Prof. Dr. M. Jäger
FB MNI

23. Oktober 2015

Übungsblatt 3 - Betriebssysteme

Aufgabe 1 (Programmaufruf mit *exec*)

a) Schätzen Sie ohne vorher nachzusehen:
   - Wieviele Argumente kann man maximal einem Programm übergeben?
   - Wie groß darf ein einzelnes Argument sein?

b) Lesen Sie im Systemmanual den Eintrag zu *execve*.

c) Überprüfen Sie ihre Schätzung durch Nachlesen:
   - Wieviele Argumente kann man maximal einem Programm übergeben?
   - Wie groß darf ein einzelnes Argument sein?

d) Wie ist *execvp* implementiert?

e) Welche Gründe könnten dazu führen, dass der Aufruf fehlschlägt?

f) Einige Prozessattribute werden durch *exec* geändert, andere nicht. Warum werden die Signal-Handler und Exit-Handler beim *exec* zurückgesetzt?

g) Was passiert beim *exec* mit den offenen Dateien?

h) Ermitteln Sie durch Experiment, welchen Wert die Variable *errno* hat, wenn *exec* wegen fehlender Zugriffsrechte scheitert. Wie heißt der symbolische Name diese Fehlers? Wie ist die Klartext-Fehlermeldung?

Aufgabe 2 (Fork/Exec Ausprobieren)

Schreiben Sie ein Programm `aktiviere`, dass zwei Subprozesse erzeugt, in denen **nebenläufig** zwei weitere Programme aufgerufen werden (fork/exec). Insgesamt entstehen also 3 Prozesse.

1. Der erste Subprozess verwendet „execlp“ zum Aufruf von

   ```
   xterm -bg red
   ```

2. Der zweite Subprozess verwendet „execvp“, um ein Programm aufzurufen, dessen Pfad und dessen Parameter an „aktiviere“ übergeben wurden.

   Beispiel:

   Der Aufruf

   ```
   aktiviere xterm -bg green
   ```

   bewirkt im zweiten Subprozess einen Aufruf des Programms „xterm“ mit den drei Parametern „xterm“, „-bg“ und „green“.

3. Der Elternprozess des Programms *aktiviere* gibt für beide Subprozesse die Prozessnummern auf den Bildschirm aus.

   Dann wartet er die Terminierung der Subrozesse ab. Sobald einer der Subprozesse terminiert, soll der Elternprozess eine entsprechende Meldung auf den Bildschirm mit der Prozessnummer des terminierten Subprozesses ausgeben.

Beachten Sie außerdem, dass Sie eine geeignete Fehlerbehandlung (mit dem Systemaufruf *perror*) für den Fall vorsehen, dass ein Systemaufruf nicht klappt.

Aufgabe 3

Schreiben Sie ein Programm `aktiviere2`, das genau wie `aktiviere` andere Programme aufruft und dabei eigene Parameter an das aufgerufene Programm weiterreicht. Falls die eigenen Parameter allerdings die Form *Name=Wert* haben, sollen diese als Umgebungsvariablen weitergereicht werden.

Beispiel:

```
aktiviere2 zeigekontext arg1 arg2 x=y
```

führt zum Aufruf von `zeigekontext` mit den Argumenten `zeigekontext arg1 arg2` und der Umgebungskomponente `x=y`. Verwenden Sie dazu *execve*. Schreiben Sie ein geeignetes Testprogramm `zeigekontext`, dass seine Argumente und seine Umgebung ausgibt.