Aufgabenblatt 4

Aufgabe - Implementierung einer Mini-Shell

Im Praktikum haben wir bisher die Shell als Benutzerschnittstelle des Betriebssystems genutzt: der Nutzer kann Kommandos eingeben, die vom Betriebssystem ausgeführt werden. Die Shell hat somit die Funktion eines Kommando-Interpreters. Die verschiedenen Shell-Varianten in UNIX übernehmen dazu weitere Funktionen: Starten und Stoppen von Prozessen, Handhabung von Platzhaltern (Wildcards), Handhabung von Umgebungs-Variablen, Verkettung von Prozessen, Ausführung von Skripten bestehend aus Shell-Kommandos.

Ein (kleiner) Teil dieser Aufgaben soll im Rahmen des Praktikums realisiert werden.

Aufgabenstellung:

Implementieren Sie eine ‚Mini-Shell' als C- oder C++-Programm, das gewisse Aufgaben der Shell umsetzt. Die von Ihnen implementierte Shell soll mindestens die folgenden Anforderungen erfüllen:

1. Das Programm gibt einen Prompt bestehend aus User-Name und aktuellem Directory aus und liest eine Eingabezeile von der Standardeingabe (**`stdin`**). Die Eingabezeile wird formatiert (mehrfache Leerzeichen eliminieren, Umgebungsvariablen ersetzen) und verarbeitet.

2. Wurde ein UNIX-Kommando eingegeben, so erzeugt das Programm per **`fork()`** einen Kindprozess.

3. Das Programm startet im Kindprozess per **`exec…()`** Funktion das aus der Benutzereingabe extrahierte Kommando.

4. Der Elternprozess wartet in der Zwischenzeit mit **`waitpid()`** auf das Ende des Kindprozesses.

Randbedingungen:

- Die Kommandos **`exit`**, **`cd`** und sowie **`showenv`**[1] und **`export …="…"`** (Abfragen und Setzen von *Umgebungsvariablen* analog zur Bash) sind keine UNIX-Befehle, sondern sollen als sog. *Built-in-Kommandos* direkt durch die Shell ausgeführt werden. Warum ist dies notwendig? Die Built-in-Kommandos müssen keine weiteren Optionen unterstützen.

- Kommandos können unter Umständen auch Variablen verändern. Welche sind dies?

- (Umgebungs-)Variablen in der Eingabezeile sollen aufgelöst werden. Beachten Sie auch, dass Variablen mehrfach in einem Kommando auftreten können.

[1] Das Kommando **`showenv`** ist kein Built-In Kommando der Bash. Orientieren Sie sich bei der Funktion an der des Befehls **`printenv`**. Hilfe zu den Built-In Kommandos können Sie mit **`help <cmd>`** erhalten.

- *Freiwillige Zusatzaufgabe*: implementieren Sie die Verwaltung von *lokalen Variablen*, d. h. die Zuweisung von Variablen mit **…="…"** und den Zugriff auf diese lokalen Variablen in Kommandos.

Hinweise:

- Anhand des Prompts sollte sofort erkennbar sein, ob Ihre Shell oder die Standard-Shell ausgeführt wird.
- Kommandos werden in der Regel mit Parametern aufgerufen. Sie müssen also einen rudimentären Interpreter implementieren. Zum Auswerten der Eingaben benötigen Sie die Standard-C-Funktionen zur Verarbeitung von Zeichenketten (z. B. **`sscanf()`**, **`strtok()`**,…) oder C++-Methoden (auf der Klasse **`std::string`**).
- Für die Behandlung von Dateinamen als Argumente ist die Funktion **`realpath()`** nützlich.
- Die Auflösung von Umgebungsvariablen bei normalen und Built-in-Kommandos kann über die Funktion **`getenv()`** erfolgen. Zum Setzen von Umgebungsvariablen kann **`setenv()`**verwendet werden.
- Ihre Shell wird in der nachfolgenden Aufgabe weiterentwickelt! Achten Sie auf ein sauberes Design (Verwendung von Funktionen oder Methoden) der Software, so dass Erweiterungen möglich sind.
- Achten Sie auch auf das Abfangen von Fehlern speziell bei der Prozesserzeugung und beim Aufruf von **`exec…()`**, da es ansonsten zu interessanten Phänomenen kommen kann.

Für das Testat ist ein **Protokoll** (Formatierung: siehe Blatt 1) des erstellten Programms (denken Sie an eine vernünftige Formatierung und hinreichende Kommentierung des Source-Codes) inklusive der durchgeführten Tests vorzulegen.