Dr. Andreas Borchert
Christian Ehrhardt

Abteilung Angewandte Informationsverarbeitung

17. Juni 2003
Blatt 7

Systemnahe Software (SS 2003)

Abgabetermin 24.06.2003

Prozeßketten

Eine Prozeßkette funktioniert genau wie eine Menschenkette: (Fast) jeder Prozeß hat einen Vorgänger und einen Nachfolger. Immer wenn der Prozeß ein Signal von seinem Vorgänger erhält schickt er es an seinen Nachfolger weiter. Umgekehrt werden Signale, die vom Nachfolger kommen an den Vorgänger weitergereicht. Ein Prozeß ohne Nachfolger schickt ankommende Signale immer an den Vorgänger zurück. Ein Prozeß ohne Vorgänger verwirft Signale, die vom Nachfolger kommen.

Bestimmen von Vorgänger und Nachfolger

Die Prozeß-ID des **Nachfolger**s wird beim Starten auf der Kommandozeile angegeben, fehlt diese Angabe, dann gibt es keinen Nachfolger. Der **Vorgänger** ist der Prozeß, von dem zuletzt ein Signal empfangen wurde. (Signale die vom Nachfolger kommen zählen dabei natürlich nicht mit!) Einzige Ausnahme: Wenn zusätzlich zu der Prozeß-ID des Nachfolgers noch der Text "start" als zweites Argument angegeben wird, dann hat der Prozeß keinen Vorgänger und bekommt auch durch ein Signal keinen.

Die Signale

Um das Ganze zu vereinfachen genügt es, wenn das oben beschriebene Verhalten für die Signale `SIGUSR1`, `SIGUSR2`, `SIGTERM` und `SIGINT` funktioniert. Signale, bei denen die Herkunft nicht feststellbar ist sollen nur vom Startprozeß (der, der nie einen Vorgänger bekommen wird) verarbeitet werden, alle anderen Prozesse sollen solche Signale ignorieren. Wenn das Signal `SIGINT` oder `SIGTERM` vom **Nachfolger** kommt, soll es noch einmal ganz normal verarbeitet werden. Anschließend soll sich der Prozeß selbst beenden. Damit sollte es möglich sein, eine ganze Signalkette durch ein einziges Signal an den Startprozeß zu beenden.

Signalinformation

Für die Aufgabe ist es notwendig, den Absender eines Signals zu ermitteln. Dazu muß beim `sigaction(2)` Systemaufruf `sa_sigaction` statt `sa_handler` verwendet werden. Der zweite Parameter des Signalhandlers ist dann ein Zeiger auf Signalinformation (oder ev. `NULL`). Dieser Datenstruktur kann unter anderem der Absender des Signals entnommen werden. Näheres dazu verrät die Manualseite siginfo(3HEAD).

Arbeiten mit asynchronen Signalen

Ein Signal kann an jedem Punkt in einem Programm eintreten. Wenn es also Zeitpunkte gibt, in denen ein Signalhandler nicht beginnen darf (z.B. weil dann Informationen inkonsistent werden könnten), dann muß dafür Sorge getragen werden, daß die betroffenen Signale in dieser Zeit blockiert werden. Selbstverständlich sollte der Zeitraum, in dem Signale blockiert sind so kurz wie möglich sein, weil sonst Signale verloren gehen könnten.