Portfolioprüfung – Werkstück A – Alternative 7

Aufgabe 1 Aufgabe

Entwickeln und implementieren Sie ein Schiffe-Versenken-Spiel mit Interprozesskommunikation, mit dem zwei Spieler in der Shell gegeneinander Sudoku spielen können.

Für mehr Informationen zum Spiel und der Spielweise, siehe Wikipedia[1].

Entwickeln und implementieren Sie Ihre Lösung als Bash-Skript, als Python-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.
- 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.**
- Die Anzahl der Felder in der Breite und Höhe der zu generierenden Spielfelder gibt der erste Spieler per Kommandozeilenargument an. Also z.B. `-xaxis 10 -yaxis 10`. Schränken Sie den Wertebereich sinnvoll ein.

[1] `https://de.wikipedia.org/wiki/Schiffe_versenken`

- Die Anzahl der Schiffe pro Spieler und deren Größen und Namen können Sie im Programm fest einprogrammieren oder alternativ per Kommandozeilenargumente vom Benutzer definieren lassen. Das ist ihre Entscheidung.
- Mit der Tastatur (und evtl. auch mit der Maus) wählt jeder Spieler zu Beginn des Spiels die Positionen seiner Schlachtschiffe aus. Eigene Boote dürfen sich beim Legen natürlich nicht überlappen.
- Der Spieler, der an der Reihe ist, definiert mit der Tastatur (und evtl. auch mit der Maus) auf welches Feld gefeuert werden soll. Alternativ soll es auch möglich sein, die Koordinaten mit der Tastatur einzugeben.
- Hat ein Spieler alle Schiffe des Gegners versenkt, hat er das Spiel gewonnen. Dieses soll entsprechend beiden Spielern angezeigt werden. Das kann beispielsweise durch eine Laufschrift geschehen, durch ein Blinken oder durch ein Invertieren der Farben in der Shell, etc.
- Der Spielstand, also die Anzahl der versenkten und noch übrigen Schiffe des aktiven Spielers und des Gegners, werden ständig über, unter oder neben dem Spielfeld des aktiven Spielers angezeigt. Es muss auch immer klar ersichtlich sein, welcher Spieler aktuell am Zug ist.
- Es soll jederzeit klar ersichtlich sein auf welche Felder man schon gefeuert hat und auf welche Felder der Gegner schon gefeuert hat.
- Für die „grafische Darstellung“ und Bedienung in der Shell können Sie eine Bibliothek wie **ncurses**[2] [3] (für C-Programme), **Termbox**[4] (für C-Programme oder Python-Scripte), **dialog**[5] [6] [7] (für Shell-Scripte) oder **Whiptail**[8] [9] [10] (für Shell-Scripte). Zwingend nötig ist das aber nicht.

Aufgabe 3 Optionale Anforderungen

- Es kann sinnvoll sein einen Countdown (z.B. 15 Sekunden) anzuzeigen, und wenn der menschliche Spieler in der Zeit kein Feld zum beschießen auswählt, feuert der PC via Zufallsfunktion. Sinnvoll wäre es auch wenn per Kommandozeilenargument definiert werden kann, ob solch ein Countdown aktiviert sein soll. Ob Sie die Countdown-Funktionalität implementieren ist ihre Entscheidung.

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

- Eventuell fallen Ihnen sinnvolle Erweiterungen ein. Um den Strategieaspekt zu erhöhen, wäre es z.B. sinnvoll, wenn die Spieler nicht nur sehen auf welche Feld ein Gegner gefeuert hat, sondern auch einen Hinweis bekommen, aus welcher Himmelsrichtung gefeuert wurde.

Aufgabe 4 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