Portfolioprüfung – Werkstück A – Alternative 6

1) Aufgabe

Entwickeln und implementieren Sie vier (bzgl. der verwendeten Interprozesskommunikation) verschiedene Varianten eines Echtzeitsystems, das aus vier Prozessen besteht:

1. **Conv**. Dieser Prozess liest Messwerte von A/D-Konvertern (Analog/Digital) ein. Er prüft die Messwerte auf Plausibilität und konvertiert sie gegebenenfalls. Da keine physischen A/D-Konverter vorliegen, soll Conv Zufallszahlen erzeugen.

2. **Log**. Dieser Prozess schreibt die Messwerte von Conv in eine lokale Datei.

3. **Stat**. Dieser Prozess berechnet aus den Messwerten von Conv statistische Daten (Mittelwert und Summe).

4. **Report**. Dieser Prozess gibt die statistischen Daten von Stat in der Shell aus.

Beachten Sie bei der Implementierung folgende Synchronisationsbedingungen:

- **Conv** muss erst Messwerte erstellen, bevor **Log** und **Stat** diese nutzen können.
- **Stat** muss erst Statistikdaten erstellen, bevor **Report** diese nutzen kann.

Entwickeln und implementieren Sie das geforderte System mit den entsprechenden Systemaufrufen oder (Standard-)Bibliotheksfunktionen und realisieren Sie den Datenaustausch in den vier verschiedenen Varianten zwischen den vier Prozessen einmal mit

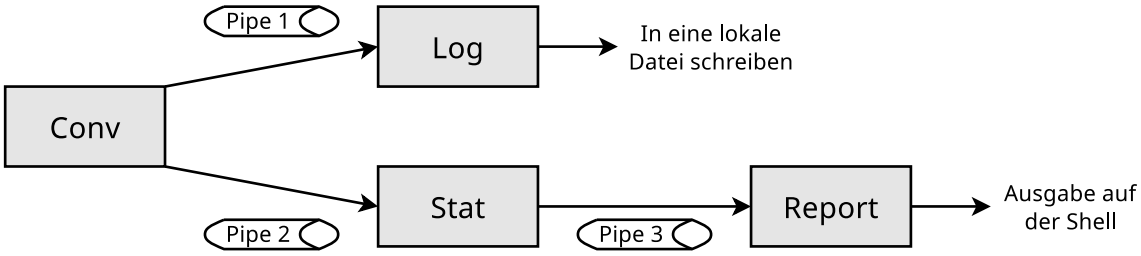
- **Benannten Pipes**,
- **Message Queues** (via POSIX API),
- **Shared Memory mit Semaphore** (via POSIX API) und mit
- **TCP-Sockets**.

Entwickeln und implementieren Sie Ihre Lösung 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 3).

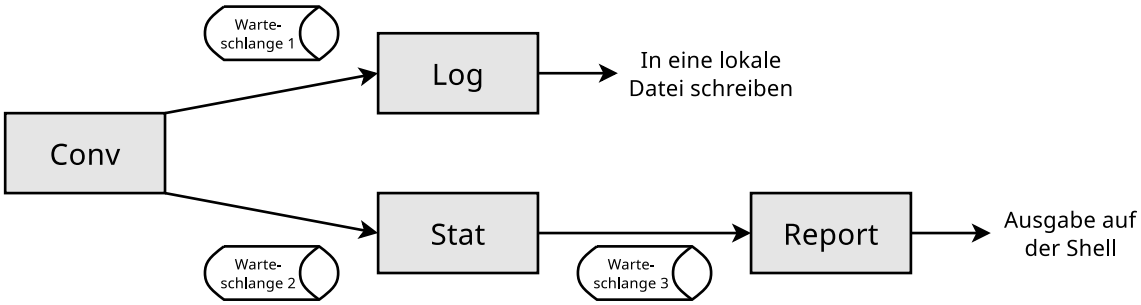
Bearbeiten Sie die Aufgabe in Teams zu **4 Personen**.

Als gefordertes Ergebnis müssen **vier Implementierungsvarianten des Programms** existieren, bei denen der Datenaustausch zwischen den vier Prozessen einmal mit **Benannten Pipes**, **Message Queues** (via POSIX API), **Shared Memory mit Semaphore** (via POSIX API) und via **TCP-Sockets** funktioniert.

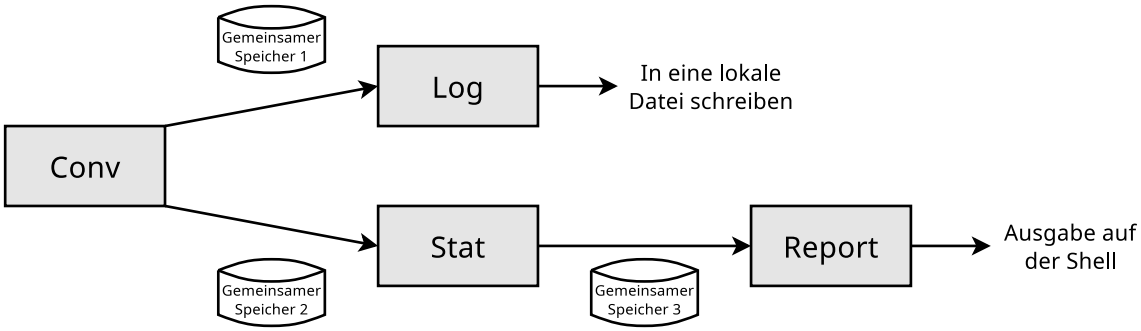
Mögliches Konzept für die Implementierungsvariante mit Pipes



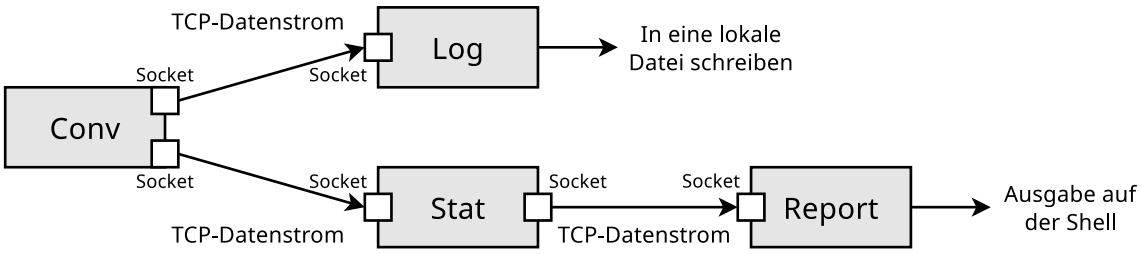
Mögliches Konzept für die Implementierungsvariante mit Message Queues



Mögliches Konzept für die Implementierungsvariante mit Shared Memory



Mögliches Konzept für die Implementierungsvariante mit Sockets



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 Prozesse Conv, Log, Stat, und Report sind parallele Endlosprozesse. Schreiben Sie ein Gerüst zum Start der Endlosprozesse mit dem Systemaufruf `fork`. Überwachen Sie mit geeigneten Kommandos wie `top`, `ps` und `pstree` Ihre parallelen Prozesse und stellen Sie die Elternbeziehungen fest.
- Das Programm kann mit der Tastenkombination `Ctrl-C` abgebrochen werden. Dazu müssen Sie einen Signalhandler für das Signal `SIGINT` implementieren. Beachten Sie bitte, dass beim Abbruch des Programms alle von den Prozessen belegten Betriebsmittel (Pipes, Message Queues, gemeinsame Speicherbereiche, Semaphoren) freigegeben werden.

- Fehlerhafte Kommandozeilenargumente soll das Programm erkennen und durch Fehlermeldungen und/oder Abbruch des Programms sinnvoll behandeln können.
- Die Kommandos `ipcs` und `ipcrm` sind bei Interprozesskommunikation via POSIX API nutzlos, da sie nur mit der System V API kompatibel sind. Zur Überwachung und eventuellen manuellen Entfernung von Message Queues und Shared Memory-Bereichen nutzen Sie die Einträge in den Verzeichnissen `/dev/mqueue` und `/dev/shm`.

3) 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.

4) Literatur

- Foliensätze 4 und 6 der Vorlesung **Betriebssysteme und Rechnernetze** im SS2024
- **Betriebssysteme kompakt**, *Christian Baun*, 3. Auflage, Springer Vieweg (2022), Kapitel 9
- **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