Implementierung Shell (10 Punkte)

Im Praktikum haben wir die Shell als Benutzerschnittstelle von UNIX verwendet. 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 Teil dieser Aufgaben soll im Rahmen des Praktikums realisiert werden.

Aufgabenstellung

Implementieren Sie eine „Minden-Mini-Shell“ als C - Programm, das gewisse Aufgaben der Shell übernimmt. 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)` (2 Punkte)

2. Das Programm erzeugt per `fork()` einen neuen Kindprozess (2 Punkte)

3. Das Programm startet per Funktion `execlp()` oder per `execvp()` das aus der Benutzereingabe extrahierte Programm im Kindprozess (3 Punkte)

4. Der Elternprozess wartet in der Zwischenzeit mit `waitpid()` auf das Ende des Kindprozesses. (3 Punkte)

Randbedingungen:

- Die Kommandos `exit`, `cd` und `set` sind keine externen Programme, sondern werden direkt von der Shell bearbeitet (Built-in-Kommandos). Warum ist dies notwendig?
- Kommandos können auch Parameter enthalten, Sie müssen also einen rudimentären Parser implementieren
- Umgebungsvariablen in der Parameterliste sollen aufgelöst werden
- Beachten Sie auch, dass Umgebungsvariablen mehrfach in einem Kommando auftreten können

Hinweise:

- Für die Behandlung von Dateinamen als Argumente ist die Funktion `realpath()` nützlich.

Tim Wächter
tim.waechter@web.de

- Die Auflösung von Umgebungsvariablen kann entweder mit dem dritten Parameter, der main übergeben wird, oder über die Funktion `getenv()` erfolgen. Zum Setzen von Umgebungsvariablen kann `putenv()` verwendet werden
- Zum Einlesen und Auswerten der Eingaben benötigen Sie die Standard-C-Funktionen zur Verarbeitung von Zeichenketten (z.B. `sscanf()`, `strtok()`, etc.)
- Ihre Shell wird in späteren Aufgaben noch benutzt! Achten Sie auf ein sauberes Design (Verwendung von Funktionen), so dass Erweiterungen möglich sind. Achten Sie auch auf das Abfangen von Fehlern speziell bei der Prozesserzeugung da es ansonsten zu nur schwer nachvollziehbaren Phänomenen kommen kann

Tim Wächter
tim.waechter@web.de