Aufgabe 8:

18.06.2007

timed (12 Punkte) Bearbeitung in Zweier-Gruppen

Entwickeln Sie ein Programm **timed** (**time d**aemon), an welchem man die aktuelle Zeit von einem anderen Rechner aus abfragen kann. Das Programm soll in zwei Module unterteilt werden, von denen eines die Annahme von Verbindungen übernimmt und eines die einzelnen Clients behandelt. Alle Vorgaben zu dieser Aufgabe finden Sie im Verzeichnis */proj/i4sos/pub/aufgabe8*.

a) Modul Connection Broker

In der Datei **connbroker.c** soll ein Modul implementiert werden, welches TCP/IP-Verbindungen annimmt. Das Modul ist zugleich das Hauptprogramm und enthält die **main()**-Funktion. Die Portnummer wird dem Programm als einziges Argument übergeben (**socket(2), bind(2), listen(2), accept(2)**). Achten Sie beim Testen im CIP-Pool darauf, dass die Portnummer im Bereich 1024-65535 liegt, da niedrigere Ports nur von *root* gebunden werden können. In Ihrem Programm müssen Sie dieses Intervall jedoch nicht prüfen. Der Connection Broker kümmert sich nur um die Annahme von Verbindungen. Nach Annahme einer neuen Verbindung wird ein Sohnprozess erzeugt, der die Kommunikation mit dem Client übernimmt. Die eigentliche Kommunikation mit den Clients wird in austauschbaren Modulen übernommen, die allesamt eine Funktion

int new_client(int comm_fd, struct sockaddr_in *peer);

bereitstellen. Die Deklaration und Beschreibung dieser Funktion findet sich in der Datei **client_handler.h**. Der Connection Broker ruft für jede neue Verbindung im Kindprozess diese Funktion auf und betrachtet bei der Rückkehr der Funktion die Kommunikation als abgeschlossen, woraufhin die Verbindung geschlossen (**close(2)**) und das Kind terminiert wird (**exit(3)**).

Der Connection Broker soll außerdem auf die Standardausgabe loggen, wenn sich neue Clients verbinden oder die Verbindung beenden. Die hierfür notwendige Ausgabefunktion **printconn()** ist bereits in der **connbroker.c** Vorgabe enthalten, kann jedoch auf den eigenen Geschmack angepasst werden. Rufen Sie diese Funktion in den Kindprozessen vor und nach Aufruf der **new_client()**-Funktion auf. Dies funktioniert nur dann, wenn **new_client()** auch sauber zurückkehrt. Zombie Prozesse sollen vom Connection Broker in einem Signalhandler für SIGCHLD entfernt werden (**sigaction(2), waitpid(2)**).

b) Modul Zeitausgabe

Implementieren Sie nun in einer Datei **printtime.c** ein zur **client_handler-**Schnittstelle konformes Modul. Dieses soll kontinuierlich in Abständen von einer Sekunde 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)**). Die Ausgabe soll so lange erfolgen, bis die Verbindung von Client-Seite aus geschlossen wird. In diesem Fall erhält der Kindprozess ein SIGPIPE, oder, falls dieses ignoriert, blockiert oder behandelt wird, kehrt **write(2)** mit einem Fehler und **errno=EPIPE** zurück. Keinesfalls soll der Sohnprozess unkontrolliert durch das SIGPIPE terminiert werden.

c) Dokumentation

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 selbst durchgehend blockiert war, zwei Kindprozesse beendet wurden?
- Warum muss beim Betreten/Verlassen von Signalbehandlungen i.d.R. der Wert von **errno** gesichert/wiederhergestellt werden. Nennen Sie ein konkretes Beispielszenario, in welchem es andernfalls zu einem Problem kommen könnte, und beschreiben sie das dabei auftretende Problem. Das von Ihnen gewählte Beispielszenario muss nicht notwendigerweise auch in dieser Aufgabe relevant sein.

Hinweis zur Lösung dieser Aufgabe:

- Sie finden in der Vorgabe ein passendes Makefile, den client_handler Header und die Vorgabe für den Connection Broker.
- Als Client zum Testen Ihrer Server-Anwendung können Sie das Programm nc(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.
- Achten Sie auf eine saubere Kapselung der Module. Der Connection Broker soll in einer späteren Aufgabe wiederverwendet werden.

Abgabe: bis spätestens Donnerstag, 28.06.2007, 12:00 Uhr