6.3 Extras/Argumente für main()

6.3.1 Problem

Es steht ein Programm zur Vefügung, mit dem man alle Gleitkommazahlen, die zeilenweise in einer Datei stehen, zusammenzählen und ausgeben kann.

```
1  #include <stdio.h>
2  #define PATH_MAX 1024
3  #define PATH_MAX_STR "1024"
4  int main(void)
5  {
6      FILE *dm;
7      char dname[PATH_MAX+1]="";
8      double summe=0.0, zahl=0.0;
9      int rc;
10     printf("Dateiname: ");
11     scanf("%" PATH_MAX_STR "[^\n]", dname);
12     dm=fopen(dname,"r");
13     if(!dm){perror(dname);return 1;}
14     rc=fscanf(dm, "%lf",&zahl); while(fgetc(dm)!='\n' && !feof(dm)){}
15     while(rc>0)
16     {
17         summe+=zahl;
18         zahl=0.0;
19         rc=fscanf(dm, "%lf",&zahl); while(fgetc(dm)!='\n' && !feof(dm)){}
20     }
21     printf("%g\n", summe);
22     fclose(dm);
23 }
```

Der Pfadname der Datei muss dazu bisher an der Tastatur von Hand eingegeben werden.

Terminal

```
schueler@debian964:~$ gcc src/zahlen.c
schueler@debian964:~$ ./a.out
Dateiname: src/zahlen.num
4.2
```

Jetzt soll das Programm aber auch per Mausklick auf das Symbol einer solchen Eingabedatei gestartet werden können.

6.3.2 GUI startet was?

Was passiert eigentlich bei einem Mausklick auf eine Datei mit dem Namen `bsp.doc`? Wenn man es ausprobiert und anschließend eine Liste der laufenden Prozesse öffnet (bei Debian: `Anwendungen`→ `Systemwerkzeuge`→`Systemüberwachung`), bekommt man die Antwort: Es wird ein Textverarbeitungsprogramm geöffnet (z. B. LibreOffice), dieses bekommt den Dateinamen `bsp.doc` als Argument Nr. 1 mitgeliefert.

Genauso ist es, wenn man im Kommandointerpreter die Befehlszeile eingibt:

Terminal

```
schueler@debian964:~$ libreoffice beispiel.doc
```

6.3.3 Argumente für `main()`

Wenn man nun im Kommandointerpreter unser Programm mit Argumenten aufruft, passiert — nichts:

Terminal
```
schueler@debian964:~$ a.out src/zahlen.num
Dateinamen eingeben:
...
```

Das Programm mag also den Dateinamen bekommen, aber es wertet ihn nicht aus. Wie kommt man nun als Programm(ierer) an diese Argumente?

Die Antwort lautet: Programme erhalten die Befehlszeile, mit der sie aufgerufen wurden, in einem Speicherbereich mitgeliefert.

C-Programme können diesen Speicherbereich nutzen mit Hilfe zweier Variablen (nennen wir sie `argc` und `argv`, die *als Parameter* an die `main()`-Funktion übergeben worden sind.

Der erste Parameter `argc` liefert dabei die Anzahl der Worte in der Befehlszeile, wobei die Befehlszeile durch Leerzeichen und Tabs in Worte getrennt wurde. Der Programmname selbst zählt dabei mit!

Der zweite Parameter `argv` liefert alle diese Worte. `argv` ist vom Typ `char *x[]`, das ist ein Array von Zeigern auf Zeichen. Es dient hier als ein Array von nullterminierten Zeichenketten.

`argv[0]` ist die Adresse der Zeichenkette, die den Befehlsnamen enthält. `argv[1]` ist die Adresse des ersten Arguments oder NULL, wenn es kein Argument gab. `argv[2]` ist die Adresse des zweiten Arguments oder NULL, wenn es kein oder nur ein Argument gab.

Damit hat `main()` den neuen Funktionskopf:

```
1 int main(int argc, char *argv[])
```

Es ist auch noch ein dritter Parameter `environ` möglich, der den Zugriff auf eine Liste so genannter *Umgebungsvariablen* erlaubt.

6.3.4 Erweiterung 1

Nun kann das bisherige Programm erweitert werden. Der Anfang kann jetzt so aussehen:

```
1  #include <stdio.h>
2  #include <string.h>
3  #define PATH_MAX 1024
4  #define PATH_MAX_STR "1024"
5  int main(int argc, char *argv[])
6  {
7      FILE *dm;
8      char dname[PATH_MAX+1]="";
9      double summe=0.0, zahl=0.0;
10     int rc;
11     if(argc>1)
12         strncat(dname, argv[1], PATH_MAX);
13     else
14     {
15         printf("Dateiname: ");
16         scanf("%" PATH_MAX_STR "[^\n]", dname);
17     }
18     dm=fopen(dname,"r");
19     if(!dm){perror(dname);return 1;}
20     rc=fscanf(dm, "%lf",&zahl); while(fgetc(dm)!='\n' && !feof(dm)){}
```

Damit kann man es jetzt so aufrufen:

Terminal
```
schueler@debian964:~$ ./a.out src/zahlen.num
4.2
```

6.3.5 Erweiterung 2

Man kann nicht nur einen Dateinamen übergeben, sondern beliebig viele. Dann muss man aber die Array-Elemente von `argv` nacheinander auswerten, so wie in diesem Beispiel:

```
1 #include <stdio.h>
2 int main(int argc, char *argv[])
3 {
4    int lauf;
5    printf("argc = %i\n", argc);
6    for(lauf=0; lauf<argc; ++lauf)
7       printf("argv[] Nr.%i: >>%s<<\n", lauf, argv[lauf]);
8    return 0;
9 }
```

6.3.6 Einbinden in die GUI

Nun soll das Programm in die GUI eingebunden werden, so dass beim Anklicken einer Datei mit der Endung `num` unser Programm mit dem Namen dieser Datei als erstem Argument gestartet wird.

Diese Aktion ist (leider) abhängig von der benutzten GUI. Bei GNOME (enthalten in Debian) funktioniert das so: Man öffnet mit der rechten Maustaste das Kontextmenü und wählt *Mit anderer Anwendung öffnen*, dort *Benutzer-definierten Befehl benutzen*, dort eingeben:

```
gnome-terminal -x /home/nutzer/a.out \%U
```