*Universität Ulm - Sektion Angewandte Informationsverarbeitung*

*11. Übungsblatt (01.02.2000 - 10.02.2000)*
*Allgemeine Informatik III (WS 1999/2000)*

# ***I/O-Systemcalls***

17. Aufgabe 10 Punkte

Erstellen Sie ein Programm *io.c* , welches von der Standardeingabe Zeichen einliest und in einer temporären Datei ablegt. Sind alle Zeichen von der Standardeingabe gelesen, so sollen die Zeichen aus der temporären Datei an die Diagnoseausgabe ausgegeben werden.

**Vorgaben:**

- Die Erzeugung der temporären Datei muss "zu Fuß" erfolgen, d.h. Bibliotheksfunktionen wie *mktemp* oder dergleichen dürfen NICHT benutzt werden.
- Der Name der temporären Datei soll aus dem Login-Namen des jeweiligen Benutzers und der Prozess-Id des aktuellen Prozesses zusammengesetzt sein!
- Die temporäre Datei soll natürlich im Verzeichnis *"/tmp"* angelegt werden; der Directory-Eintrag soll - egal wie das Programm terminiert - gelöscht sein!
- Wenn alle Daten an die Diagnoseausgabe geschrieben sind, so soll der mit Zeichen belegte Bereich der benutzen Datenblöcke der temporären Datei aus "Geheimhaltungsgründen" vollständig mit Nullen überschrieben werden, die Datei-Länge darf sich dabei nicht vergrößern.
- Es dürfen nur Funktionen aus der Menge

  *{ sprintf, strcat, getpid, getenv, read, write, open, close, lseek, fstat, perror, unlink, exit }*

  benutzt werden - also kein *scanf, gets, fgets, printf, fprintf, fopen, fseek, ftell* oder dergleichen.

## 18. Aufgabe 10 Punkte

Datenbanksysteme verwenden zur Speicherung von Daten sehr häufig eine Hash-Organisation und feste Datensatzlängen. In dieser Aufgabe soll dies in Teilen nachgebildet werden.

Die Datensätze bestehen aus Namen, die zeilenweise aus einer Datei eingelesen werden sollen (verwenden Sie *fgets* - warum nicht *gets*?). Diese Namen werden in einer Hash-Datei, die

```
const int table_length = 19
```

Datensätze aufnehmen kann. Dort wird jeder Datensatz mit der festen Länge

```
const int record_length = 31
```

abgelegt. Alle nicht explizit belegten Bytes der Dateien sollen Null-Bytes sein!

Aus dem Namen wird wie folgt die Position in der Datei bestimmt:

```
int hash(char string[], int len) {
   int i, s;

   s = i = 0;
   while((i < len) && (string[i] != '\0')) {
      s = (s * 10 + string[i]) % table_length;
      i++;
   }
   return (s);
}
```

Dabei kann es natürlich passieren, dass verschiedene Namen auf dieselbe Position abgebildet werden (Kollision). Dies kann dadurch erkannt werden, dass am Beginn des Speicherortes ein Zeichen ungleich dem Null-Byte steht. Der Datensatz, der auf eine bereits belegte Position trifft, wird in einer Überlauf-Datei wieder mit fester Länge (s.o.) hinten angefügt.

Das Programm erhält zwei Argumente:

1. Namen der Hash- oder Primär-Datei
2. Namen der Überlauf-Datei

Die Hash-Datei wird zu Beginn mit `table_length * record_length` Null-Bytes initialisiert.

**Hinweise:**

- Zum Kopieren von Strings, die Null-Bytes enthalten, sollte die Funktion *memcpy* verwendet werden!
- Zur Ausgabe von Fehlermeldungen darf *fprintf* bzw. *perror* verwendet werden!
- Schreiben, Lesen und Positionieren bei den Dateien darf nur mit den System Calls *read, write, lseek* erfolgen!
- Das Kommando *od* kann zum Testen nützlich sein!

Die Lösung dieses Übungsblattes wird ab Freitag, dem 11.02.2000, im Netz zur Verfügung stehen.