Vorlesung Betriebssysteme
Wintersemester 2021/2022

Prof. Dr. Lars-Olof Burchard
Hochschule Darmstadt

Aufgabe 2 (Shell, Systemaufrufe)

Programmieren Sie in C / C++ eine in Linux lauffähige Shell, die in beliebige Programme starten kann. Dazu sollen Sie sich über einige Begriffe im Zusammenhang mit Prozessen informieren und diese Begriffe erläutern können. Dazu müssen Sie sich mit den relevanten Systemaufrufen **`fork`**, **`waitpid`**, **`execvp`** vertraut machen und diese verwenden.

Ihre Shell soll analog einer realen UNIX Shell (wie z.B. bash) folgende Funktionalität aufweisen:

1.1 Nach Start der Shell können mittels Tastatur namen von Programmen mit einer beliebigen Anzahl an Parametern eingegeben werden. <enter> (bzw. <return>) beenden die Eingabe eines Befehls und danach wird von Ihrer Shell ein neuer Prozess (!) erzeugt, der den eingegebenen Befehl (ein beliebiges existierendes Programm) ausführt.
Bsp.: "> ls -la".

1.2 Die Prozesse sollen auf zwei Arten gestartet werden können:

1.2.1 Ohne weitere Eingabe soll Ihre Shell auf Ende des Prozesses warten und erst dann neue Eingaben akzeptieren (Ausführung im *Vordergrund*).

1.2.2 Bei zusätzlicher Angabe von "&" nach dem Befehl soll die Shell nicht warten, sondern sofort weitere Eingaben akzeptieren (Ausführung im *Hintergrund*). Die Funktion **`fork`** liefert Ihnen die Prozess ID (*pid*) eines neu gestarteten Prozesses zurück. Für Hintergrundprozesse sollen diese *pids* in einer

Datenstruktur (Liste o.ä.) gespeichert werden und bei jedem neuen Start eines Prozesses diese Liste ausgegeben werden.

1.3 Evtl. eingegebene Parameter sollen korrekt extrahiert und dem aufzurufenden Programm übergeben werden.

1.4 Der Befehl **`logout`** soll Ihre Shell beenden. Dabei soll zuerst abgefragt werden, ob wirklich ein *logout* erfolgen soll und erst dann die Shell beendet werden.

2. Nutzen Sie das Programm "**`ps ax`**" (aus einer anderen Shell als der Ihren), um die von Ihrer Shell ausgegebenen *pids* mit den real im System befindlichen Prozessen zu vergleichen. Was stellen Sie fest? Was wird für Hintergrundprozesse, die bereits beendet wurden, von Ihrer Shell bzw. von **`ps`** angezeigt?
3. Geben Sie zum Testen u.a. die folgenden Befehle (in dieser Reihenfolge) in Ihrer Shell ein:

```
> ./myshell
myshell> echo 1 2 3
[Hintergrund: -]
myshell> date
[Hintergrund: -]
myshell> date &
[Hintergrund: 1234]
myshell> ps ax
[...]
myshell> trasdasd
[Hintergrund: 1234]
myshell> ps ax
[Hintergrund: 1234]
[...]
myshell> logout
Wollen Sie die Shell wirklich beenden (J/N)? J
>
```

4. Frage: Was passiert bei Eingabe von „**`trasdasd`**“ im vorigen Beispiel mit den Prozessen? Gibt es einen Unterschied bei der Ausgabe von **`ps`** vor bzw. nach Eingabe von **`trasdasd`**?

Hinweise:

1. Die Shell soll beliebige Programme (z.B. auch **`firefox`**, **`gedit`**, **`ls`**, ...) ausführen können. Diese Programme sollen in der Shell **nicht** hardcodiert sein.
2. Der Systemaufruf **`system`** kann hier ***nicht*** verwendet werden. Nutzen Sie stattdessen **`execvp`** (oder eine andere Variante).
3. Die genaue Syntax und weitere Erläuterungen zu den Systemaufrufen finden Sie u.a. in den man pages, d.h. zum Beispiel mittels „**`man fork`**“ oder „**`man execvp`**“.
4. Achtung: Es ist nicht notwendig, in der Shell Befehle wie „**`cd`**“ zum Verzeichniswechsel zu implementieren.

Die Aufgabe wird testiert, wenn Sie eine eigene Shell programmiert und mit den hier auf Seite 2 angegebenen Befehlen lauffähig vorgezeigt haben, den Code und die Funktionsweise erklärt haben.