Werkstück A

im Modul

Betriebssysteme und Rechnernetze

im Studiengang Wirtschaftsinformatik

Lösungsskizze

Alternative 2

vorgelegt von:

Tristan Buls 1440643

Yannis Körner 1432965

Daniel Schor 1435234

Benjamin Macholz 1459728

Prüfer: Christian Baun

Abgabetermin: 30.04.2023

Inhaltsverzeichnis

Methodisches Vorgehen

Vorbereitung

Jeder von uns hat die erforderlichen Schritte unternommen, um seine Arbeitsumgebung einzurichten.
Zunächst haben wir Oracle VM installiert und Ubuntu innerhalb der VM eingerichtet. Anschließend haben wir erfolgreich Visual Studio Code und Git innerhalb von Ubuntu heruntergeladen, eingerichtet und für die C++ Programmierung vorbereitet. Jeder von uns hat zudem ein eigenes GitHub-Konto erstellt und wir haben gemeinsam ein Repository angelegt. Schließlich haben wir das gemeinsame Arbeiten mit GitHub und Liveshare (VS-Code) erfolgreich getestet.

Somit haben wir die Einrichtung abgeschlossen und begonnen, die Programmiersprache C++ zu erlernen. Wir haben festgestellt, dass Java und C++ syntaktisch sehr ähnlich sind (was natürlich Sinn ergibt, da Java von C++ abstammt). Da wir im letzten Semester OOP mit Java gelernt haben, fiel uns das Erlernen der Syntax von C++ nicht sehr schwer.

Vorgehen

Beim Programmieren werden wir ab und zu gleichzeitig mit der VS Code Erweiterung Live Share arbeiten. Dabei wird der commit auf GitHub nur von einer Person von der Live Share Session erstellt, jedoch steht in der commit Beschreibung wer daran mitgearbeitet hat.

Wir werden versuchen möglichst viele Kommentare zu erstellen, damit jeder zu jeder Zeit versteht, worum es sich bei den jeweiligen Codeblocks handelt.

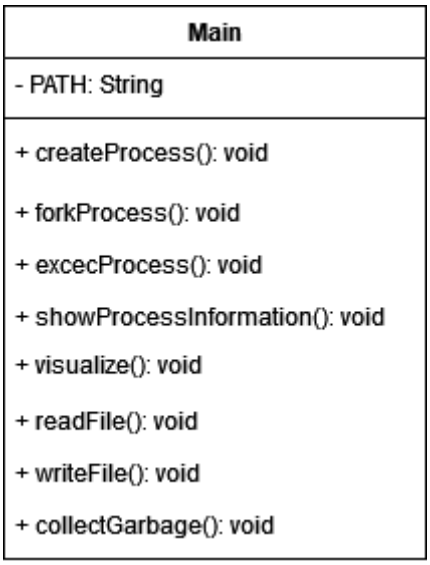
Wenn jemand einen Teil der Aufgabe implementiert und getestet hat, wird dieser Code commited. Das gesamte Team überprüft die Funktionalität und bespricht diesen commit. Somit kennt jeder den gesamten Code – Das Einbinden/Bug fixen fremden Codes fällt leichter.

Lösungsansatz des Problems

Für die Implementierung des Codes mit C++ haben wir uns folgende Schritte vorgestellt.

1. Tests in C++
    a. Simple Kommandozeilenanwendung (CLI)
    b. Zugriff/Manipulation von Dateien
    c. Freigabe von Systemressourcen
2. Beschäftigung mit
    a. /proc
    b. /proc/$pid/statm
3. Zugriff auf 2. Innerhalb von C++
    a. Anzeigen der Prozesse
        i. Strukturierung in Terminal
        ii. Livedarstellung
    b. Abspeichern der Prozesse
        i. (Statistiken erstellen)
        ii. ((Auslesen der gespeicherten Prozesse ermöglichen))
    c. Lesen der Dokumentation zu den Systemaufrufen fork() und exec()
    d. Erstellen/Verzweigen von Prozessen mit exec()/fork()
4. Zusammenfassung
    a. Implementierung der nötigen Methoden
        i. Lesen/Schreiben von Dateien
        ii. Create/Fork Prozess
        iii. Visualisierung
        iv. Freigabe Systemressourcen
    b. Dokumentation

Skizzen zum Lösungsansatz



Die Variable PATH speichert den Ort für die Datei, die in writeFile() erstellt wird und in readFile() gelesen wird.

Die Methode createProcess() erzeugt ein Prozess.

Die Methode forkProcess() erstellt ein Duplikat von dem erstellten Prozess aus createProcess().

Die Methode execProcess() ersetzt den Prozess aus createProcess().

Die Methode showProcessInformation() gibt die zentralen Informationen der erzeugten Prozesse aus.

Die Methode visualize() erstellt textuell eine Baumstruktur in der Konsole und zeigt die Beziehung zwischen exec() und fork().

Die Methode writeFile() erzeugt eine Datei und speichert die zentralen Informationen der erzeugten Informationen aus der Methode showProcessInformation() in einer maschinenlesbaren Sprache.

Die Methode readFile() liest eine Datei ein und zeigt die geschriebenen Informationen in der Konsole aus.

Die Methode collectGarbage() gibt die Betriebssystemressourcen frei vor beenden des Programms. Sprich beendet die erstellten Prozesse und gibt somit die Ressourcen frei.