Portfolioprüfung – Werkstück A – Alternative 3

1) Aufgabe

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

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

Buzzword-Bingo ist ein Klassiker der IT-Geschichte und hilft dabei, die übermäßige Verwendung inhaltsleerer Schlagwörter (häufig Anglizismen) in Meetings, Vorträgen bzw. Präsentationen sichtbar zu machen. Der Beschreibung von Buzzword-Bingo auf Wikipedia ist nichts hinzuzufügen:

> Buzzword-Bingo, in der späteren Verbreitung auch Bullshit-Bingo und Besprechungs-Bingo genannt, ist eine humoristische Variante des Bingo-Spiels, die die oft inhaltslose Verwendung zahlreicher Schlagwörter in Vorträgen, Präsentationen oder Besprechungen persifliert.
>
> Statt Bingokarten mit Zahlen werden Karten mit Schlagwörtern (engl. buzzwords) benutzt. Im Gegensatz zum originalen Bingo, bei welchem die zu streichenden Zahlen aus einer Lostrommel gezogen werden, werden Wörter gestrichen, wenn sie genannt werden. Bei einer vollständig gefüllten Reihe, Spalte oder Diagonale soll der Spieler den Regeln nach aufstehen und *Bingo* oder auch, um die Inhaltsleere der Wortphrasen hervorzuheben, *Bullshit* rufen. Mit dem Spiel und diesem Ausruf wird gleichzeitig die übermäßige Verwendung oft inhaltsleerer Schlagwörter kritisiert.
>
> Quelle: `http://de.wikipedia.org/wiki/Buzzword-Bingo`

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 oder GitLab (siehe Abschnitt 5).

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

Schreiben Sie eine aussagekräftige und ansehnliche **Dokumentation** (Umfang: **8-10 Seiten**) über Ihre Lösung. Eine Vorlage für das Textsatzsystem LaTeX und weitere Informationen zum Verfassen einer guten Dokumentation finden Sie auf der Vorlesungsseite. Sie müssen die Vorlage nicht zwingend verwenden und Sie müssen auch nicht zwingend LaTeX verwenden. Dieses Projekt ist aber eine sehr gute Gelegenheit sich mit LaTeX auseinander zu setzen und die Vorlage enthält viele hilfreiche Hinweise. In der Dokumentation beschreiben Sie Ihre Lösung (Architektur, etc.), die Aufteilung der Komponenten auf die im Team beteiligten Personen, ausgewählte

Probleme und deren Lösungen, etc. Auch einzelne Screenshots können sinnvoll sein. Die Dokumentation soll ein technischer Bericht sein, in der dritten Person formuliert sein, und ganz auf relevante Aspekte Ihrer Lösung fokussieren.

Bereiten Sie einen **Vortrag mit Präsentationsfolien und Live-Demonstration** (Umfang: **15-20 Minuten**) vor. Demonstrieren Sie die Funktionalität der Lösung in der Übung. Eine Vorlage für Präsentationsfolien für das Textsatzsystem LaTeX mit der Erweiterungsklasse `beamer` und weitere Informationen zum Verfassen einer guten Projektpräsentation finden Sie auf der Vorlesungsseite. Auch diese Vorlage müssen Sie nicht zwingend verwenden und Sie müssen auch für die Präsentationsfolien nicht zwingend LaTeX verwenden. Dieses Projekt ist aber eine sehr gute Gelegenheit sich mit LaTeX und der Erweiterungsklasse `beamer` auseinander zu setzen und die Vorlage enthält viele hilfreiche Hinweise. Der Vortrag fokussiert ganz auf Ihre Lösung und nicht auf die Aufgabe oder Inhalte der Vorlesung. Es geht hierbei ausschließlich um Ihre Leistung im Projekt. Es versteht sich von selbst, dass alle im Team beteiligten Personen, ihre selbst entwickelten und implementierten Komponenten vorstellen und demonstrieren können und natürlich auch Fragen zur Lösung und zu ihren Komponenten beantworten können.

2) Anforderungen an den Simulator

- Das fertige Programm soll eine Kommandozeilenanwendung sein.
- Der Quellcode soll durch Kommentare verständlich sein.
- Die Spieler (mindestens zwei Spieler) sollen auf einem Computer gegeneinander spielen können.
- **Jeder Spieler ist ein eigener Prozess. Nutzen Sie zur Interprozesskommunikation entweder Nachrichtenwarteschlangen (Message Queues nach den Standards System V oder POSIX), Anonyme Pipes und/oder Benannte Pipes.**
- Jeder Spieler, der am Spiel teilnimmt, bekommt vom Spiel eine eigene Bingokarte in der Shell generiert.
- Die Anzahl der Felder in der Breite und Höhe der zu generierenden Bingokarten legt der Benutzer, der das Spiel eröffnet, per Kommandozeilenargument fest. Also z.B. `-xaxis 5 -yaxis 5`
- Die Werte der Felder auf der Bingokarte sollen aus einer Textdatei eingelesen und per Zufall verteilt werden. Konkret soll der Benutzer, der das Spiel eröffnet, die Möglichkeit haben, per Kommandozeilenargument den Pfad und Dateinamen der Textdatei zu definieren, also z.B. `-wordfile <dateiname>`.
- Die Textdatei kann auch gerne mehr Wörter enthalten und es werden entsprechend so viele Wörter zufällig vom Buzzword-Bingo-Spiel aus der Datei

importiert und auf der bzw. den Bingokarte(n) verteilt, wie es die definierte Höhe und Breite vorgibt.

- Die Spieler wählen mit der Tastatur (und evtl. auch mit der Maus – das ist aber nicht zwingend) einzelne Felder aus, um diese zu streichen (bzw. zu markieren). Alternativ ist eine Auswahl der Felder durch Eingabe der Koordinaten mit der Tastatur auch denkbar. Verwenden Sie dafür eine geeignete Bibliothek oder eine vergleichbare Lösung, die es ermöglicht, grafisch ansehnliche Ausgaben auf der Kommandozeile zu realisieren. Beispiele für Bibliotheken, die „grafische Darstellung“ und Bedienung in der Shell ermöglichen, sind **ncurses**[1] [2] (für C-Programme), **Termbox**[3] (für C-Programme oder Python-Scripte - wird nicht mehr weiterentwickelt), **Textual**[4] (für Python-Scripte), **Typer**[5] (für Python-Scripte), **Asciimatics**[6] (für Python-Scripte), **pyTermTk**[7] (für Python-Scripte), **dialog**[8] [9] [10] (für Shell-Scripte) oder **Whiptail**[11] [12] [13] (für Shell-Scripte).

- Fehler bei der Bedienung (z.B. fehlerhaftes Streichen eines Feldes sollen die einzelnen Benutzer auch rückgängig machen können.

- Fehlerhafte Kommandozeilenargumente soll das Programm erkennen und durch Fehlermeldungen und/oder Abbruch des Programms sinnvoll behandeln können.

- Hat ein Spieler oder ein Gegenspieler eine komplette Spalte, Zeile oder Diagonale seiner Bingokarte an Feldern gestrichen bzw. markiert, gilt das Spiel als gewonnen, was bei allen Spielern angezeigt wird. Das kann beispielsweise durch eine Laufschrift geschehen, durch ein Blinken oder durch ein Invertieren der Farben in der Shell, etc.

- Bei 5x5 oder 7x7 Feldern bleibt das Feld in der Mitte üblicherweise frei (*Joker*).

- Zur Dokumentation des Spiels soll das Programm für jede generierte Bingokarte eine Logdatei mit folgendem Dateinamen anlegen:

  `YYYY-MM-DD-HH-MM-SS-bingo-SpielerNummer.txt`

  Die Felder stehen stellvertretend für Jahr (`YYYY`), Monat (`MM`), Tag (`DD`), Stunde (`HH`), Monat (`MM`) und Tag (`SS`) und müssen die jeweils aktuell gültigen Werte

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

enthalten. Die Logdatei soll während des Spiels mit sinnvollen Daten (Zeilen) befüllt werden. Sinnvolle Beispiele sind:

  - `YYYY-MM-DD-HH-MM-SS Start des Spiels`
  - `YYYY-MM-DD-HH-MM-SS Größe des Spielfelds: (X-Achse/Y-Achse)`
  - Eine Zeile für jedes gestrichene bzw. markierte Wort (in der Reihenfolge, in der es gestrichen bzw. markiert wurde) inklusive der Koordinaten des Worts auf dem Spielfeld in folgender Struktur:
    `YYYY-MM-DD-HH-MM-SS Wort (X-Achse/Y-Achse)`
  - `YYYY-MM-DD-HH-MM-SS Sieg` oder `Abbruch`
  - `YYYY-MM-DD-HH-MM-SS Ende des Spiels`

- Ein Spieler definiert und eröffnet eine Partie bzw. eine Bingo-Runde, und die anderen Benutzer auf dem gleichen Computer können beitreten. Sie haben die freie Wahl, wie sie die Propagierung bzw. Definition von Spielrunden und den Beitritt zu einer Partie bzw. zu einem Spiel realisieren. Dieses ist zum Beispiel über eine (als Kommandozeilenargument) definierbare lokale Datei möglich, die einen Namen für die Spielrunde und die beteiligten Prozess-ID-Nummern (PIDs) enthält.

3) Beispielhafte Befehle

Beispiel für einen Start des Buzzword-Bingo-Spiels und die Eröffnung einer neun Bingo-Runde:

```
$ bingo.sh -newround \
           -roundfile rundendatei.txt \
           -xaxis 5 -yaxis 5 \
           -wordfile wortdatei.txt \
           -maxplayers 10
           -player name
```

Beispiel für einen Start des Buzzword-Bingo-Spiels und den Beitritt zu einer bestehenden Runde, die in einer lokalen Datei definiert ist:

```
$ bingo.sh -joinround \
           -roundfile rundendatei.txt \
           -player name
```

4) Einige Buzzwords zur Inspiration

- Synergie
- Rating
- Wertschöpfend
- Benefits
- Ergebnisorientiert
- Nachhaltig
- Hut aufhaben
- Visionen
- Zielführend
- Global Player
- Rund sein
- Szenario
- Diversity
- Corporate Identitiy
- Fokussieren
- Impact
- Target
- Benchmark
- Herausforderung(en)/Challenges
- Gadget
- Synergie
- Value
- Smart
- Web 2.0 oder 3.0
- Qualität
- Big Picture
- Revolution
- Pro-aktiv
- Game-changing
- Blog
- Community
- Social Media
- SOA
- Skalierbar
- Return on Invest (ROI)
- Wissenstransfer
- Best Practice
- Positionierung/Positionieren
- Committen
- Geforwarded
- Transparent
- Open Innovation
- Out-of-the-box
- Dissemination
- Blockchain
- Skills
- Gap
- Follower
- Win-Win
- Kernkompetenz

5) Einige abschließende Worte zum Code-Repository

Das Code-Repository müssen alle am Team beteiligten Personen erkennbar verwenden, das heißt, sie müssen ihre Komponente(n) aktiv entwickeln. Die aktive Mitarbeit bei der Entwicklung und Implementierung des Programms muss für alle Teammitglieder in der Commit-Historie des Code-Repositories erkennbar sein. Kommen einzelne am Team beteiligten Personen nicht in der Commit-Historie des Code-Repositories vor, ist deren Beitrag bei der Implementierung des Programms mehr als unglaubwürdig.

Das Code-Repository soll während der gesamten Projektlaufzeit einsehbar und die kontinuierliche Entwicklung des Simulators erkennbar sein. Stellen Sie hierfür Ihr Repository auf Public und nicht auf Private. Ein oder nur sehr wenige Commits einer einzelnen Person am Ende der Projektlaufzeit sprechen gegen eine gemeinsame Entwicklung und sinnvolle Teamarbeit.

6) Literatur

- Foliensätze 4 und 6 der Vorlesung **Betriebssysteme und Rechnernetze** im SS2024
- **Betriebssysteme kompakt**, *Christian Baun*, 3. Auflage, Springer Vieweg, Kapitel 9 „Interprozesskommunikation“
- **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