Freitags 14:00 – 17:00
CIP Pool INF 226

Einführung in die Datenanalyse mit dem C++ Toolkit ROOT

Jörg Marks, Physikalisches Institut, INF 226
marks@physi.uni-heidelberg.de

Programm Überblick

- Linux/Unix Arbeitsumgebung
- Grundlagen der Programmiersprache C++
- Einführung in das Analysewerkzeug ROOT

Organisatorisches

- 4 Leistungspunkte:
  - Anwesenheitspflicht mit Lösung der Übungsaufgaben
  - Klausur / Vortrag
- Kurs web page

  http://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/

Informationen zur Veranstaltung (1)

Ziel

- Nutzung eines Computers mit UNIX Betriebssystem zum Erstellen von C++ Programmen im Hinblick auf Datenanalyse
- Grundlagen zur Programmierung in C++ mit dem Ziel
  - Kennenlernen des Sprachraums, um die C++ Schnittstelle von ROOT effiktiv nutzen zu können.
  - ROOT Quellcode ansehen zu können.
- Einführung in das Datenanalysewerkzeug ROOT
  - Input / Output von Messungen und Resultaten
  - Nutzung vorhandener Funktionen zur graphischen Darstellung von Messungen
  - Datenanpssung zur Bestimmung von Modelparametern
  - Statische Methoden der Datenauswertung
- Beispielorientiert Konzepte so erläutern, dass Sie mit den Erklärungen selbständig (kleine) Datenanalyseaufgaben lösen können.
  - Kein stringenter Programmierkurs → Tutorial
  - Grundlagen für das Erstellen problemorientierter Lösungen schaffen

Voraussetzungen

- Keine Vorkenntnisse notwendig, aber hohe Informationsdichte und Tempo
- User ID zur Benutzung der CIP Pools der Fakultät für Physik.

Informationen zur Veranstaltung (2)

Struktur des Kurses

- Wechsel zwischen Vorlesung und Übungen
- Wechsel zwischen selbstständigem Üben und Übungen in Kleingruppen
- Erläutern und Diskutieren der Lösungsvorschläge
- Kurszeiten: Freitags 14:00 – 17:00 (4 stündige Veranstaltung)
- Kurs Web Page:
  http://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/
  - Vorlesungstransparente
  - Beispiel Code
  - Übungsaufgaben
  - Lösungsvorschläge

Voraussetzungen für einen Leistungsnachweis (4 LP)

- Anwesenheitsliste / mehr als 1 x Abwesenheit nur mit Attest
- Aktive Mitarbeit und kleine Übungen als Hausarbeit
- Klausur am Ende des Kurses, Termin: 20.1.2017 um 14:00

Einleitung und Motivation

Daten

- Norm des internationalen Technologiestandards (ISO/IEC 2382-1, 1993)
  data: „ a reinterpretable representation of information in a formalized manner, suitable for communication, interpretation, or processing “
- Informatik
  - Maschinenlesbare und -bearbeitbare, digitale Repräsentation von Information.
  - In Zeichen bzw. Zeichenketten kodiert, deren Aufbau Regeln (Syntax) folgt.
  - Um aus Daten wieder die Informationen zu abstrahieren, müssen sie in einem Bedeutungskontext interpretiert werden.
- Speicherung der Daten auf Festplatten, Magnetbändern, Flashspeicher, ...
  - Erwartete Gesamtmenge 2020: $40 \cdot 10^{21}$ Bytes
- Vorlesung: durch Messung / Beobachtung gewonnene Information in Form von Zahlen und Text mit folgenden Eigenschaften
  - die Menge der Daten ist typischer Weise gross, keine Durchsicht „von Hand“
  - die gemessenen Werte müssen in physikalische Information verwandelt werden
  - die Information liegt nicht in reiner Form vor, sondern ist in anderen Daten versteckt

  → Lerne Techniken zur Verarbeitung von Daten.

Einleitung und Motivation

Datenanalyse

- Individuelle Lösung durch Erstellen eines selbstgeschriebenen Computerprogramms zur Auswertung der Daten

  - z. B. Mittelwerte, Zeitabhängigkeit, graphische Darstellung, Anpassung und Extraktion von Modelparametern, …

    → ok , aber nicht sehr effizient

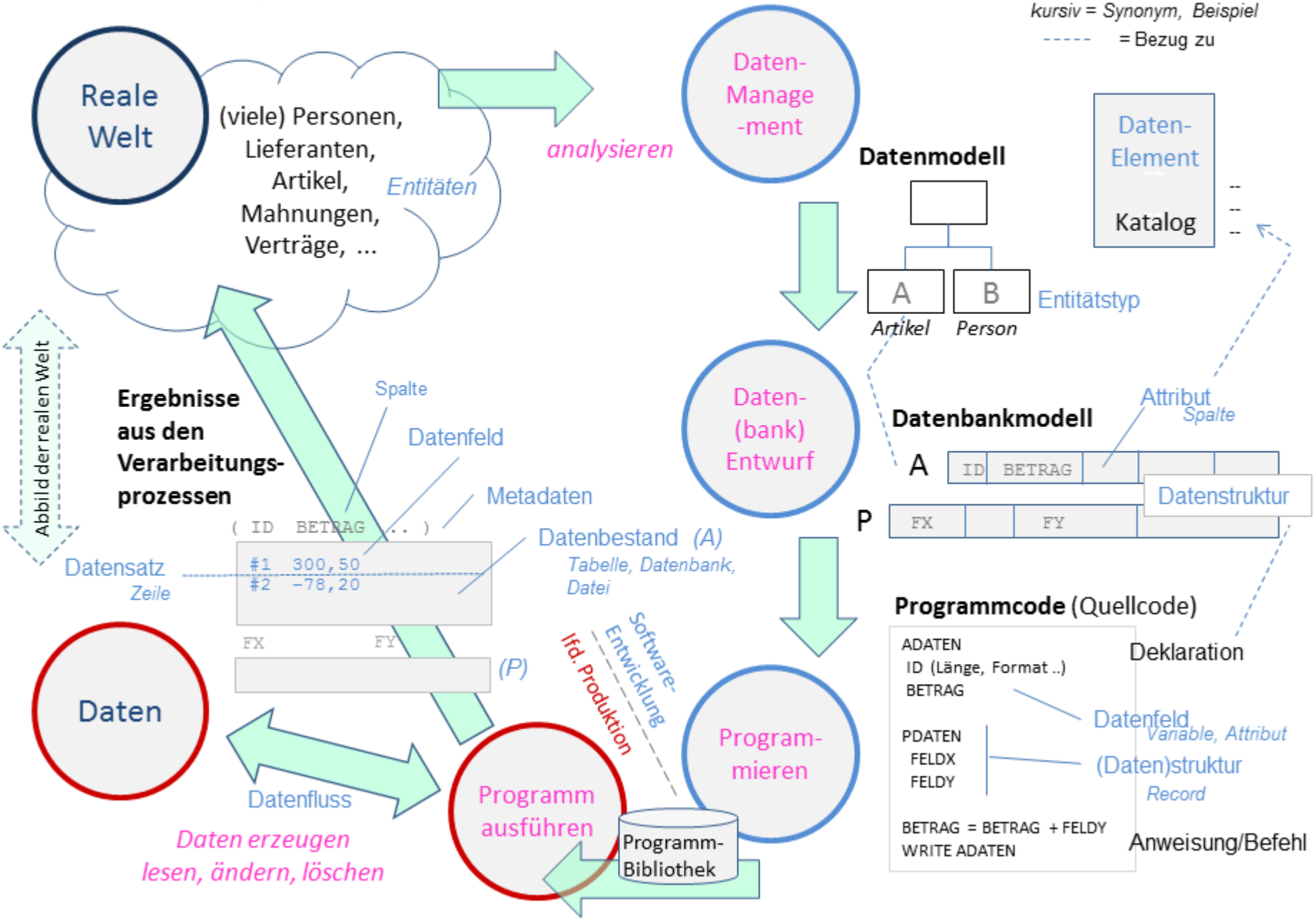
- Verwende Toolkit, das uns möglichst viel von der Programmierarbeit abnimmt

  - vorgefertigte Programmbausteine
  - Beispiele: mathematica, matlab, origin, …..
  - Nachteil: proprietäre software mit eigener Syntax (Sprachinterface), (obwohl häufig auch eine Anbindung an höhere Programmiersprachen existiert)

- Vorlesung: verwende Toolkit, das Public Domain Software ist und ein interpretiertes Sprachinterface zu C++ und Python und zum C++ Compiler hat.

  ROOT Toolkit, das zur Analyse der LHC Daten entwickelt wurde / wird.

Datenbegriffe



Messungen und Messfehler

Messung

$$x_m = x_w + \Delta x$$

x_m : gemessener (angezeigter) Wert der Messgrösse

x_w : wahrer Wert der Messgrösse (nicht bekannt)

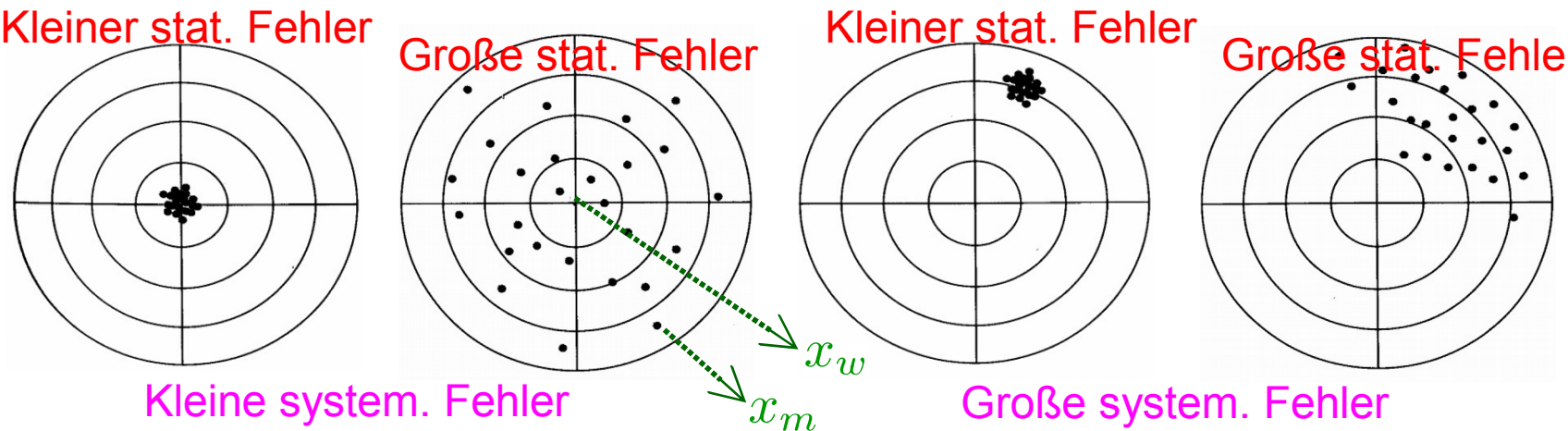
Δx : Messabweichung (Messfehler)

Messfehler

Eine Messung erfolgt immer nur mit endlicher Genauigkeit, 2 Beiträge:

- Systematische Fehler: Konstante, einseitg gerichtete Abweichung vom wahren Wert unter gleichen Messbedingungen.
- Zufällige oder statistische Fehler: Zufällige, nicht einseitig gerichtete Abweichungen vom wahren Wert. (Mittelwert M und Messunsicherheit s).

Beispiel: Messergebnisse einer Sternposition



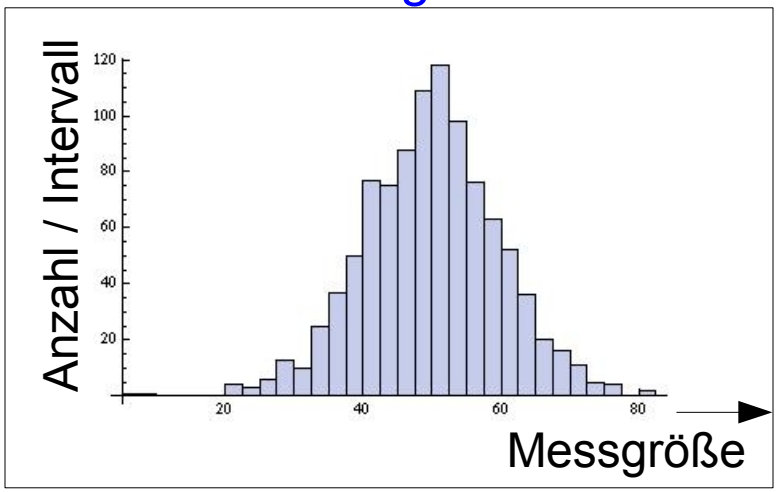
Fehlerrechnung

Quantitative Bestimmung

- Systematische Fehler:
  Schwierig! Genaue Analyse des Messaufbaues
- Zufällige oder statistische Fehler:
  Mehrfache Messung der selben Größe
- Schätzung des Messwertes bei mehrfacher Messung:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$ Arithmetrisches Mittel

Histogramm



- Fehler einer Einzelmessung

  Eigenschaft des Arithmetischen Mittels $\sum_{i=1}^{n} (x_i - \bar{x})^2 = min$

$$\sigma_E = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$$ Standardabweichnung

- Mittlerer Fehler des Mittelwertes
  Mittelwert von n Messungen ist um $\frac{1}{\sqrt{n}}$ genauer als die Einzelmessung

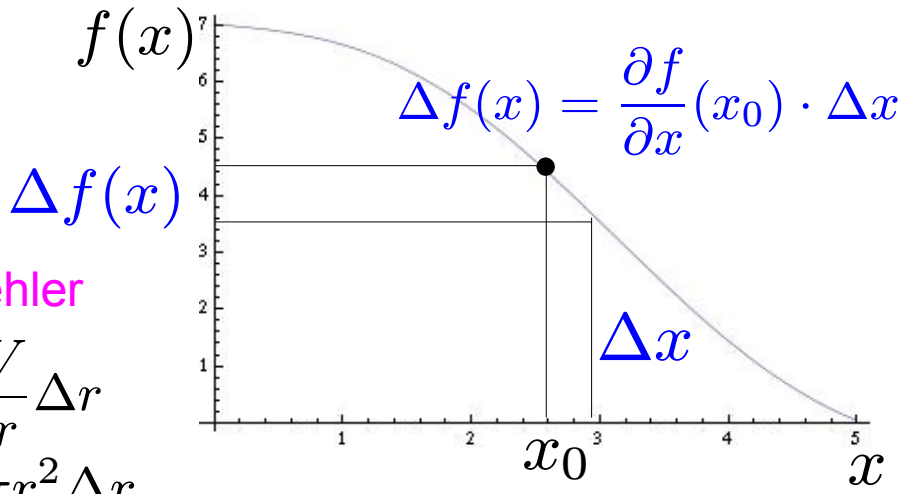
$$\sigma_M = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n(n-1)}}$$

Fehlerfortpflanzung

Fehler zusammengesetzter Größen

Wie wirkt sich ein gemessener Fehler auf eine zusammengesetzte Größe aus?

$$\Delta f(x) = \frac{\partial f}{\partial x}(x_0)\Delta x$$



Beispiel: Fehler von V bei gegebenem Fehler von r

$$V = \frac{4}{3}\pi r^3 \qquad \Delta V = \frac{\partial V}{\partial r}\Delta r$$

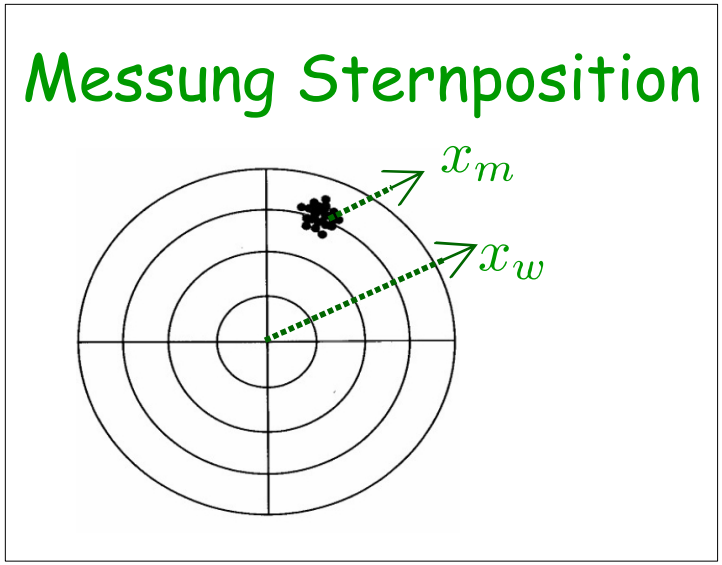
$$\Delta V = 4\pi r^2 \Delta r$$

Gauß'sches Fehlerfortpflanzungsgesetz

Wie wirken sich mehrere gemessene Fehler auf eine zusammengesetzte Größe aus?

$$\kappa = f(x, y) \qquad \Delta\kappa = \sqrt{(\frac{\partial f}{\partial x}\Delta x)^2 + (\frac{\partial f}{\partial y}\Delta y)^2}$$

Analyse der Messungen



Anforderungen an ein Computer-Programm zur Analyse von gemessenen Daten:

- Messdaten einlesen
- Kalibration / Berechnungen
- Datenanpassung / Bestimmung von Modelparametern
- Graphische Darstellung
- Simulationsrechnungen / theoretische Beschreibung und Vergleich mit den Messungen
- Ausgabe der Ergebnisse

1. Teil des Programms

- Einführung UNIX
  - Progammierwerkzeuge: Editor, Shell Commands, Compiler, Linker
  - CIP Pool
- C++ - ein Beispielprogramm
  - Typen, Variablen, Operatoren
  - Eingabe und Ausgabe
- C++
  - Bedingte Anweisungen und Schleifen
  - Arrays und Pointer
  - Funktionen und Klassen
- C++ - Informationen und Beispiele

  Wiki Book C++ Programmierung

  http://www.cplusplus.com/

  http://www.tutorialspoint.com/cplusplus/index.htm

Einführung in die Programmierung mit C++ , Bjarne Stroustrup, ISBN 978-3-86894-005

C++ Einleitung

- Mitte der sechziger Jahr wurde bei AT&T an Mehrbenutzer Betriebssytemen gearbeitet.
- Nach dem Rückzug von AT&T aus dem Großprojekt Multics entwickelten Thompsen und Ritchie Ende der sechziger Jahre eine erste in Assembler geschriebene Version, UNIX, auf einer PDP 7.
- Die Implementierung von UNIX in der von Ritchie entwickelten prozeduralen Programmiersprache C stellt einen Schlüsselbaustein unserer heutigen Computing Technologie dar.
- C++ wurde von Bjarne Stroustrup (Bell Laboratories) ab 1979 entwickelt um objektorientiertes Programmieren zu erleichtern.
- C++ ist eine Erweiterung der Sprache C.
- 1985 wurde aus „C with classes“ C++.
- GNU Compiler collection (gcc), ISO C++ standard von 1998, see http://www.open-std.org/jtc1/sc22/wg21/
- Der letzte offizielle Standard wurde 2011 festgelegt, ISO/IEC 14882:2011

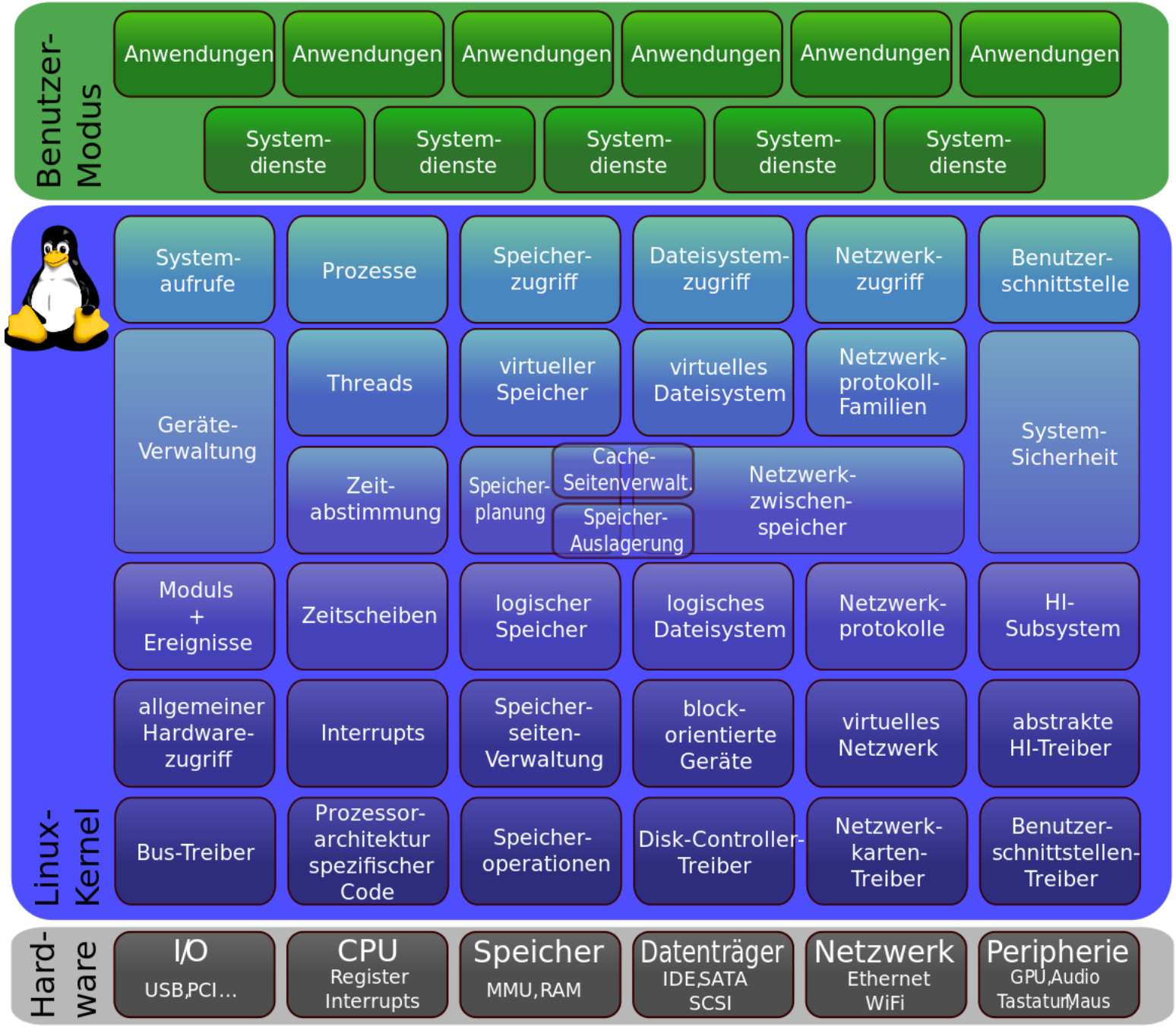
C++ Einleitung

- Computerprogramm
  Computer prozessieren Daten, führen Berechnungen durch, fällen Entscheidungen mit Hilfe eines Satzes von Anweisungen.
  Eine Aktion wird mit Hilfe von Schlüsselwörtern beschrieben, die durch eine Programmiersprache, z. B. C++, charakterisiert werden.
  Die Schlüsselwörter werden im Klartext in einer Datei gespeichert und von einem Übersetzungsprogramm (Compiler) in Maschinensprache umgewandelt
- Wir benutzen als C++ Compiler die öffentlich verfügbare GNU Compiler Collection (gcc)
- Im Rahmen dieses Kurses wird keine Entwicklungsumgebung verwendet. Wir arbeiten auf der shell in der Linux Umgebung des CIP pools.
- Arbeiten Sie zu zweit, dadurch werden Fehler beim Implementieren der Programme reduziert.
  Es ist ok Programme oder Programmteile zu kopieren, aber
  - Verwenden Sie nur code, den Sie auch verstehen
  - Urheberrechte beachten (GNU General Public License)

Unix/Linux Einführung

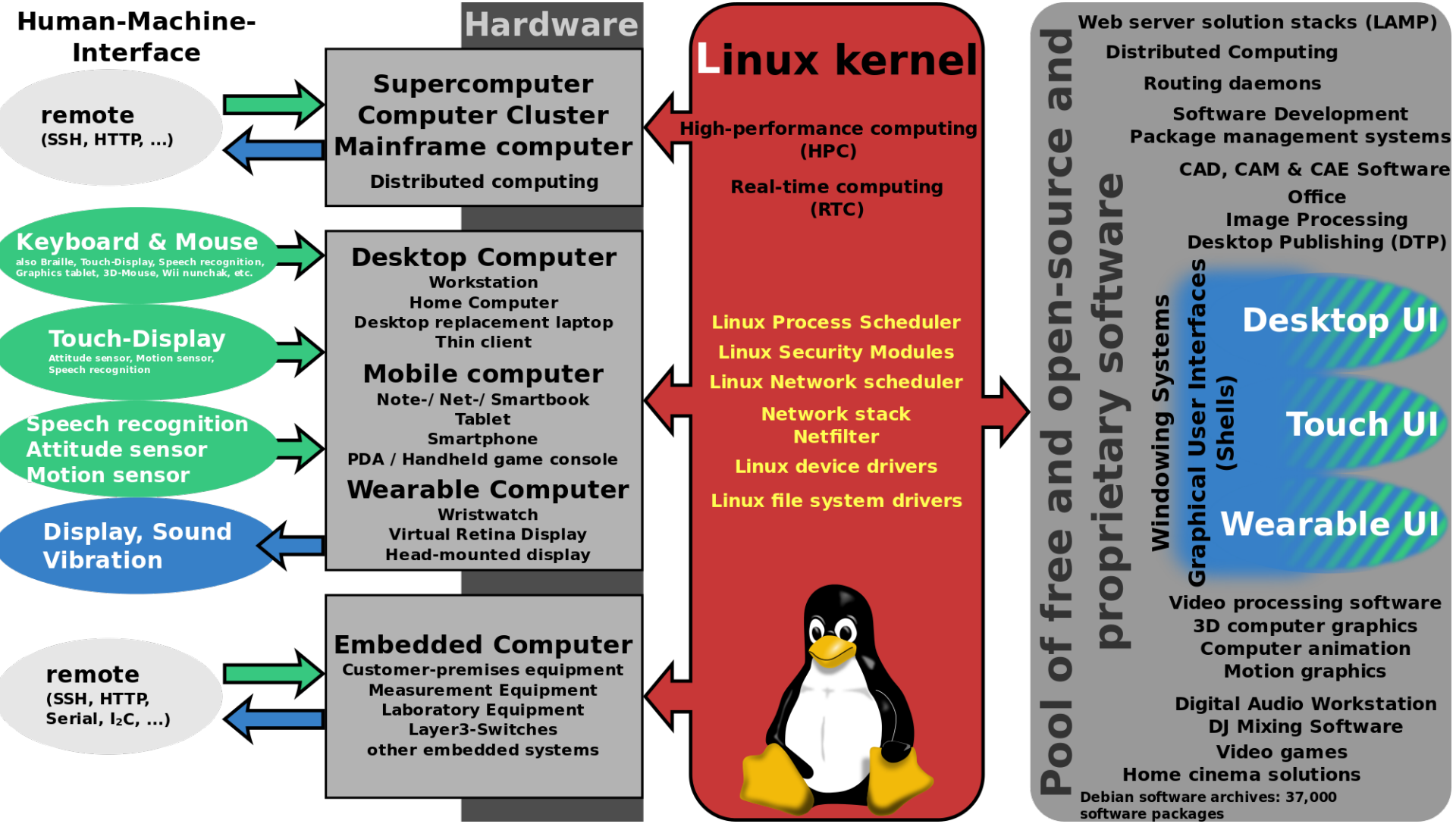
"...the number of UNIX installations has grown to 10, with more expected..."
- Dennis Ritchie and Ken Thompson, June 1972 (Bell Lab.)

- 1973 release von Unix V4, die in die Programmiersprache C portierte Version von Unix (frei verfügbar).
- 1987 schrieb Andrew S. Tanenbaum Minix als Lehrbetriebsystem, weil der Unix source code nicht mehr frei verfügbar war.
- 1991 veröffentlichte Linus Torvalds unter der GNU General Public License (GPL) den ersten Linux Kernel für die x86 Architektur.
- Linux wird synonym für Linux Distributionen verwendet. Diese verwenden alle den Linux Kernel, der auf Grund des Lizensierungsmodels von einer Vielzahl von Entwicklern in internationaler Zusammenarbeit gewartet, verbreitet und weiterentwickelt wird. Die Linux Distributionen unterscheiden sich durch die mitgelieferten Dienstprogramme.

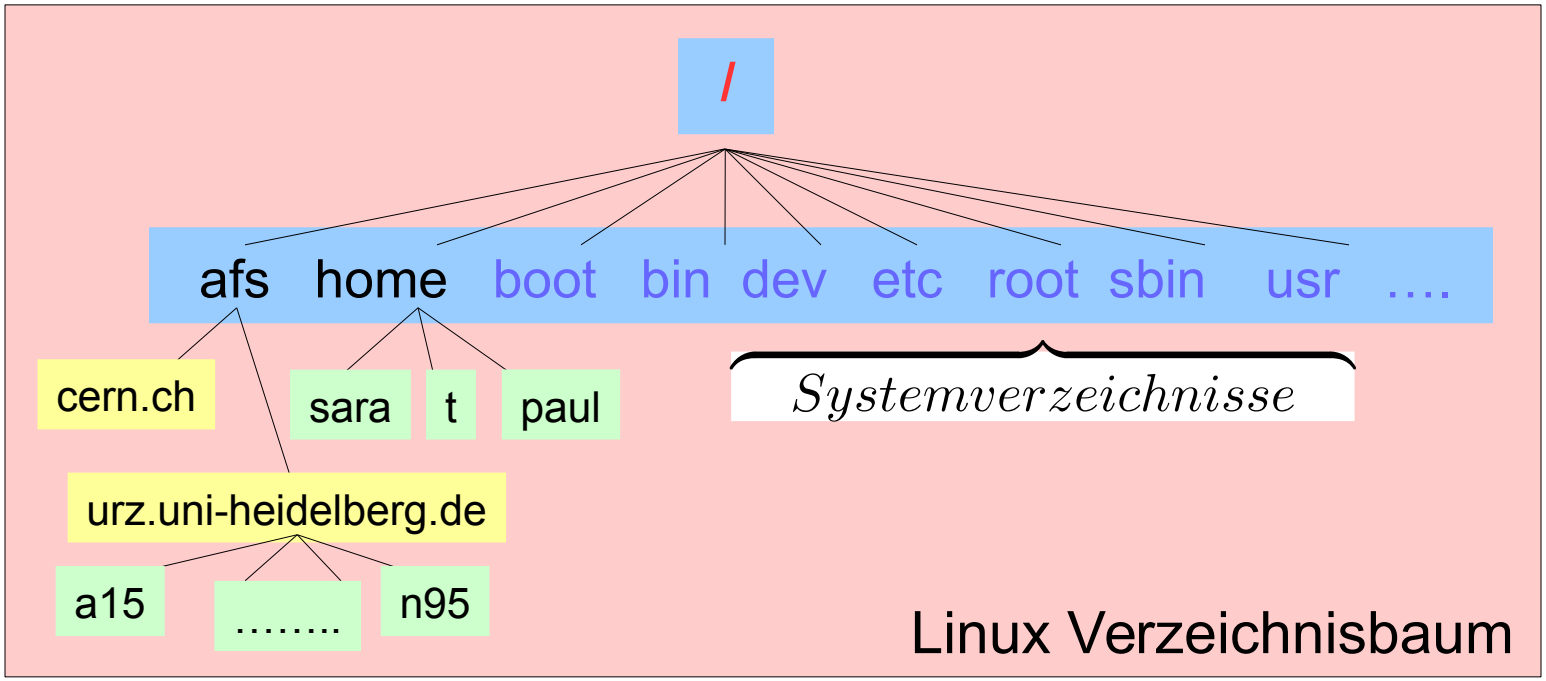

Benutzer-Modus
Anwendungen
Anwendungen
Anwendungen
Anwendungen
Anwendungen
Anwendungen
System-dienste
System-dienste
System-dienste
System-dienste
System-dienste
Linux-Kernel
System-aufrufe
Prozesse
Speicher-zugriff
Dateisystem-zugriff
Netzwerk-zugriff
Benutzer-schnittstelle
Geräte-Verwaltung
Threads
virtueller Speicher
virtuelles Dateisystem
Netzwerk-protokoll-Familien
System-Sicherheit
Zeit-abstimmung
Speicher-planung
Cache-Seitenverwalt.
Speicher-Auslagerung
Netzwerk-zwischen-speicher
Moduls + Ereignisse
Zeitscheiben
logischer Speicher
logisches Dateisystem
Netzwerk-protokolle
HI-Subsystem
allgemeiner Hardware-zugriff
Interrupts
Speicher-seiten-Verwaltung
block-orientierte Geräte
virtuelles Netzwerk
abstrakte HI-Treiber
Bus-Treiber
Prozessor-architektur spezifischer Code
Speicher-operationen
Disk-Controller-Treiber
Netzwerk-karten-Treiber
Benutzer-schnittstellen-Treiber
Hard-ware
I/O
USB,PCI…
CPU
Register
Interrupts
Speicher
MMU,RAM
Datenträger
IDE,SATA
SCSI
Netzwerk
Ethernet
WiFi
Peripherie
GPU,Audio
Tastatur,Maus

- Kernelgröße: 16 KB (1. Unix version, 1970) → 230 KB (1. Linux version, 1991) → 100 MB (Linux Kernel version, 2016)

- Linuxkernel Einsatzbereiche



Unix/Linux Einführung



- Interaktion mit Userprogrammen der Linux Distributionen erfolgt über eine graphische Oberfläche oder über die shell (Eingabe Fenster).
- Für unseren Kurs brauchen wir nur wenige shell Komandos.

Zusammenfassung einiger shell Kommandos mit link zu einem guten Online Tutorial:
http://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/UnixEinfuehrung.pdf
Mikroübersicht zum Arbeiten im CIP Pool:
http://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/cipEinfuehrung.pdf

Nützliche Linux Komandos

Linux Programme, die wir brauchen

- Hilfe unter Linux

`man <cmd>`	Beschreibung zu Komandos anzeigen
`info`	Linux Hilfe System anzeigen

- Suchen im Verzeichnisbaum oder im File

`find <dir> -name <filename>`	Suchen eines Files vom Directory <dir>
`grep pattern <file>`	Ausdruck pattern in <file> finden

- Andere Tipps
  - history erzeugt eine Liste der eingebenen letzten 1000 Befehle.
  - Mit den Pfeil Up/Down Tasten lässt sich in den bereits eingegebenen Komandos blättern.
  - !g führt das letzte mit g beginnende Komando aus.
  - Tabulator Taste ergänzt eine begonnene Eingabe
  - Kommando A | Komando B die Ausgabe von Kommando A wird als Eingabe an Komando B übergeben.
  - Strg + c bricht ein Komando ab, das gerade ausgeführt wird.
  - Komando & das Komando wird im Hintergrund ausgeführt.
  - Strg + z bg <return> ein Komando, das gerade ausgeführt wird, wird in den Hintergrund schickt.

Nützliche Linux Komandos

Linux Programme, die wir brauchen

- Noch mehr …….

- ps	zeigt Prozesse die in der shell ausgeführt werden
- ps -aux	zeigt Prozessliste
- wc -l <file>	zählt die Anzahl der lines in einem File <file>
- kill -9 <process id>	entfernen des Prozesses mit der id <process id>

Nützliche L

Linux Programme, die wir brauch

- Noch mehr …….

- ps — zeigt P
- ps -aux — zeigt P
- wc -l <file> — zählt d
- kill -9 <process id> — entfern

Arbeitsvorschläge:

- loggen Sie sich mit Ihrem userid ein.
- welchen Editor verwenden Sie?
- finden wir den compiler g++ ?
- haben Sie genügend Speicherplatz im AFS home.
- haben Sie Zugang zur Kurs Web page
- legen Sie ein directory an, indem Sie arbeiten werden und wechseln Sie in das directory.
- öffnen Sie hier ein Textfile und speichern es. Wie wird es gelöscht bzw. umbenannt?
- Zahl der aktiven Kernelmodule: lsmod | wc -l
- Zahl der zur Verfügung stehende Module: find /lib/modules/$(uname -r)/ -name '*.ko' | wc -l
- System Prozesse: top
- eigene Prozesse: ps
- man grep → suchen Sie einen string im Text File
- suchen Sie einen Prozess ihres Nachbarn.
- Kann man Prozesse beenden?

Schlüsselworte in C++

alignias	*char16_t*	dynamic_cast
alignof	*char32_t*	else
and	class	enum
and_eq	compl	explicit
asm	const	export
auto	*constexpr*	extern
bitand	const_cast	false
bitor	continue	float
bool	*decltype*	for
break	default	friend
case	delete	goto
catch	do	if
char	double	inline

Schlüsselworte in C++

int	protected	template	voatile
long	public	this	wchar_t
mutalbe	register	*thread_local*	while
namespace	reinterpret_cast	throw	xor
new	return	true	xor_eq
noexcept	short	try	
not	signed	typedef	
not_eq	sizeof	typeid	
nullptr	static	typename	
operator	*static_assert*	union	
or	static_cast	using	
or_eq	struct	virutal	
private	switch	void	

Ein Beispiel

myFirst.cc

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

Ein Beispiel

// oder /* */	Kommentare

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

#	Preprozessoranweisungen
include < >	File suchen und einfügen
<iostream >	Input/Output Definitionen

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

using namespace std ; Programm - namen kommen aus Standard C und C++ Bibliotheken

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

main() {C ++ Anweisungen} Haupt - programm Block, Anweisungen werden sequentiell ausgeführt

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

Definition von Variablen werden angegeben — Typ und Name

Variablen und Konstanten

- Syntax

```
Datentyp Name ;
```

- Typ Definitionen

Datentyp	Speicherbedarf
Ganze Zahlen	
`int`	16/32
`short int`	16
`long int`	32
`unsigned int`	16/32
`unsigned short int`	16
`unsigned long int`	32
`unsinged longlong int`	64

- Wertebereich ist durch Speichergrösse gegeben und betriebssystemabhängig.

short int ≤ int ≤ long int

Datentyp	Speicherbedarf
Reelle Zahlen	
`float`	32
`double`	64
`long double`	64
Buchstaben und Zeichen	
`char`	16/32
`unsigned char`	16/32
Logische Zeichen	
`bool`	

Variablen und Konstanten

Wertebereich einer durch n bit dargestellten Integer Variablen:

$-2^{n-1}, ...0....2^{n-1} - 1$

das Vorzeichen ist im Bit am linken Rand kodiert

Bei Operationen mit Überschreitung des Wertebereiches kommt es zu falschen Ergebnissen (undefiniertem Verhalten) ohne Fehlermeldung.

Zum Wertebereich der Variablen erhält man Zugang mit Hilfe von Funktionen, deren Definitionen wir hinzufügen müssen.

```
#include <limits> ;
```
http://www.cplusplus.com/reference/limits/numeric_limits/

```
…..
cout << numeric_limits <int>::min() << endl;
```

oder wie in C üblich

```
#include <climits> ;
```
http://www.cplusplus.com/reference/climits/

```
…..
cout << MIN_INT << endl;
```

Variablen und Konstanten

`double` und `float` werden als 64 und 32bit Gleitkommazahlen dargestellt.

Darstellung der Zahlen: $r = (-1)^V \cdot 2^{exp-127} \cdot 1.mant$

Zum Wertebereich, den Limits der Exponenten und weiteren Infos erhält man Zugang mit Hilfe von Funktionen, deren Definitionen wir hinzufügen müssen.

```
#include <limits> ;http://www.cplusplus.com/reference/limits/numeric_limits/
…..
cout << numeric_limits <double>::epsilon() << endl;
cout << numeric_limits <double>::round_error() << endl;
……..
```

Abweichung zwischen 1 und dem nächst grösseren Wert

Maximaler Rundungsfehler

Variablen und Konstanten

Ermittlung des Speicherbedarfs von Variablen oder Typen

- Syntax

```
sizeof ( Variable ) ;
sizeof ( Datentyp ) ;
```

Mit dem Zeichen = werden Variablen Werte zugewiesen

- Syntax

```
Datentyp Name = Wert ;
```

Notwendig, sonst ist der Speicherinhalt zufällig !

Mit dem Schlüsselwort const werden konstante Variable definiert. Der Wert wird zur Compile Zeit zugewiesen und darf nicht mehr geändert werden.

- Syntax

```
const Datentyp Name = Wert ;
```

Für Zeichenketten werden weitere Definitionen gebraucht, string

- Syntax

```
#include <string> ;
const string ZeichenKette = ”Strings bestehen aus char“;
```

Variablen und Konstanten

Umwandlung von Typen während der Ausführung des Programms erfolgt automatisch oder mit Anweisungen.

Die automatische Konversion ist Compiler spezifisch und sollte vermieden werden !

- Syntax

```
int myInteger ;
float myFloat ;
myFloat = (float) myInteger ;

static_cast<type> expression ;
```

Automatische Konversion – Beispiele :

double / float zu int	Nachkommastellen werden abgeschnitten ohne zu runden
char zu int	Character werden mit einer Integer Zahl entsprechend des Ascii Zeichensatzes dargestellt. Bei einer Konversion werden diese Zeichen dargestellt

ASCII-Zeichentabelle

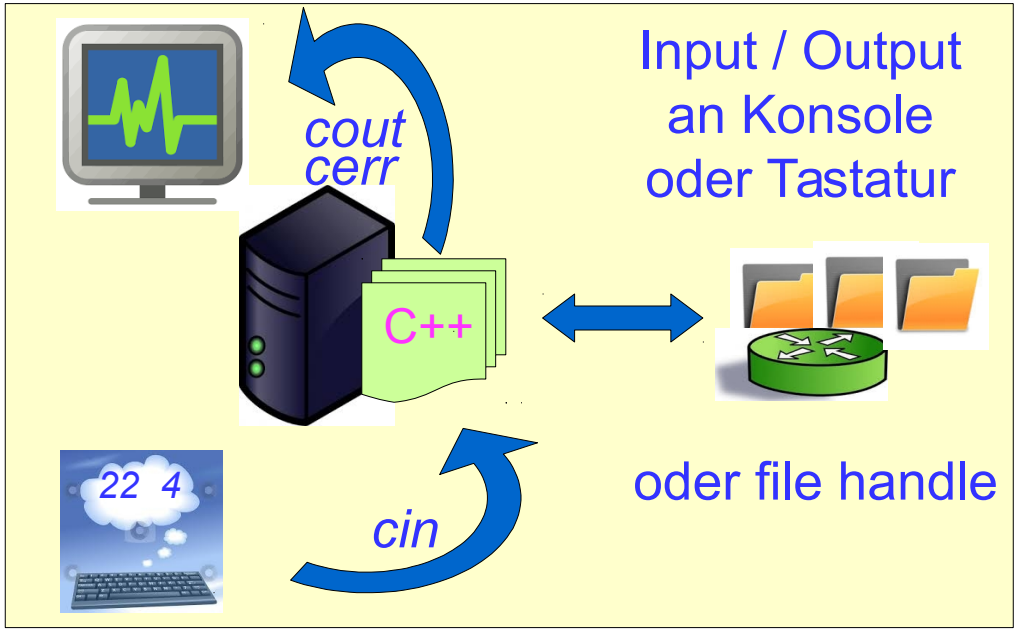
0 =	18 = ↕	36 = $	54 = 6	72 = H	90 = Z	108 = l
1 = ☺	19 = ‼	37 = %	55 = 7	73 = I	91 = [	109 = m
2 = ☻	20 = ¶	38 = &	56 = 8	74 = J	92 = \	110 = n
3 = ♥	21 = §	39 = '	57 = 9	75 = K	93 =]	111 = o
4 = ♦	22 = ▬	40 = (	58 = :	76 = L	94 = ^	112 = p
5 = ♣	23 = ↨	41 =)	59 = ;	77 = M	95 = _	113 = q
6 = ♠	24 = ↑	42 = *	60 = <	78 = N	96 = ‘	114 = r
7 = •	25 = ↓	43 = +	61 = =	79 = O	97 = a	115 = s
8 = ◘	26 = →	44 = ,	62 = >	80 = P	98 = b	116 = t
9 = ◇	27 = ←	45 = -	63 = ?	81 = Q	99 = c	117 = u
10 = ◙	28 = ∟	46 = .	64 = @	82 = R	100 = d	118 = v
11 = ♂	29 = ↔	47 = /	65 = A	83 = S	101 = e	119 = w
12 = ♀	30 = ▲	48 = 0	66 = B	84 = T	102 = f	120 = x
13 = ♪	31 = ▼	49 = 1	67 = C	85 = U	103 = g	121 = y
14 = ♫	32 =	50 = 2	68 = D	86 = V	104 = h	122 = z
15 = ☼	33 = !	51 = 3	69 = E	87 = W	105 = i	123 = {
16 = ►	34 = "	52 = 4	70 = F	88 = X	106 = j	124 = ¦
17 = ◄	35 = #	53 = 5	71 = G	89 = Y	107 = k	125 = }

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

cout / cin Ausgabe und Eingabe von Variablen oder strings

Input / Output



- Funktionsdefinitionen in

```
#include <iostream>;
```

- Daten werden in einen output stream geschrieben, gepuffert und an die Konsole gesendet.

  - Output stream: cout
  - Input stream: cin
  - Output stream für Fehler: cerr

- Syntax

```
cout <<  output string << Variable << endl;
cin >> Variable1  >> Variable2 ;
```

- Beispiel:

```
const double Pi = 3.14, d = 12.0 ;
double myResult;   myResult = Pi * d ;
cout << "Umfang = " << myResult << endl;
```

Viele Optionen für Formatierungen und Steuerzeichen: #include <iomanip>

Input / Output

- Mit Hilfe von Manipulatoren lässt sich die Ausgabe formatieren. Die Manipulatoren werden in den Ausgabeströmen zwischen die Umleitungsoperatoren eingefügt.

```
#include <iomanip>;
using namespace std ;
main(){
……
cout << setw(8) << i << endl;
cout << setfill('-');
cout << left << setw(8) << i << endl;
```

Details sind unter diesem link gut erklärt

http://www.willemer.de/informatik/cpp/iostream.htm

- Eine alternative und weitreichendere Formatierung lässt sich mit Hilfe von C print Befehlen erreichen.

http://www2.informatik.uni-halle.de/lehre/c/c_printf.html

```
printf(format_string,ausgabe_liste);
sprintf(zeichenkette,format_string,ausgabe_liste);
fprintf(dateizeiger,format_string,ausgabe_liste);
```

Steuerzeichen (Escape-Sequenzen)

- Nicht druckbare Zeichen des gewählten Zeichensatzes.
- Beginn mit einem umgekehrten Schrägstrich (Backslash).
- Datentyp char.

… in C++

Escape-Sequenz	Erläuterung
\b	Backspace. Einen Schritt zurück.
\f	Form Feed. Seitenvorschub
\n	New Line. Zeilenvorschub
\r	Carriage Return. Wagenrücklauf
\t	Horizontaler Tabulator
\v	Vertikaler Tabulator
\“	Anführungszeichen
\'	Apostroph
\\	Backslash. Umgekehrter Schrägstrich
\0	Ende-Zeichen eines C-Strings.

Ein Beispiel

```
// Add 2 Integer typed in by the user via keyboard

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" << endl;

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

sum = a + b Zuweisung von Ausdrücken

return 0 Rückgabewert des Programms

Operatoren

Operatoren verknüpfen Variable zu neuen Ausdrücken, wir unterscheiden

- Arithmetische Operatoren
  Berechnung von Werten
- Bit Operatoren
  Manipulation einzelner Bits
- Vergleichsoperatoren
  Überprüfung von Aussagen
- Logische Operatoren
  Verknüpfung von Aussagen

Operatoren	Berechnungen
*	Multiplikation
/	Division
%	Division mit Rest
+	Summe
-	Differenz

Operatoren	Berechnungen
*=	Multiplikation
/=	Division
%=	Division mit Rest
+=	Summe
-=	Differenz

Berechnung wird mit den linken Operatoren durchgeführt und verändern dann den links stehenden Wert.

```
P += 7; →  P=P+7
P += 7+2; → P=P+7+2
```

N++	N um 1 erhöhen
m- -	m um 1 verkleinern

- Benutzung von Klammern wie in der Mathematik

Operatoren

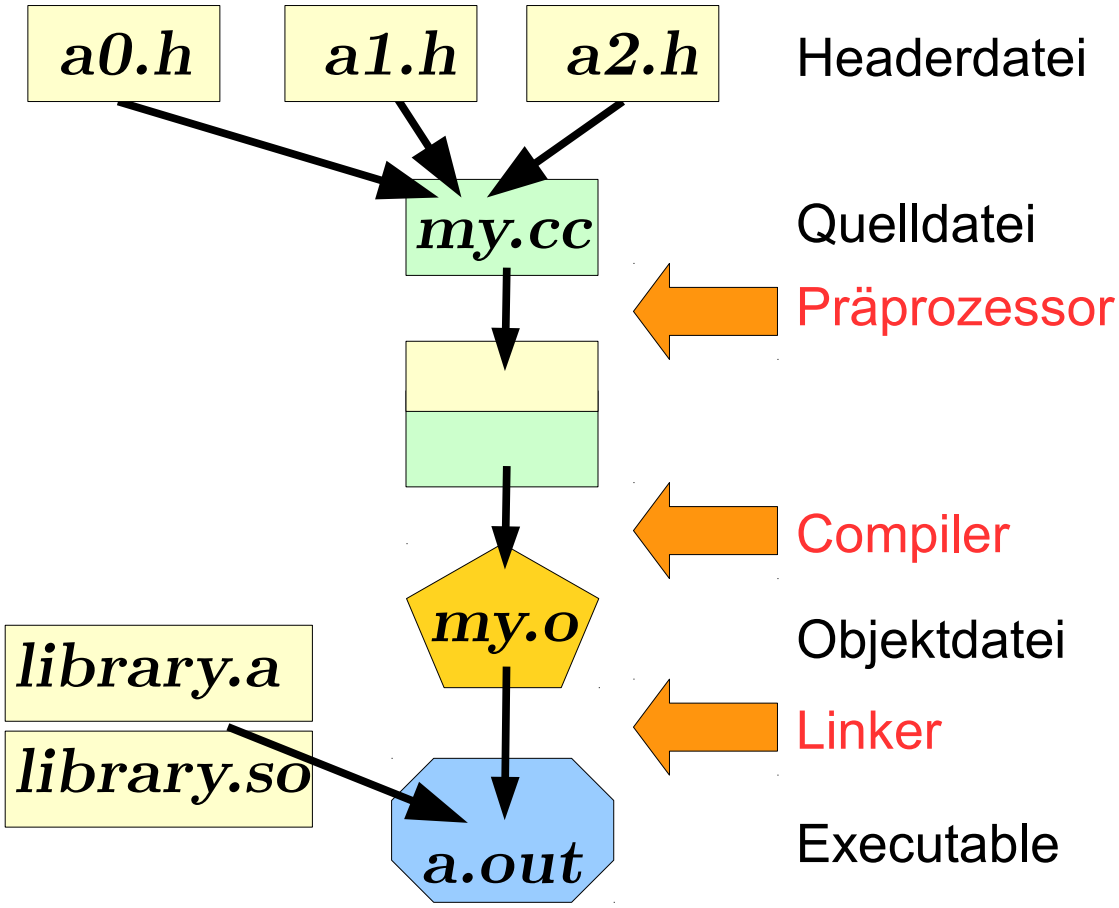
Operatoren verknüpfen Variable zu neuen Ausdrücken, wir unterscheiden

- Arithmetische Operatoren
  Berechnung von Werten
- Bit Operatoren
  Manipulation einzelner Bits
- Vergleichsoperatoren
  Überprüfung von Aussagen
- Logische Operatoren
  Verknüpfung von Aussagen

Operatoren	Bit Operationen
~	nicht
&	und
|	oder
^	entweder oder
>>	nach rechts verschieben
<<	nach links verschieben

Ein ausführbares C++ Programm

Die Erzeugung eines ausführbaren C++ Programmes erfolgt in 3 Schritten:



Daten zur Struktur und Definitioen des Programms

Enthält Anweisungen zum Ablauf und Verhalten des Programms

Aus dem C++ Quellcode übersetzter Maschinencode

Alle vom Linker zusammen-gebundenen Objektdateien bilden das ausführbare Programm

GNU Compiler Collection gcc

- Übernimmt folgende Aufgaben: preprocessing, compilation, assembly and linking
- Verhalten wird durch Optionen in Form von Flags und File-Namen gesteuert

  - optionen haben „multi letter names“, daher nicht wie in Unix Befehlen: -dv $\neq$ -d -v
  - File-Name Endungen bestimmen welcher compiler aktiviert wird
  - Details auf der shell mit dem Befehl `man gcc` oder ausührlicher `info gcc` (falls die Hilfen installiert sind)

- Sprachoptionen:
  - c c-header cpp-output
  - c++ c++-header c++-cpp-output
  - objective-c objective-c-header objective-c-cpp-output
  - objective-c++ objective-c++-header objective-c++-cpp-output
  - assembler assembler-with-cpp
  - ada
  - f77 f77-cpp-input f95 f95-cpp-input
  - go
  - java

→ g++

- Verwendung von g++ setzt Standard Library Pfade für den Linker

GNU Compiler Collection gcc

- Übersetzen eines C++ Programmes

```
g++ myFirst.cc        →    a.out
```

- Nützliche Optionen von g++
  - Programm Namen vergeben

```
g++ myFirst.cc -o MyProgramm        →    MyProgramm
```

  - Compiler Warnungen einschalten

```
g++ -Wall -Wextra myFirst.cc -o MyProgramm
```

  - Nur Compiler verwenden ohne Linker

```
g++ -c myFirst.cc -o myfirst.o        →    myFirst.o
```

  - Abschalten der nicht standard-conformen Compiler Optionen von g++

```
g++ -ansi myFirst.cc
```

  - Warnungen bei nicht standard-conformen Erweiterungen von g++

```
g++ -pedantic myFirst.cc
```

  - Automatische Optimierungen des Programms

```
g++ -Ox myFirst.cc     x [1,2,3]
```

Ein Be

```
// Add 2 Integer typed in by the user via

#include <iostream>
using namespace std;

int main()

{

  int a,b,sum;

  cout << "Enter integers to be added:" <

  cin >>  a  >> b ;

  sum = a + b ;

  cout << "The sum is " << sum << endl ;

  return 0 ;

}
```

Arbeitsanweisungen:

- loggen Sie sich mit Ihrem userid ein.

- erzeugen Sie ein Arbeits – Directory.

- starten Sie eine Editor, z.B. emacs

- schreiben Sie unserem Beispiel entsprechend ein C++ Programm und speichern sie es als myFirst.cc

- Übersetzen Sie Ihr File und erzeugen Sie ein ausführbares Programm mit g++ myFirst.cc -o myFirst

- Probieren Sie es mit ./myFirst

- Bauen Sie C++ Syntaxfehler in myFirst.cc ein, übersetzen Sie das File und suchen Sie nach Hinweisen auf den Fehler.

Vergleiche gcc und g++:

- Probieren Sie `g++ myFirst.cc -o myFirst` und `gcc myFirst.cc -o myFirst`

- fügen Sie link Pfade in gcc hinzu um die Fehlermeldungen zu entfernen
  `gcc myFirst.cc -L/usr/lib/gcc/x86_64-linux-gnu/4.9/ -lstdc++ -o myFirst`
  -L setze Pfadnamen ↘ Pfad ist installationsabhängig!
  -l setze Bibliotheksnamen (dabei wird das beginnende lib weggelassen)

- Sprache wird durch die Endungen festgelegt
  mv myFirst.cc myFirst.kk
  `gcc myFirst.kk -L/usr/lib/gcc/x86_64-linux-gnu/4.9/ -lstdc++ -o myFirst`
  → Compiler Fehler

  Hinzufügen von `-x c++`
  `gcc -x c++ myFirst.kk -L/usr/lib/gcc/x86_64-linux-gnu/4.9/ -lstdc++ -o myFirst`
  aktiviert den c++ Teil von gcc
  → kein Compiler Fehler

- Weitere Optionen mit der Online Hilfe auf der shell
  `man gcc`