Prof. Dr. M. Jäger
FB MNI

27. April 2014

Übungsblatt 2 - Betriebssysteme I

Aufgabe 1

Lesen Sie die Dokumentation der unten aufgeführten UNIX-Systemfunktionen (im Skript, Kapitel 10 und/oder im System-Manual):

fork	Erzeugung eines Subprozesses
execlp	Aufruf eines Programms
execvp	Aufruf eines Programms
execve	Aufruf eines Programms
waitpid	Warten auf die Terminierung eines Subprozesses
perror	Fehlermeldung für fehlgeschlagenen Systemaufruf

Aufgabe 2

a) Welche Gründe könnten dazu führen, dass folgende Systemaufrufe fehlschlagen?

- fork
- execlp
- waitpid

b) In welcher Weise kann im Programm das Scheitern eines Systemaufrufs überprüft werden? Wie erfährt man die Fehlerursache?

Schlagen Sie dazu im Online-Manual die Erklärung von *execlp* nach.

c) Bewerten Sie folgende Fehlerbehandlungs-Varianten:

1)
```
execvp(programm, argumente};
/* nicht erreichbarer Code
   wozu Fehlerbehandlung?
*/
```

2)
```
if (execvp(programm, argumente) == <0) {
  perror("Fehler bei exec");
  exit(1);
}
```

3)
```
execvp(programm, argumente);
perror("Fehler bei exec");
```

4)
```
execvp(programm, argumente};
puts("Fehler bei exec");
exit(0);
```

Aufgabe 3

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 4

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.