Aufgabenblatt 5

Aufgabe - Erweiterung der Shell um Pipes

Die im Rahmen der letzten Übung realisierte rudimentäre Shell soll nun erweitert werden.

Aufgabenstellung:

1. Implementieren Sie in der Shell aus der Aufgabe zuvor eine Pipe-Funktionalität.

   D. h. in der Kommandozeile sollen dann auch zwei durch eine Pipe verbundene Kommandos eingegeben werden können, wobei der erste Prozess die Eingabe für den zweiten liefert. Gekennzeichnet wird eine solche Pipe durch das auch von der bash her bekannte Symbol ‚|'.

   Beispiel: **`ls –als | more`**

   Die Ausgabe von **`ls -als`** dient hier als Eingabedatenstrom von **`more`**.

   Alle bereits implementierten Funktionen der Shell sollen beibehalten werden. Achten Sie bei der Realisierung auf eine adäquate Maskierung von Fehlern!

2. Experimentieren Sie mit Ihrer erweiterten Shell und vergleichen Sie das Verhalten mit dem der Bash. Finden Sie Antworten auf die folgenden Fragen:

   a. Was passiert, wenn der erste Prozess mehr Daten produziert als der zweite Prozess konsumieren kann?

   b. Was passiert, wenn sich keine Daten in der Pipe befinden?

   c. Was passiert, wenn einer der Prozesse in der Pipe terminiert oder beendet wird?

   d. Was passiert, wenn der Eltern-Prozess terminiert?

   e. Wie kann die Richtung des Datentransfers zwischen den Prozessen verändert werden?

   Für die Beantwortung der Fragen sollten Sie sich Pipe-Kommandos überlegen, mit denen Sie die in den Fragen beschriebenen Situationen herbeiführen können. Dokumentieren Sie diese im Protokoll. Falls Ihre Shell-Implementierung die benötigten Funktionalitäten nicht bereitstellt, so führen Sie die Untersuchungen mit der Bash durch.

3. *Freiwillige Zusatzaufgabe*: Erweitern Sie Ihre Lösung so, dass auch mehr als 2 Prozesse über die Pipe verknüpft werden können.

Randbedingungen:

- Alle bereits in der vorherigen Aufgabe implementierten Funktionen sollen beibehalten werden.

- Es muss nur eine einstufige Pipe implementiert werden. Befehle mit einer mehrstufigen Pipe wie z. B.:
  `ls –als /usr/bin | grep root | grep r-s`

  müssen nicht berücksichtigt werden (freiwillige Zusatzaufgabe). Sehen Sie aber eine Fehlerausgabe vor.

Hinweise:

- Zur Beantwortung der Frage 2.a müssen Sie sich überlegen, wie ein schreibender Prozess mehr Daten produzieren als der lesende Prozess konsumieren kann. Dies passiert z. B. beim Aufruf des Pipe-Kommando **`yes |(sleep 10; cat –n)`**. Um die Puffergröße zu beobachten, ist allerdings eine *mehrstufige* Pipe zu realisieren. Überlegen Sie sich, wie das Kommando **`tee`** (liest Standardeingabe und schreibt sowohl auf die Standardausgabe als auch in eine als Parameter übergebene Datei) genutzt werden kann, um die Puffergröße zu ermitteln.
- Ein ähnlicher Befehl kann unter Verwendung des Kommandos **`tail`** auch für die Beantwortung der Frage 2.b entwickelt werden.

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