Dr. Andreas Borchert
Christian Ehrhardt

Systemnahe Software (SS 2003)

Abgabetermin 17.06.2003

Instant Messaging

Beim sog. “Instant Messaging” geht es darum, kurze Nachrichten an andere gerade angemeldete Teilnehmer zu verschicken. Diese Nachrichten werden unmittelbar nach dem Absenden auf dem Bildschirm dargestellt. Eine einfache Variante eines solchen Systems soll in diesem Blatt mit Hilfe von Unix-Signalen implementiert werden.
Um Nachrichten empfangen zu können, muß jeder Teilnehmer ein Programm, das im folgenden mit `listener` bezeichnet wird, starten. Dieses Programm registriert den Benutzer und wartet anschließend auf eingehende Nachrichten. Wird das Programm beendet, so sollte der Nutzer nach Möglichkeit auch wieder “deregistriert” werden. Ein zweites Programm dient dann dazu, Nachrichten an einen anderen Nutzer zu versenden.

Kommunikation

Die komplette Kommunikation soll innerhalb eines Verzeichnisses stattfinden, das vom Programmierer festgelegt wird. Im folgenden wird davon ausgegangen, daß dieses Verzeichnis `messages` heißt.

Das `listener`-Programm

Dieses Programm bekommt als Kommandozeilenargument den Namen des Nutzers. Um Nachrichten empfangen zu können, wird in `messages` ein Unterverzeichnis angelegt. Der Name des Unterverzeichnisses entspricht der Prozeß-ID des `listener` Programms. In diesem Unterverzeichnis werden später von anderen Teilnehmern die eingehenden Nachrichten abgelegt. Außerdem muß in diesem Verzeichnis eine Datei mit dem Namen `name` erzeugt werden, die den Namen des Teilnehmers enthält.
Nachdem das Unterverzeichnis und die Datei `name` angelegt wurden, kann der Teilnehmer Nachrichten empfangen. Eingehende Nachrichten werden vom `listener`-Programm auf dem Bildschirm angezeigt und dann gelöscht. Um auch Nachrichten verschicken zu können, soll das `listener`-Programm außerdem die Namen und die zugehörigen Prozeß-IDs von allen gerade registrierten Teilnehmern anzeigen.
Wenn das Programm beendet wird, muß das Unterverzeichnis und dessen Inhalt natürlich wieder gelöscht werden.

Das `sendmsg`-Programm

Dieses Programm dient dazu, eine Nachricht an einen anderen Teilnehmer zu schicken. Die Nachricht selbst wird von der Standardeingabe gelesen. Um den Empfänger zu identifizieren, wird auf der Kommandozeile die Prozeß-ID des zum Empfänger gehörenden `listener`-Prozeß angegeben (wer möchte kann natürlich auch den Namen des Empfängers statt der PID verwenden, das wird aber ein klein wenig aufwendiger).
Das `sendmsg` Programm erzeugt dann eine neue Datei, die mit dem Text “msg.” beginnt im Unterverzeichnis des Empfängers. Diese Datei enthält den Inhalt der Nachricht. Das `listener`-Programm sollte nur Dateien, die wirklich mit “msg.” beginnen als Nachrichten betrachten.

Signale

Das `listener`-Programm darf nicht in einer Art “Busy loop” laufend nachsehen, ob neue Nachrichten eingegangen sind oder ob sich Teilnehmer an- oder abgemeldet haben. Das würde zuviel Rechenzeit benötigen. Statt dessen soll das Programm schlafen und bei einem dieser Ereignisse durch ein Signal geweckt werden. Die folgenden Ereignisse sollen durch Signale mitgeteilt werden:

- Wenn sich ein Teilnehmer an- oder abmeldet, dann soll dessen `listener`-Programm an alle anderen Teilnehmer das Signal `SIGUSR1` verschicken. Die anderen `listener`-Programme sollen darauf reagieren, indem sie eine aktualisierte Liste der Teilnehmer mit Ihren Namen und den zugehörigen PIDs ausgeben.
- Wenn eine neue Nachricht verschickt wurde, soll das `sendmsg` Programm das Signal `SIGUSR2` an den `listener`-Prozeß des Empfängers verschicken. Der Empfänger soll darauf reagieren, indem er alle neuen Nachrichten anzeigt und die zugehörigen Dateien löscht (im Normalfall wird es nur eine neue Nachricht geben).
- Wenn ein `listener`-Prozeß durch `SIGTERM` oder durch Ctrl-C (`SIGINT`) beendet wird, dann soll sichergestellt werden, daß der Teilnehmer ordnungsgemäß abgemeldet wird.

Hinweise

- Signalhandler kann man entweder mit `signal` oder mit `sigaction` setzen. Bei `signal` wirkt sich die Änderung nur auf das nächste Signal aus, anschließend muß der Handler neu gesetzt werden. Bei `sigaction` ist das nicht so.
- Wenn ein Prozeß schläft (z.B. mit `sleep`), dann führt ein eintreffendes Signal auch dazu, daß der Aufruf von `sleep` vorzeitig abgebrochen wird.
- Für das auslesen von Verzeichnissen lohnt es sich ev. einen Blick auf ein passendes Übungsblatt aus AI 3 zu werfen, etwa Blatt Nr. 10 aus dem letzten Wintersemester.