Aufgabe 7:

22.06.2006

timed (12 Punkte)

Die Aufgabe ist einzeln zu bearbeiten! Keine Zweiergruppen!

Entwerfen und programmieren Sie ein Programm **timed** (**time d**aemon), an welchem man die aktuelle Zeit von einem anderen Rechner aus abfragen kann.

a) Verbindung vom Client-Rechner akzeptieren und Zeit ausgeben

Der **timed** soll an einer festen Portnummer Verbindungen akzeptieren. Wählen Sie als Portnummer eine Zahl zwischen 1024-65535 (**socket(2), bind(2), listen(2), accept(2)**). Nach erfolgreichem Verbindungsaufbau soll der **timed** die aktuelle Zeit in einer Zeile im Format "`Jahr-Monat-Tag Stunde:Minute:Sekunden`" auf der Socket-Verbindung ausgeben (nur numerisch, keine Monatsnamen und das Jahr vierstellig) (**time(2)**, **localtime(3), strftime(3)**). Im Verzeichnis */proj/i4sos/pub/aufgabe7* finden Sie ein Modul mit der Funktion **current_time**, welche die aktuelle Zeit entsprechend formatiert. Sie dürfen diese Funktion in Ihrem Programm verwenden. Nach der Ausgabe der Zeit, soll der **timed** die Verbindung schließen und eine neue Verbindung entgegen nehmen.

b) Kontinuierliche Zeitansage und Unterstützung mehrerer Clients

Erweitern Sie das Programm jetzt so, dass die Verbindung zum Client nicht sofort nach Ausgabe der Zeit geschlossen wird, sondern die Zeit kontinuierlich im Abstand von einer Sekunde ausgegeben wird (**sleep(3)**). Außerdem soll der **timed** jetzt in der Lage sein, mehrere Clients gleichzeitig zu bedienen. Erzeugen Sie nach dem Verbindungsaufbau einen neuen Prozess (**fork(2)**), der die Kommunikation mit einem Client übernimmt.

c) Aufräumen von Kind-Prozessen

Richten Sie nun einen Signalhandler für SIGCHLD ein (**sigaction(2)**), und entfernen Sie die anfallenden Zombie-Prozesse (**waitpid(2)**).

Beantworten Sie in einer Datei *doc/timed.txt* folgende Fragen:

- Warum gibt es in Unix einen Zombie-Prozesszustand?
- Wieviele SIGCHLD-Signale bekommt der Vaterprozess zugestellt, wenn während er selber durchgehend blockiert war, zwei Kindprozesse beendet wurden?

Hinweis zur Lösung dieser Aufgabe:

- Sockets sind nicht Bestandteil des POSIX-Standards. Deshalb müssen Sie Ihr Programm mit dem Define `-D_XOPEN_SOURCE=500` übersetzen.
- Als Client zum Testen Ihrer Server-Anwendung können Sie das Programm telnet(1) verwenden.
- Beachten Sie, dass die offenen Filedeskriptoren an die Kind-Prozesse vererbt werden und alle Sockets solange bestehen bleiben bis der letzte Prozess seine Filedeskriptoren geschlossen hat. Darum müssen Sie darauf achten, dass jeder Prozess nur die Filedeskriptoren geöffnet hält, die er auch wirklich braucht.

Abgabe: bis spätestens Mittwoch, 05.07.2006, 17:30 Uhr