– Praktikumsaufgabe 9 –

Thema: *Pipes*

Zielstellung: Erlernen der Verknüpfung selbstgeschriebener C-Programme mit Shellkommandos, Implementierung der Datenübertragung mittels unbenannter Pipe; korrekte Synchronisation.

1. Erledigen Sie zunächst alle Aufgaben (ohne Sternchen) aus dem vorangegangenen Praktikum), d. h. machen Sie sich mit `fork()`, `exec()`, `exit()` und `wait()` vertraut.

2. Schreiben Sie ein C-Programm, das zeilenweise von `stdin` liest, jedes eingelesene Zeichen ROT-13-kodiert und die kodierte Zeile nach `stdout` schreibt. Verschlüsseln Sie dann einen beliebigen Text mit Ihrem Programm (Nutzen Sie die Kommandozeile!)! Was passiert bei zweimaliger Verschlüsselung? Überlegen Sie sich einen passenden Mechanismus zur Beendigung des Programms.

   ROT-13 bedeutet, dass jeder Buchstabe um 13 Positionen im Alphabet nach vorn rückt mit Wrap-Around ('a' → 'n', 'b' → 'o', ..., 'm' → 'z', 'n' → 'a' usw.). Vergleiche auch *http://de.wikipedia.org/wiki/ROT-13*, aber schreiben Sie den Code nicht ab!

   Alle, denen die Aufgabe zu primitiv ist, kehren stattdessen die eingelesene Zeile um („Eine alte Gans“ → „Snag etla enie“) und gehen auf die Suche nach Palindromen.

   Hinweise:

   - Nutzen Sie `fgets()` zum Einlesen von `stdin` in etwa folgender Form:

     ```
     ret = fgets(line, LINESIZE, stdin);
     ```

     – `line` ist ein Feld von Zeichen,

     – `LINESIZE` ein Symbol, das die Größe von `line` enthält (die maximal erwartete Zeilenlänge),

     – `stdin` ist eine vordefinierte Konstante.

   - Es ist kein `open()` bzw. `close()` nötig.

   - Nutzen Sie bitte nicht `gets()`; diese Funktion ist eine der Hauptursachen für sogenannte *Buffer Overflows*, die gemeinhin als Eintrittsweg von Schadcode bekannt sind.

   Wenn Sie Lust haben, implementieren Sie die gleiche Funktionalität als Shellskript. Nutzen Sie dafür das Kommando `tr`. Vergleichen Sie den Aufwand für die Implementierung!

3. a) Schreiben Sie ein C-Programm, das

- eine Pipe und
- einen neuen Prozess erzeugt.

Der Vaterprozess soll danach zeichenweise eine (beliebige) Datei einlesen und diese über die Pipe dem Sohn schicken; der Sohn wiederum soll die empfangenen Daten auf dem Bildschirm darstellen. Achten Sie auf korrekte Beendigung der Prozesse!

Nutzen Sie das Gerüst in `geruest-aufgabe-09-03a.c`. Implementieren Sie die fehlende Fehlerbehandlung und modifizieren Sie den Code, so dass der Sohn zeilenweise aus der Pipe liest und nach `stdout` schreibt.

b)* Der Sohn soll die Daten anstatt anzuzeigen ROT-13-verschlüsseln und diese über eine zweite Pipe dem Vater zurückschicken, der sie in eine zweite Datei schreibt.

4.* Wir wollen experimentell untersuchen, ob es möglich ist, mehr als zwei Deskriptoren auf eine Pipe offenzuhalten und zu nutzen. Schreiben Sie dazu ein C-Programm, das folgendes leistet:

- Es soll eine variable Anzahl Prozesse erzeugt werden.
- Jeder der erzeugten Prozesse soll über *ein- und dieselbe Pipe* mit dem erzeugenden Vater verbunden sein.
- Nach der Prozesserzeugung soll der Vater eine Textnachricht über die Pipe an *alle* Söhne schicken.
- Die Söhne sollen die Nachricht ausgeben und sich dann beenden.
- Der Vater soll sich erst dann beenden, wenn alle Söhne beendet wurden.