertige Programm soll eine Kommandozeilenanwendung sein. Es soll möglich sein, das Spiel nur aus einer Shell heraus zu spielen. Es soll komplett in der Shell ablaufen!
- Der Quellcode soll durch Kommentare verständlich sein.
- Die Spieler (mindestens zwei Spieler) sollen gegeneinander spielen können.
- **Jeder Spieler ist ein eigener Prozess. Nutzen Sie zur Entwicklung und Implementierung beliebige Formen der Sychronisation von Prozessen, Interprozesskommunikation und Sychronisation von Prozessen wie Signalieren, Gemeinsamer Speicher, Nachrichtenwarteschlangen, Pipes, Sockets, Semaphoren, etc. Sie haben die freie Auswahl.**

[1] `https://de.wikipedia.org/wiki/Sudoku`
[2] `https://en.wikipedia.org/wiki/Mathematics_of_Sudoku`

- Ein 9x9 Felder großes (lösbares) Sudoku wird mit einigen vorgegeben Zahlen erzeugt und in der Shell angezeigt bzw. ausgegeben.
- Ob die Spieler, die gegeneinander spielen, alle das gleiche Sudoku lösen sollen, oder ob jeder Spieler ein eigenes Sudoku lösen soll ist Ihnen überlassen.
- Wenn Sie es nicht schaffen, einen Algorithmus zur finden bzw. für die Shell anzupassen, der lösbare Sudokus generiert, dann können Sie alternativ eine Lösung implementieren, bei der Sudokus aus einer oder mehreren Textdateien eingelesen werden (können). Das feste Einprogrammieren von Sudokus in den Quellcode ist keine zulässige Lösung!
- Die Benutzer können in die freien Felder Zahlen eingeben.
- Die Benutzer können selbst eingegebene Zahlen auch wieder löschen.
- Die Anwendung kontrolliert nach jeder Eingabe einer Zahl, ob das Sudoku-Feld konsistent ist. Ist das Feld nach der Eingabe nicht mehr konsistent, reagiert die Anwendung angemessen.
- Die Anwendung kontrolliert nach jedem Löschen einer Zahl, ob das Sudoku-Feld konsistent ist.
- Es soll zu jeder Zeit klar ersichtlich sein, welche Zahlen vorgegeben waren und welche Zahlen durch den Benutzer eingegeben wurden.
- Hat der Benutzer oder ein Gegenspieler das Sudoku erfolgreich gelöst, soll dieses bei allen Spielern angezeigt werden. Das kann beispielsweise durch eine Laufschrift geschehen, durch ein Blinken oder durch ein Invertieren der Farben in der Shell, etc.
- Sobald ein Sudoku geladen wurde, läuft eine Uhr, die anzeigt wie viel Zeit bislang vergangen ist. Hat der Benutzer das Sudoku gelöst, wird auch die vergangene Zeit ausgegeben.
- Für die „grafische Darstellung“ und Bedienung in der Shell können Sie eine Bibliothek wie **ncurses**[3] [4] (für C-Programme), **Termbox**[5] (für C-Programme oder Python-Scripte), **dialog**[6] [7] [8] (für Shell-Scripte) oder **Whiptail**[9] [10] [11] (für Shell-Scripte). Zwingend nötig ist das aber nicht.

[3] http://openbook.rheinwerk-verlag.de/linux_unix_programmierung/Kap13-002.htm
[4] https://de.wikibooks.org/wiki/Ncurses:_Grundlegendes
[5] https://github.com/nsf/termbox
[6] http://openbook.rheinwerk-verlag.de/shell_programmierung/shell_007_007.htm
[7] https://www.linux-community.de/ausgaben/linuxuser/2014/03/mehr-komfort/
[8] https://linuxkurs.spline.de/Ressources/Folien/Linux-Kurs-7.pdf
[9] https://en.wikibooks.org/wiki/Bash_Shell_Scripting/Whiptail
[10] https://saveriomiroddi.github.io/Shell-scripting-adventures-part-3/
[11] https://www.dev-insider.de/dialogboxen-mit-whiptail-erstellen-a-860990/

Aufgabe 3 Literatur

- Foliensätze 4 und 6 der Vorlesung **Betriebssysteme und Rechnernetze** im SS2022
- **Betriebssysteme kompakt**, *Christian Baun*, 2. Auflage, Springer Vieweg, S. 200-252
- **Betriebssysteme**, *Erich Ehses, Lutz Köhler, Petra Riemer, Horst Stenzel, Frank Victor*, 1. Auflage, Pearson (2005), S. 55-84
- **Betriebssysteme**, *Carsten Vogt*, 1. Auflage, Spektrum (2001), S. 109-127
- **Betriebssysteme**, *William Stallings*, 4. Auflage, Pearson (2003), S. 334-339