Echtzeitsysteme - Übung

C - Tutorial Zusammenfassung

Felix Binder

27. Oktober 2015

Inhaltsverzeichnis

Vorwort

Alle hier verwendeten Beispiele stammen aus dem C-Tutorial von Elias Fischer [1]. Dieses Dokument fasst das Tutorial lediglich auf seine elementaren Bestandteile zusammen.

1 Einführung

1.1 „Hello World!“

Datei: **HelloWorld.c**

```c
#include<stdio.h>

int main() {
    printf("Hello World\n");
    return 0;
}
```

Kompilieren der Hochsprache zu ausführbarem Maschinencode unter UNIX:

```
$ gcc -o HelloWorld HelloWorld.c
```

Maschinencode ausführen mit Ergebnis im Terminal.

```
$ ./HelloWorld
$ Hello World!
```

1.2 Kommentare

Datei: **HelloWorld.c**

```c
#include<stdio.h>

int main() {
    //printf("Diese Zeile ist auskommentiert!\n");

    /*
    printf("Diese Zeilen\n");
    printf("sind ebenfalls\n");
    printf("auskommentiert\n");
    */

    printf("Ziemlich leer oder?\n");
    return 0;
}
```

```
$ ./HelloWorld
$ Ziemlich leer oder?
```

1.3 Hexadezimale Zahlen

Datei: **HelloWorld.c**

```c
#include<stdio.h>

int main() {
    int tmp = 0xFF;
    printf("Zahl: %d\n", tmp);
    return 0;
}
```

```
$ ./HelloWorld
$ Zahl: 255
```

Schlüsselwort: **0x**..

1.4 Binäre Zahlen

Datei: **HelloWorld.c**

```c
#include<stdio.h>

int main() {
    int tmp = 0b0101;
    printf("Zahl: %d\n", tmp);
    return 0;
}
```

```
$ ./HelloWorld
$ Zahl: 5
```

Schlüsselwort: **0b**....

2 Variablen

2.1 Elementare Datentypen

Einfache arithmetische Datentypen der Programmiersprache C (1)

Datentyp	Verwendung	Wertebereich	Größe
char	kleine natürliche Zahl mit Vorzeichen oder Zeichen	$-128 \ldots 127$	1 Byte
unsigned char	kleine natürl. Zahl ohne Vorzeichen	$0 \ldots 255$	1 Byte
int	natürliche Zahl	$-2^{31} \ldots 2^{31}-1$	4 Byte
unsigned int	natürliche Zahl ohne Vorzeichen	$0 \ldots 2^{32}-1$	4 Byte
float	Dezimalzahl	$3.4e+/-38$ (7-stellig)	4 Byte
double	Dezimalzahl	$1.7e+/-308$ (15-stellig)	8 Byte
long double	Dezimalzahl	$1.1e+/-4932$	10 Byte

- Größenangaben : abhängig von verwendeter Computerachitektur;
- Angaben gelten für die (noch) üblichen 32-Bit Architekturen.
- Zusätzliche Varianten an Ganzzahl-Typen:
  - **signed** / **unsigned** : möglich mit `char` und `int`
  - **short** : möglich mit `int`
  - **long** : möglich mit `int` und `double`

Abbildung 1: Elementare Datentypen in C, Quelle: [2]

2.2 Variablen Deklaration

```
//Variablen Deklaration und Initialisierung
int var1, var2 = 10, var3;

//Konstante festlegen
const int var4 = 21;
```

2.3 Rechenoperationen

```
int a, b, c;

// Zuweisung eines konstanten Wertes, a ist 1
a = 1;
// Zuweisung eines Variablenwertes, b ist 1
b = a;

// Summe
a = b + c;
// Subtraktion
a = b - c;
// Multiplikation
a = c * b;
// Division
a = c / b;

// a mit Inkrement-Operator hochzaehlen (a = a + 1)
a++;
// a mit Dekrement-Operator erniedrigen (a = a - 1)
a--;

// Erst Zuweisung, dann inkrementieren
a = b++;
// Erst inkrementieren, dann zuweisen
a = ++b;

// Modulo (Restberechnung)
a = b % c;

// Uebernahme des linksseitigen Wertes nach der rechtsseitigen Berechnung (Bsp.
    a = (b+c)+a)
a += b + c;
a -= b + c;
a *= b + c;
a /= b + c;
a %= b + c;
```

2.4 Logische Operatoren

```
int a, b, c;

//AND
```

```
a = b & c;
//OR
a = b | c;
//XOR
a = b ^ c;
//NOT
a = ~b;

//Bit-shifting
a = b << 1;
c = b >> 1;
```

2.5 Casting (Typumwandlung)

```
int a;
float b;

a = (int) b;
```

3 Benutzerinteraktion

3.1 Bildschirmausgabe

In den Ausgabestrings können Formatierungsplatzhalter verwendet werden. Damit ist es möglich in einen eigentlich statischen String den Wert einer Variable einzufügen. Als Platzhalter wird der Typ der Variable angegeben und diese anschließend durch Komma getrennt und von links nach rechts gelesen an die Konsolenausgabe mit angehängt. %c ist der Platzhalter für ein einzelnes Zeichen (Char), %d für eine ganze Zahl und %f für eine Gleitkommazahl. Zusätzlich können weitere Formatierungsparamter mit angegeben werden. So kann zum Beispiel mit „%2.3f“ ein Platzhalter definiert werden, der die entsprechende Gleitkommazahl auf zwei Vorkommastellen und 3 Nachkommastellen in der Ausgabe reduziert.

```
float laenge=312.5789, breite=5.6;
printf("\nLaenge: %f cm\nBreite: %f cm\n", laenge, breite);
printf("\nLaenge: %10.2f cm\nBreite: %10.2f cm\n", laenge, breite);
```

```
$ Laenge: 312.578888 cm
$ Breite: 5.600000 cm

$ Laenge:     312.58 cm
$ Breite:       5.60 cm
```

3.2 Benutzereingabe

Einzelnes Zeichen einlesen mit „getchar()“:

```
char c;
printf("Mit welchem Buchstaben beginnt ihr Vorname? ");
c = getchar();
printf("\nIch weiss jetzt, dass Ihr Vorname mit '%c' beginnt.\n", c);
```

Platzhalter (%d und %f) einlesen mit „scanf()“:

```
int tag, monat, jahr;
printf("Bitte geben Sie ihr Geburtsdatum ein [TT.MM.JJJJ]: ");
scanf("%d.%d.%d", &tag, &monat, &jahr);
printf("\nIhr internationales Geburtsdatum: %04d-%02d-%02d\n", jahr, monat, tag)
    ;
```

Bei der Benutzereingabe über die Tastatur wird immer die Eingabebestätigung mittels Enter auch von „scanf()“ mitgelesen. Daher ist eine temporäre Variable notwendig um die ungewünschte Eingabebestätigung abzufangen.

```
char a, b, temp;

printf("\nGeben sie ein Zeichen ein: ");
scanf("%c%c", &a, &temp);

printf("\nGeben sie ein Zeichen ein: ");
scanf("%c%c", &b, &temp);

printf("\nDie ASCII-Codes ihrer Zeichen sind %d und %d\n", a, b);
```

4 Operatoren und Funktionen

4.1 Operatoren

Vergleichsoperatoren

```
int a = 0,b = 2;

//Gleichheit
if(a == b);

//Ungleich
if(a != b);

//Groesser bzw. groesser gleich
if(a > b);
if(a >= b);

//Kleiner bzw. kleiner gleich
if(a < b);
if(a <= b);
```

Logische Operatoren

```
int a = 0,b = 2;

//Negation
if(!a);

//Logisches UND
if(a && b);

//Logisches ODER
```

```
10 if(a || b);
```

4.2 Verzweigungen und Schleifen

4.2.1 If - Else if - Else

```
1 int zahl=6;
2
3 if(zahl==5) printf("fuenf\n");
4 else if(zahl==6) printf("sechs\n");
5 else printf("nicht fuenf und nicht sechs\n");
```

4.2.2 Switch - Case

```
1 int a=2;
2
3 switch(a) {
4     case 1: printf("a ist eins\n"); break;
5     case 2: printf("a ist zwei\n"); break;
6     case 3: printf("a ist drei\n"); break;
7     default: printf("a ist irgendwas\n"); break;
8 }
```

4.2.3 While

```
1 int i=1;
2 while(i <= 100) {
3     printf("Zahl %d\n", i);
4     i++;
5 }
```

4.2.4 For

```
1 int i;
2
3 for(i=0; i<5; i++) {
4     printf("Zahl %d\n", i+1);
5 }
```

4.2.5 Do - while

```
1 int alter;
2
3 do {
4     printf("\nBitte geben sie ihr Alter ein: ");
5     scanf("%d", &alter);
6 } while(alter <  5 || alter > 100);
7
8 printf("Danke.\n");
```

Mit **break** können alle Schleifen unterbrochen werden, ohne den Kontrollpunkt nochmal zu passieren. Mit **continue** wird in jeder Schleife direkt zum Kontrollpunkt gesprungen und alle nachfolgenden Argumente somit ignoriert.

Literatur

[1] Fischer, Elias: „Das C Tutorial (deutsch)“, http://www.c-howto.de/tutorial-einfuehrung.html, Abgefragt am: 23.10.2015

[2] Spurk, R.: „Datentypen in der Programmiersprache C“, Universität Saarland, 2009, http://www.rw.cdl.uni-saarland.de/teaching/c08/mat/cc++_ws0809.27-10-2008_ElementareDatentypen-Teil1.pdf, Abgefragt am: 24.10.2015

Anhang

A Ascii Tabelle

ASCII Tabelle

Scan-code	ASCII hex	dez	Zeichen	Scan-code	ASCII hex	dez	Zeichen	Scan-code	ASCII hex	dez	Zeichen	Scan-code	ASCII hex	dez	Zeichen
	00	0	NUL		20	32	SP		40	64	@	0D	60	96	`
	01	1	SOH ^A	02	21	33	!	1E	41	65	A	1E	61	97	a
	02	2	STX ^B	03	22	34	"	30	42	66	B	30	62	98	b
	03	3	ETX ^C	29	23	35	#	2E	43	67	C	2E	63	99	c
	04	4	EOT ^D	05	24	36	$	20	44	68	D	20	64	100	d
	05	5	ENQ ^E	06	25	37	%	12	45	69	E	12	65	101	e
	06	6	ACK ^F	07	26	38	&	21	46	70	F	21	66	102	f
	07	7	BEL ^G	0D	27	39	'	22	47	71	G	22	67	103	g
0E	08	8	BS ^H	09	28	40	(	23	48	72	H	23	68	104	h
0F	09	9	TAB ^I	0A	29	41	)	17	49	73	I	17	69	105	i
	0A	10	LF ^J	1B	2A	42	*	24	4A	74	J	24	6A	106	j
	0B	11	VT ^K	1B	2B	43	+	25	4B	75	K	25	6B	107	k
	0C	12	FF ^L	33	2C	44	,	26	4C	76	L	26	6C	108	l
1C	0D	13	CR ^M	35	2D	45	-	32	4D	77	M	32	6D	109	m
	0E	14	SO ^N	34	2E	46	.	31	4E	78	N	31	6E	110	n
	0F	15	SI ^O	08	2F	47	/	18	4F	79	O	18	6F	111	o
	10	16	DLE ^P	0B	30	48	0	19	50	80	P	19	70	112	p
	11	17	DC1 ^Q	02	31	49	1	10	51	81	Q	10	71	113	q
	12	18	DC2 ^R	03	32	50	2	13	52	82	R	13	72	114	r
	13	19	DC3 ^S	04	33	51	3	1F	53	83	S	1F	73	115	s
	14	20	DC4 ^T	05	34	52	4	14	54	84	T	14	74	116	t
	15	21	NAK ^U	06	35	53	5	16	55	85	U	16	75	117	u
	16	22	SYN ^V	07	36	54	6	2F	56	86	V	2F	76	118	v
	17	23	ETB ^W	08	37	55	7	11	57	87	W	11	77	119	w
	18	24	CAN ^X	09	38	56	8	2D	58	88	X	2D	78	120	x
	19	25	EM ^Y	0A	39	57	9	2C	59	89	Y	2C	79	121	y
	1A	26	SUB ^Z	34	3A	58	:	15	5A	90	Z	15	7A	122	z
01	1B	27	Esc	33	3B	59	;		5B	91	[		7B	123	{
	1C	28	FS	2B	3C	60	<		5C	92	\		7C	124	|
	1D	29	GS	0B	3D	61	=		5D	93	]		7D	125	}
	1E	30	RS	2B	3E	62	>	29	5E	94	^		7E	126	~
	1F	31	US	0C	3F	63	?	35	5F	95	_	53	7F	127	DEL

Abbildung 2: Ascii Tabelle, Quelle: [1]