KV Betriebssysteme
Einführung in C

KV Betriebssysteme
Überblick über die LVA

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Ziele und Voraussetzungen

- **Ziele**
  - Einführung in die systemnahe Programmierung mit Hilfe der Programmiersprache C.
  - Vertiefung der Kenntnisse aus der Vorlesung Betriebssysteme
  - Übung der erworbenen Kenntnisse anhand praktischer Beispiele
- **Voraussetzungen**
  - Java-Kenntnisse
  - Programmiererfahrung

Systemvoraussetzungen

- **Hardware**
  - PC (Intel- oder kompatibler Prozessor)
- **Betriebssystem**
  - Windows NT 4, Windows 2000, Windows XP
  - (Windows 95, Windows 98, Windows ME)
  - (Linux)
- **Software**
  - Für C: Microsoft Visual C/C++ 6.0 (5.0)
  - Für Java: Pow! 3.0, Sun JDK

Geplante Termine (I)

Datum	Inhalt	Übung
05.03. / 06.03.	Einführung in C	
12.03. / 13.03.	Einführung in C	1
19.03. / 20.03.	Einführung in C	2
26.03. / 27.03.	Osterferien	
02.04. / 03.04.	Osterferien	
09.04. / 10.04.	Einführung in C	3
16.04. / 17.04.	Einführung in C	4
23.04. / 24.04.	Windowsprogrammierung	5
30.04. / 01.05.	Tag der Arbeit	

Geplante Termine (II)

Datum	Inhalt	Übung
07.05. / 08.05.	Windowsprogrammierung	6
14.05. / 15.05.	Windowsprogrammierung	7
21.05. / 22.05.	Pfingsten	
28.05. / 29.05.	Windowsprogrammierung	8
04.06. / 05.06.	Java VM, Java Multiprocessing	9
11.06. / 12.06.	Java VM, Java Multiprocessing	10
18.06. / 19.06.	Java VM, Java Multiprocessing	
25.06. / 26.06.	Klausur	

Modus

- **Jede Woche wird eine Übung ausgeteilt. Es gibt 10 Übungen. 8 von 10 Übungen müssen abgegeben werden, um zur Klausur antreten zu können.**
- **1/3 der Übungen werden korrigiert. Übungen mit weniger als der Hälfte der möglichen Punktezahl werden als nicht abgegeben gewertet.**
- **Die Endnote setzt sich zusammen aus dem Klausurergebnis, der Anzahl der abgegebenen Übungen, der Mitarbeit und der erreichten Punktezahl bei den korrigierten Übungen.**

KV Betriebssysteme
Einführung in C

Ansprechpartner

- **DI. Roland Eggetsberger**
  - Tel.: 0732 / 2468 - 8442
  - E-Mail: eggetsberger@fim.uni-linz.ac.at
  - Sprechstunden: Mo, 9:00 - 12:00
- **DI. Markus Jöbstl**
  - E-Mail: joebstl@fim.uni-linz.ac.at
- **DI. Dr. Peter René Dietmüller**
  - E-Mail: dietmueller@fim.uni-linz.ac.at

KV Betriebssysteme Einführung in C

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Geschichte & Literatur

- **Geschichte**
  - 1970 von Dennis Ritchie für eine DEC PDP-11 entw.
  - 1978 DeFacto Standard durch das erste C-Buch von Dennis Ritchie und Brian Kernighan (Kernighan/Ritchie Standard)
  - 1982 Definition des heute üblichen Standards des American National Standards Institute (ANSI)
- **Standardwerke**
  - Kernighan Brian W., Ritchie Dennis: Programmieren in C, Zweite Auflage: ANSI C, Carl Hanser, 1990
  - Stroustrup Bjarne: Die C++ Programmiersprache; Addison-Wesley, 1987

Eigenschaften von C (I)

- **C ist eine weit verbreitete Programmiersprache.**
  - C-Compiler existieren für die meisten Zielsysteme.
- **C kann auf kleinstem Raum beschrieben werden.**
- **C kann schnell erlernt werden.**
- **C ist sehr effizient.**
  - Die meisten C-Systeme können den erzeugten Code auf Codelänge oder Geschwindigkeit hin optimieren.
- **C selbst definiert keine Speicherverwaltung.**
- **C selbst hat keine Ein- und Ausgabeoperationen.**

Eigenschaften von C (II)

- **C kennt nur Funktionen, keine Prozeduren.**
- **C kennt keine verschachtelten Funktionen.**
- **C-Funktionen können eine variable Anzahl von Parametern haben.**
- **Parameter werden in C immer als Call-by-value-Parameter übergeben.**

Das erste C-Programm

- **Jedes C-Programm besteht zumindest aus einer Funktion main. Im folgenden Beispiel gibt das Programm den Text "Hello world." aus.**

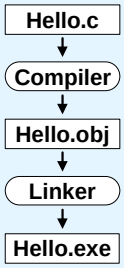
```
#include <stdio.h>
int main (void) {
    printf ("Hello world.\n");
    return 0;
}
```

KV Betriebssysteme
Einführung in C

Struktur eines C-Programmes

- Preprocessor-Anweisungen
  - z.B.: #include, #define
- Typdeklarationen
  - z.B.: typedef, struct
- Funktionsprototypen
- Globale Variablen
- Funktionen
- Hauptprogramm (Funktion main)

C-Programm übersetzen



Grundlegende Syntaxregeln

- Anweisungen sind durch ; zu trennen; sie lassen sich durch { } in Blöcke zusammenfassen (nach } kein Strichpunkt!)
- C ist case-sensitiv
- Funktionen müssen immer mit () aufgerufen werden, auch wenn sie keine Parameter haben!
- Es können nur Variablen und Funktionen verwendet werden, die vorher deklariert wurden.
- Variablen können global (außerhalb der Funktionen) bzw. lokal (an jedem Blockanfang) deklariert werden

Kommentare

- Syntax
  - /* ... */
  - Kommentare können nicht geschachtelt werden!
  - Kommentare mit // gibt es nur in C++! (Ausnahme: MS Visual C++)
- Was soll kommentiert werden?
  - Am Programmanfang (Header): Programmname, Autor, Beschreibung, Verwendung, Dateiformate, Einschränkungen, Fehlerbehandlung
  - Im Programm: Komplizierte Stellen des Programmes

Programmierstil

- Sprechende Bezeichner (mit Kommentar)
  - int a;
  - int a; /* Anzahl der Elemente */
  - int anzahl; /* Anzahl der Elemente */
  - int nAnzahlElemente;
- Einrückung
  - Einheitlich!
- Klarheit (Unterteilung in Einheiten)
  - Funktionen, Module
- Einfachheit

KV Betriebssysteme
Elementare Datentypen

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

KV Betriebssysteme
Einführung in C

Elementare Datentypen (I)

Typ	Länge	von	bis
char	1	-128	127
unsigned char	1	0	255
short	2	-32768	32767
unsigned short	2	0	65535
int		Systemabhängig	
unsigned int		Systemabhängig	
long	4	–2,147,483,648	2,147,483,647
unsigned long	4	0	4,294,967,295

Elementare Datentypen (II)

Typ	Länge	von	bis
float	4	3.4*10^-38	3.4*10^38
double	8	1.7*10^-308	1.7*10^308
long double	10	1.2*10^-4932	1.2*10^4932
void	0		

Literale

- Ganze Zahlen
  - Dezimal: 160
  - Hexadezimal: 0xA0
  - Oktal: 0240
  - Long: 160L
  - Unsigned: 160U
- Reelle Zahlen
  - Dezimal: 34.2
  - Exponential: 3.42E1
- Zeichen
  - 'a'
- Zeichenkette
  - "My name is Bond, James Bond"

Datentyp char

- Der Datentyp „char“ repräsentiert ein einzelnes ASCII-Zeichen.
- Mit dem Datentyp „char“ kann man auch rechnen. Als Wert fungiert dann der ASCII-Wert des Zeichens.
- Beispiel
  - 'B' - 'A' ergibt 1 (66 - 65)
  - 'Z' - 'A' ergibt 25 (90 - 65).
  - '9' - '0' ergibt 9 (57 - 48).

Sonderzeichen (I) (Escape Sequence)

Konstante	ASCII-Wert	Bedeutung
\0	0	Stringende
\b	8	Backspace
\t	9	Tabulatorsprung
\n	10	Zeilenvorschub
\f	12	Seitenende
\r	13	Zeilenende
\"	34	"-Zeichen
\'	39	' -Zeichen
\\	92	\-Zeichen
\nnn	nnn	Oktal-Zeichen nnn

Sonderzeichen (II)

Zeichenkette	entspricht
"\n"	(Leerzeile)
"Hallo,\nwie geht's.“	Hallo, wie geht's.
"Vorname\tNachname“	Vorname Nachname
"c:\\autoexec.bat"	c:\autoexec.bat
"Hallo\rWelt"	Welt (am Bildschirm)

Datentyp boolean

- C kennt keinen expliziten Datentyp Boolean.
- Stattdessen wird ein Integer-Wert verwendet, der folgendermaßen interpretiert wird:
  - ein Wert von 0 bedeutet FALSE
  - ein Wert ungleich 0 bedeutet TRUE
- Diese Regelung ist manchmal praktisch:
  - Statt „if (i != 0) ...“ kann man schreiben „if (i) ...“

KV Betriebssysteme
Variablen und Funktionen

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Variablendeklaration

- Syntax
  - Typ Variablenname {"," Variablenname}
  - Variablennamen müssen mit einem Buchstaben oder „_“ beginnen. Danach können Buchstaben oder Ziffern kommen. Reservierte Wörter dürfen nicht verwendet werden.
- Beispiel
  - int i, j;
  - char c;

Funktionsdeklaration

- Syntax
  - Typ Funktionsname (Parameter) { Funktionsrumpf }
  - Zur Erinnerung: In C gibt es keine Prozeduren! Dafür gibt es den Datentyp void!
  - Default-Rückgabetyp ist int.
- Beispiel
  - int Max(int a, int b) {
    int c;
    c = (a > b) ? a : b;
    return c;
    }

Funktionsaufruf

- Syntax
  - Funktionsname(Aktualparamter);
  - Parameter werden immer „Call by Value“ übergeben.
  - Der Rückgabewert kann unter den Tisch fallen.
  - Klammern nach dem Funktionsnamen müssen immer angegeben werden, auch wenn keine Parameter übergeben werden!
- Beispiele
  - int c;
    c = Max(1, 2);
  - Max(1, 2);

Datentyp void

- Bedeutung
  - Der Datentyp void wird meist als Rückgabetyp von Funktionen verwendet. Durch Verwendung von void kann man explizit angeben, daß eine Funktion keinen Wert zurückgibt. In diesem Fall darf „return“ nicht verwendet werden.
- Beispiel
  - void DoMax (int a, int b) { /* Funktion o. Rückgabe */
    int c;
    c = Max(a,b);
    }

KV Betriebssysteme
Einführung in C

Funktionsprototyp

- **Bedeutung**
  - Schnittstelle einer Funktion bekanntmachen
- **Syntax**
  - Typ Funktionsname (Parameter);
- **Beispiel**
  - int Max(int x, int y);

    ... (Aufruf der Funktion Max) ...

    int Max(int x, int y) { ... }

Sichtbarkeit von Variablen

```
int i;    /* globale Variable */

void test (void) {
  int i;  /* lokale Variable */
  i = 2;
}

int main (void) {
  i = 1;
  test();
  return 0;
}
```

glob. i	lok. i
0	
	2
1	
1	
1	

Speicherklasse

- **Auto**
  - Standard für lokale Variablen. (Kein Schlüsselwort)
- **Static**
  - Variablen werden nur einmal initialisiert.
  - Variablen können beliebig verändert werden.
  - Bsp.: static int zaehler = 0;
- **Extern**
  - Variablen werden aus anderen Quellen übernommen.
  - Bsp.: extern int zaehler;
- **Register**

Lifetime, Linkage, Visibility

Level	Item	Storage Class	Lifetime	Linkage	Visibility
File scope	Variable	static	Global	Internal	source file
	Variable	extern	Global	External	source file
	Variable	auto	Local	External	source file
	Variable	register	Local	External	source file
	Function	static	Global	Internal	Single source file
	Function	extern	Global	External	source file
	Function	auto	Local	External	source file
Block scope	Variable	static	Global	Internal	Block
	Variable	extern	Global	External	Block
	Variable	auto	Local	Internal	Block
	Variable	register	Local	Internal	Block