Portfolioprüfung – Werkstück A – Alternative 1

1 Aufgabe

Entwickeln Sie ein Echtzeitsystem, 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 liest die Messwerte von Conv aus und schreibt sie in eine lokale Datei.

3. **Stat**. Dieser Prozess liest die Messwerte von Conv aus und berechnet statistische Daten (Mittelwert und Summe).

4. **Report**. Dieser Prozess greift auf die Ergebnisse von Stat zu und gibt die statistischen Daten in der Shell aus.

Beachten Sie bei der Implementierung folgende Synchronisationsbedingungen:

- **Conv** muss erst Messwerte schreiben, bevor **Log** und **Stat** diese auslesen können.

- **Stat** muss erst Statistikdaten schreiben, bevor **Report** diese auslesen kann.

Entwickeln und implementieren Sie das geforderte System mit den entsprechenden Systemaufrufen oder (Standard-)Bibliotheksfunktionen und realisieren Sie den Datenaustausch zwischen den vier Prozessen einmal mit **Pipes**, **Message Queues**, **Shared Memory mit Semaphore** und mit **Sockets**.

Als gefordertes Ergebnis müssen **vier Implementierungsvarianten des Programms** existieren.

2 Anforderungen

- 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.

- Entwickeln und implementieren Sie die vier Varianten, bei denen der Datenaustausch zwischen den vier Prozessen einmal mit **Pipes**, **Message Queues**, **Shared Memory mit Semaphore** und via **Sockets** funktioniert.
- Überwachen Sie die Message Queues, Shared Memory-Bereiche und Semaphoren mit dem Kommando `ipcs`. Mit `ipcrm` können Sie Message Queues, Shared Memory-Bereiche und Semaphoren wieder freigeben, wenn Ihr Programm dieses bei einer inkorrekten Beendigung versäumt hat.
- **Entwickeln und implementieren Sie Ihre Lösung als Bash-Skript[1], als C-Programm oder als Python-Skript** als freie Software (Open Source) und verwenden Sie hierfür ein Code-Repository, z.B. bei GitHub.
- Ihre Anwendung soll eine Kommandozeilenanwendung sein.
- Der Quellcode soll durch Kommentare verständlich sein.
- Bearbeiten Sie die Aufgabe in Teams zu **3 Personen**.
- Schreiben Sie eine aussagekräftige und ansehnliche Dokumentation (Umfang: **8-10 Seiten**) über Ihre Lösung.
- Die Funktionalität der Lösung müssen Sie in der Übung demonstrieren. Bereiten Sie einen Vortrag mit Präsentationsfolien und eine Live-Demonstration (Umfang: **15-20 Minuten**) vor.

3 Literatur

- Foliensätze 4 und 6 der Vorlesung **Betriebssysteme und Rechnernetze** im SS2021
- **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

[1] Bei dieser Alternative ist die Implementierung als Bash-Skript die schwierigste Alternative. Einfacher ist es mit C oder Python.