Klausur -Lösung zu Soft

WS 97

03. Februar 1998

Name ______________________________

Vorname ______________________________

Matrikel-Nr. ______________________________

Studienfach ______________________________

Aufgabe		Punkte	Bewertung
1		10	
2		10	
3		10	
4		10	
5		10	
6		10	
7	a)	4	
	b)	3	
	c)	3	
8		10	
9		10	
10		10	
Summe		**100**	

Hinweis: Bitte benützen Sie den jeweils vorgesehenen Platz. Sollte dieser nicht ausreichen, verwenden Sie bitte die Rückseite und machen Sie eine eindeutige Markierung!

Aufgabe 1

10 Punkte

Schreiben Sie ein Programm zum sog. 3a+1 Problem, welches seine Eingabe als Parameter über die Kommandozeile erhält.

Bei dieser Eingabe muß es sich um eine natürliche Zahl handeln. Diese wird, falls sie gerade ist, halbiert, und sonst mal 3 plus 1 genommen, bis man bei der Eins landet. Eine Beispiel:

```
melzer@thales:~/> aplus1 3
3 10 5 16 8 4 2 1
```

Lösung:

```
#include <stdio.h>
#include <stdlib.h>

void apluseins(int a) {
   printf("%9d ", a);
   if (a - 1) apluseins(((a & 1)?(a * 3 + 1):(a >> 1)));
}

void main(int argc, char *argv[]) {
   if (argc != 2) { fprintf(stderr, "Usage %s Zahl\n", argv[0]); exit(1); }
   if (atoi(argv[1]) <= 0) { fprintf(stderr, "Nat. Zahl\n"); exit(2); }
   apluseins(atoi(argv[1]));
   printf("\n");
   exit(0);
}
```

Aufgabe 2

10 Punkte

Schreiben Sie eine C Struktur fuer einen binären Baum. In einem solchen Baum seien immer fuer jeden Knoten alle Elemente im linken Teilbaum kleiner, als alle Elemente im rechten Teilbaum. Schreiben Sie eine rekursive Prozedur, die einen Zeiger auf einen Knoten erhält und den darunterliegenden Baum inorder ausgibt.

Loesung:

```
struct treenode {
   int whatever;
   struct treenode *left, *right;
};

void printtree(struct treenode *p) {
   if (p) {
      printtree(p->left);
      printf("%d ", p->whatever);
      printtree(p->right);
   }
}
```

Aufgabe 3

10 Punkte

Was liefert folgendes Programm?

```
#include<stdio.h>

void whatdoido(int *a, int *b) {
   *a ^= *b ^= *a ^= *b;
   /*
   => a1 = a xor b; a ^= b^= a1;
   => a1 = a xor b; b1 = b xor a1; a2 = a1 xor b1
   => a1 = a xor b; b1 = b xor a xor b; a2 = a xor b xor b1;
   => a1 = a xor b; b1 = b xor b xor a; a2 = b xor a xor a xor b xor b;
   => a1 = a xor b; b1 = 0 xor a; a2 = b xor 0 xor 0;
   => a1 = a xor b; b1 = a; a2 = b;
   */
}

void main() {
   int i = 17, j = 20;
   printf("Before whatdoido: %d, %d.\n", i, j);
   whatdoido(&i, &j);
   printf("After whatdoido:  %d, %d.\n", i, j);
   exit(0);
}
```

Daher ist die Ausgabe 20, 17.

Aufgabe 4

10 Punkte

Schreiben Sie ein kleines C-Programm, das nach der Eingabe eines Strings überprüft, ob dieser nur jeden der 26 Kleinbuchstaben genau einmal enthält.

Als Tip: qsort

Lösung:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int charcmp(char *a, char *b) {
   return *a - *b;
}

void main() {
   char s[80];
   scanf("%s", s);
   qsort(s, strlen(s) , 1, (void *)charcmp);
   if (strcmp(s, "abcdefghijklmnopqrstuvwxyz")) printf("Nein\n");
   else printf("Ja\n");
   exit(0);
}
```

Aufgabe 5

10 Punkte

Dieses Programm gibt die Parameter nach einem '-' aus. Dabei ist es egal, ob der Aufruf mit -a -b oder mit -ab erfolgt.

```
#include <stdio.h>

void main(int argc, char *argv[]) {
   char *c;
   while (--argc > 0 && (*++argv)[0] == '-') {   /********/
      for (c = argv[0] + 1; *c != '\0'; c++)
      switch (*c) {
         case 'a':
         case 'b':
         case 'c': printf("%c", *c); break;
         default:
            fprintf(stderr, "Illegal option %c\n", *c); break;
      }
   }
   printf("\n");
   exit(0);
}
```

Welche Versionen sind äquivalent zu der markierten Zeile?

```
a) while (--argc > 0 && *++argv[0] == '-') {
b) while (--argc > 0 && **++argv == '-') {
c) while (--argc > 0 && *++(argv[0]) == '-') {
```

Antwort: Nur die mittlere Zeile

Aufgabe 6

10 Punkte

In einem Verzeichnis seien die Dateien talking.c, sock.h, sock.c und testsock.c, sowie folgendes Makefile:

```
MAIN=talking
CC=gcc
LIB=-lsocket -lnsl
FLAGS = -Wall

OBJ=$(MAIN).o sock.o
TOBJ=testsock.o sock.o

main:    $(OBJ)
    $(CC) $(FLAGS) $(LIB) -o $(MAIN) $(OBJ)

testsock:     $(TOBJ)
    $(CC) $(FLAGS) $(LIB) -o testsock $(TOBJ)

.c.o:
    $(CC) $(FLAGS) -c $*.c

clean:
    rm -f $(OBJ) $(TOBJ) core
```

Der Besitzer der Dateien führt folgende Befehle aus:

```
make clean
make
make testsock
make clean
```

Welche Befehle werden von make nun ausgeführt?

Lösung:

```
rm -f talking.o sock.o testsock.o sock.o core

gcc -Wall -c talking.c
gcc -Wall -c sock.c
gcc -Wall -lsocket -lnsl -o talking talking.o sock.o

gcc -Wall -c testsock.c
gcc -Wall -lsocket -lnsl -o testsock testsock.o sock.o

rm -f talking.o sock.o testsock.o sock.o core
```

Aufgabe 7

10 Punkte

Die Dateisysteme neuerer UNIX Systeme haben meist eine Blockgröße von 4K. Eine Inode eines solchen Systems enthält u.a. 15 Zeiger (long int) auf Plattenblöcke. Von diesen sind 12 direkt, einer einfach, einer zweifach und einer dreifach indirekt.

(*a*) Dateien bis zu welcher Größe könnten mit den ersten 14 Zeigern verwaltet werden?

(*b*) Was steht sonst noch in einer solchen inode?

(*c*) Nennen Sie 2 Unterschiede zwischen hard und symb. Links. Geben Sie je ein Beispiel, wann der hard und wann der symb. Link sinnvoller ist.

Lösung:

(*a*) 49152 Bytes direkt + 4194304 einfach indirekt + 4294967296 = 4.299.210.752

(*b*) Art, Owner (2), Zeit (3), Groesse, Links, 15 Zeiger.

(*c*) Ein Hardlink ist lediglich ein weiterer Verweiß auf eine Inode. Ein solcher Link erhöht den Linkcount im 1. Beide Verweisse müssen sich allerdings im gleichen Filesystem befinden, da die ID dieses Systems nicht mitgespeichert wird.

Ein symb. Link hat eine eigene Inode. Er enthält den Namen der ersten Datei. Daher kann er sowohl Filesystemgrenzen überwinden, als auch auf Verzeichnisse und nicht existente Dateien zeigen.

2 Beispiele wären gleichberechtigte Dateien in versch. Verzeichnissen, oder Verweisse zur Versionsunabhänigkeit von Programmen. Also Links auf die neuste Version.

Aufgabe 8

10 Punkte

Gegeben sei folgendes Assembler-Programm (zweispaltig ausgegeben):

thales$ **cat prog.0**

```
label1  ds     0
        store  par2
        store  par1
        print  par1
        print  par2
        store  rueck
        load   par2
        load   par1
        sub
        store  diff
        print  diff
        bnp    label2
        load   par1
        b      label3
label2  ds     0
        load   par2
label3  ds     0
        b      (rueck)
par1    dsf    1
par2    dsf    1
rueck   dsf    1
diff    dsf    1

start     read   a
          read   b
          read   c
          load   Ad.rueck2
          load   Ad.rueck1
          load   a
          load   b
          b      label1
rueck1    ds     0
          load   c
          b      label1
rueck2    ds     0
          store  erg
          print  erg
          leave

a         dsf    1
b         dsf    1
c         dsf    1
erg       dsf    1

Ad.rueck1 dcf    rueck1
Ad.rueck2 dcf    rueck2

          end    start
```

Geben Sie in unten stehender Ausgabe an, was das Programm bei den *print*-Anweisungen liefert:

thales$ **nad prog.0**
execution

119 [start]	read	a	input ?	**4**
120	read	b	input ?	**6**
121	read	c	input ?	**3**
102	print	par1	output =	**4**
103	print	par2	output =	**6**
109	print	diff	output =	**2**
102	print	par1	output =	**4**
103	print	par2	output =	**3**
109	print	diff	output =	**-1**
130	print	erg	output =	**3**
131	leave	halted in cycle 40		

Aufgabe 9

10 Punkte

Sie stellen an Ihrem Unix-Arbeitsplatz fest, wie die *File Creation Mask* definiert ist:

```
thales$ umask
02
thales$
```

Anschließend schreiben Sie eine C-Programm, bei dem die Datei *myfile* wie folgt erzeugt wird:

```
fd = open("myfile", O_WRONLY | O_CREAT, 0777);
```

Anschließend stellen Sie mit dem *ls*-Kommando fest, daß die Datei angelegt wurde und wir fragen Sie nun, mit welchen Rechten. Tragen Sie bitte in folgende Tabelle mit ‘r’, ‘w’, ‘x’ diese Rechte ein und begründen Sie Ihre Wahl!

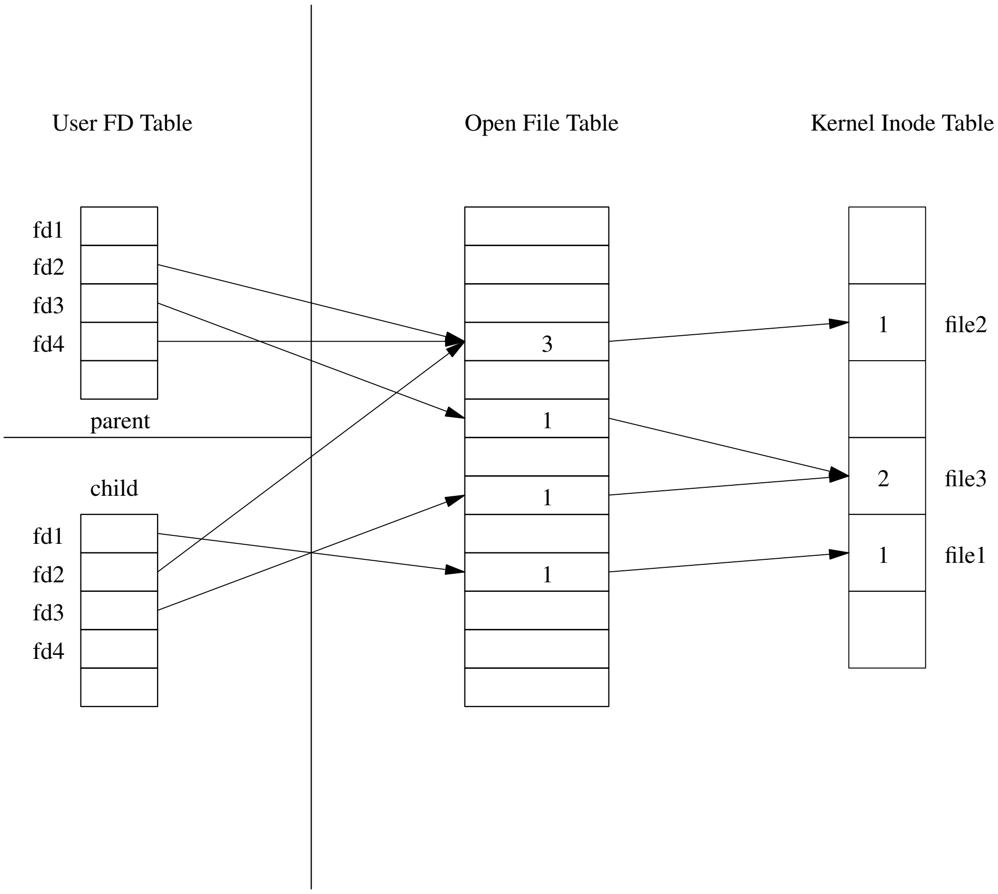
user			group			others		

Lösung:

user			group			others		
r	w	x	r	w	x	r	-	x

Aufgabe 10

10 Punkte



Geben Sie für folgende Sequenz von Befehlen die Belegung der obigen Tabellen am Ende an!

Parent Process	Child Process
fd1=open("file1", O_RDWR); fd2=open("file2", O_RDWR);	
fork();	

fd3=open("file3", O_RDONLY);
fd4=dup(fd2);
close(fd1);
	close(fd1);
	fd1=open("file1", O_RDONLY);
	fd3=open("file3", O_RDWR);