Prof. Dr. Franz Schweiggert Sektion Angewandte Informationsverarbeitung 28. April 1999
Susanne Schmucker und Michael Lehn Blatt 2

Implementierung kleiner Datenbanken unter UNIX I (SS 1999)

Abgabetermin 6. Mai 1999

1 Dateiumlenkung: "*ls* > *file*" (5 Punkte)

Schreiben Sie ein kleines C-Programm, das das Kommando *ls* zur Ausführung bringt und die Ausgabe in eine Datei schreibt. Den Dateinamen können Sie hart in den Quelltext schreiben, es sind also keine Kunststücke für die Argumentverarbeitung nötig.

2 Pipeline: "do it yourself" (15 Punkte)

Gibt man in der Bash eine Zeile wie *cat xyz* | *sort* ein, passiert bekanntlich folgendes:

- eine Pipe wird erzeugt
- ein neuer Prozeß (P1) wird erzeugt
- P1 führt *cat xyz* aus und schickt die Ausgabe in die Pipe
- der Vaterprozeß schließt das Schreibende der Pipe
- ein zweiter Prozeß (P2) wird erzeugt
- P2 führt *sort* aus, wobei die Eingabe aus der Pipe gelesen wird.

Schreiben Sie ein C-Programm, das der Bash diese Arbeit abnimmt. Statt

cat xyz | *sort* | *grep Schiller*

soll man mit gleichem Erfolg

a.out cat xyz @ sort @ grep Schiller

schreiben können.
Damit Sie sich wirklich nur auf das Wesenliche konzentrieren müssen, erhalten Sie zwei Funktionen, die Ihnen die Argumentverarbeitung (hoffentlich) erleichtern:

- *cmdc:* liefert die Anzahl der zu erzeugenden Prozesse. Im vorigen Beispiel wäre das 3 (cat+sort+grep)
- *cut_cmd:* Liefert Ihnen ein passendes Argument für *execvp*:

```
argv = cut_cmd(0, argv);
execvp(argv[0], argv);
```

  würde im vorigen Beispiel *cat xyz* ausführen. Die Kommandos werden also mit "0" beginnend durchgezählt.
  Achtung: *cut_cmd* macht *argv* kaputt, deshalb nur verwenden, falls Sie *argv* im aufrufenden Prozeß nicht mehr benötigen!

```
#include <sys/types.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int cmdc(int argc, char *argv[]) {
   int i;
   int c = 0;

   for (i = 0; i < argc; i++) {
      if (! strcmp(argv[i],"@")) c++;
   }
   return c + 1;
}

char **cut_cmd(int nr, char *argv[]) {
   int  i, c;
   char **new_argv;

   for (i = 1, c = 0; c < nr; i++) {
      if (!strcmp(argv[i],"@")) c++;
   }
   new_argv = &(argv[i]);
   for (i = i+1; argv[i] != (char *)NULL; i++) {
      if (!strcmp(argv[i],"@")) argv[i] = (char *)NULL;
   }
   return new_argv;
}

int main(int argc, char *argv[]) {
   int i, err;

   /* int anz_prozesse;

      anz_prozesse = cmdc(argc, argv);

      argv = cut_cmd(0, argv);
   */
```

```
   /* ... hier kommt Ihr Teil ..
   */

   while( wait(&i) > 0) { if (i) err = 1; }
   return err;
}
```

3 Pipeline + I/O-Umlenkung (3 Punkte)

Überlegen Sie sich wie Sie Ihr Programm aus Aufgabe 2 erweitern müßten, um Dinge wie

$$cat\ xyz \mid sort > outfile$$

oder

$$mail -s\ Beschwerde\ lehn < kritik$$

zu realisieren.
Erklären Sie Ihrem Tutor wo, wie und was alles geändert werden müßte.