Dinge, die du wissen musst...
aus unserem Informatik-Unterricht

Lukas Prokop

30. April 2009

Inhaltsverzeichnis

1 Begriffe

Algorithmus Schrittweise Anleitung zur Lösung eines Problems. Am besten in endlichen Schritten und mit geringem Ressourcenverbrauch.

Programmiersprache Formal strikt definierte Sprache, die eine Schnittstelle zwischen natürlicher Sprache und der Maschinensprache bildet. Erlaubt verschiedene Programmierparadigmen. Das Resultat der Formulierung eines Algorithmus in einer Programmiersprache nennt man Quelltext. Programmiersprachen lassen sich in verschiedene Generation einteilen. Beispiele für Programmiersprachen sind Fortran, Algol, COBOL, LISP, Logo, Simula, BASIC, Smalltalk, ... oder modernere Sprachen sind ECMA-Script (JavaScript), Java, C#, C++ und PHP

Programm Formulierung eines Problems in einer Programmiersprache

Prozess Abbild eines Programms im Hauptspeicher, um es zu exekutieren. Es verbraucht dafür CPU-Leistung und Speicherressourcen.

Datenbank Eine Datenbank verbindet Daten reihenweise in Zeilen. Die Definition entspricht damit einer Tabelle, wobei die Tabelle auf die Darstellung hinweist und eine Datenbank vor allem das Speichern von riesigen Datenmengen und die einwandfreie Wiedergabe betont

2 Prozessverwaltung

Hauptzustände eines Prozesses] 3 Hauptzustände:

- **Wartend:** Prozess wartet auf freie Ressourcen
- **Laufend:** Der Prozess wird gerade abgearbeitet
- **Unterbrochen:** Der Prozess wurde von einem anderen Prozess unterbrochen

3 Linux Filesystem Grundbefehle

cd change directory; wechselt das Verzeichnis

rm remove; löscht Datei (Option r steht für rekursiv, also auch alle Unterverzeichnisse)

mv move; verschiebe Datei von ... nach ... (Option r steht für rekursiv)

rmdir remove directory; löscht Ordner

pwd print working directory; gib den Namen des aktuellen Verzeichnisses aus

mkdir make directory; erzeuge einen neuen Ordner

touch Setzt Bearbeitungsdatum (etc) auf das aktuelle, falls bereits existiert. Sonst wird leere Textdatei erzeugt

chmod change mode; verändert Zugriffsrechte auf Datei

Die Manuals zu allen Befehlen können mit `man <command>` oder `info <command>` nachgeschlagen werden.

4 My name is Linus Torvalds and I am your god

1991 begann die Entwicklung eines eigenen freien Betriebsystemkernels. Er stellte sein System seiner Schwester Sara vor, indem er ihr sie fragte "Es sieht wie DOS aus, oder?". Sie nickte. Doch als Linus eine Funktionstastenkombination drückte, zeigte er einen anderen Eingabeprompt. Dort konnte er sich mit einem anderen Benutzernamen nochmals einloggen. Sara glaubte, dass Linus etwas Phantastisches geschaffen hatte, hatte aber kein Problem damit, da sie Nummer Eins am Billardtisch blieb. Am 25. August 1991 schreibt Linus in die Newsgroup comp.os.minix

> Hello everybody out there using minix -
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional lik gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
> ...
> Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus hat damit Minix (einer UNIX-Distribution) den Kampf angesagt. Ebensowenig hielt er von den Ideen sein geistiges Werk irgendwie zu vermarkten. Als junger Student wollte er nur, dass jeder damit lernen kann, Ideen einbringt und das passende Betriebsystem für sich schaffen kann. Als ein Resultat schloß er sich mit Richard Stallman (dem Gründer von GNU) (GNU ist ein Backronym und steht für "GNU is not UNIX") zusammen und stellte seine Werke unter die GPL ("General Public License"), die eine freie Weitergabe ermöglicht, aber eine kommerzielle Weiternutzung verbietet.

Wichtig ist, dass Linux nur ein Kernel ist. Der Kernel bildet die Schnittstelle zwischen Software und Hardware (zB die Software sagt "Speichere die Datei" und die Hardware macht es – der Kernel koordiniert solche Vorgänge). Linux bezeichnet den Kernel und die Anwendungsprogramme (zB der Texteditor vi, der C-Compiler gcc) stammen von GNU. Somit ist der korrekte Name von Linux Linux/GNU. Richard Stallman besteht auf

die korrekte Ausdrucksweise von Linux/GNU, sowie auch die Unterscheidung zwischen Freier Software und OpenSource.

Linux ist heute vollkommen frei verfügbar. Den Linux-Kernel beaufsichtigt noch immer Linus Torvalds (der für die Linux Foundation in den USA arbeitet). Linus bekam schnell Anhänger für seine Ideen. Sie kopierten die Quelltexte (Programmcodes) von Linus und modifizierten sie so, dass sie ihren eigenen Wünschen entsprachen. Um die Idee der "freien Software" zu unterstützen, stellten sie selbst wiederum die modifizierte Version online. Dadurch, dass ein reger Austausch zwischen Entwicklern stattfand, tauschten sie ihre Informationen, verbesserten Softwares und organisierten sich. Es steht ihnen komplett frei, aber manche Distributionen (also Abspaltungen von Linux-Software) konzentrieren sich beispielweise auf Sicherheit während sich andere auf Benutzerfreundlichkeit spezialisieren. Dadurch konnte Linux die verschiedensten Bereiche abdecken. Bis heute hat sich Linux nicht als Massenprodukt für Homecomputer etabliert (manche Linux-Entwickler sind der Meinung zum Glück), aber in Bereichen mit hohen Sicherheits- und Effizienzansprüchen (zB Webserver, Supercomputer) befindet sich Linux an der Spitze.

Bekannte Linux-Distributionen sind ubuntu, openSuSE, Debian, Fedora, RedHat, Knoppix, Arch Linux, gentoo und grml.

5 Imperative Programmierung

Das bekanntestes Programmierparadigma ist die imperative Programmierung. Die auszuführenden Befehle werden nacheinander notiert, um der Maschine zu erklären wie sie an ihr Ziel kommt. Es werden bedingte Anweisungen und Schleifen erlaubt, um die Turing-Vollständigkeit zu ermöglichen. Bedingte Anweisungen werden nur durchgeführt, wenn ein Problem zuvor auf den boolschen Wert `wahr` geprüft wurde (zB die vom User eingegebene Zahl ist kleiner 4 – Wahr oder falsch). Schleifen wiederholen Anweisungen mit unbekannter Laufanzahl (while), mit unbekannter Laufanzahl wobei es mindestens einmal durchgeführt wird (do... while), für eine Anzahl von Elementen (foreach oder for) oder mit bekannter Laufanzahl (for). Weitere Arten sind eher selten vertreten.

6 Prozedurale Programmierung

In der prozeduralen Programmierung (ein Programmierparadigma) werden Befehlsblöcke in Unterprogrammen (vor allem Prozeduren) eingeteilt. Dadurch kann leichter der Code wieder verwendet werden ohne ihn wieder schreiben zu müssen. Zum Beispiel können wir eine Funktion `test` definieren, die die Summe zweier Zahlen berechnet (also übergibt man ihr auch 2 Argumente). Diese Funktion schreiben wir möglichst weit oben am Anfang des Programms (oder gar in einer separaten Datei) (dient nur der Übersichtlichkeit). Nun können wir jederzeit mit dem Schlüsselwort `test` wieder eine Funktion aufrufen, die uns aus zwei Argumenten die Summe ausrechnet.

Der Vorteil liegt darin Probleme in kleinere Teilprobleme formulieren zu können und die Teilprobleme immer mit gleichen Komponenten lösen zu können. Die Programmiersprachen Delphi, PASCAL, VBScript (und Ähnliche) machen eine klare Unterscheidung zwischen Prozeduren (geben keinen Wert zurück) und Funktionen (mit Rückgabewert). Andere Programmiersprachen machen keine Unterscheidung und verwenden zur Definition von Funktionen und Prozeduren das selbe Schlüsselwort (zB `Sub`, `function`, `def`). Unser Beispiel oben wäre unbedingt als Funktion zu definieren, da wir die Summe zurückgeben müssen. Das Schlüsselwort `return` hat sich als Befehl durchgesetzt, Werte am Ende einer Funktion zurückzugeben.

```
#!/usr/bin/env python

def test(zahl1, zahl2):
    summe = zahl1 + zahl2
    return summe

a = test(1, 2) # use function with arguments 1 and 2
b = test(2, 4) # use function test multiple times
```

7 Objektorientierte Programmierung

In der OOP werden Daten in logische Einheiten zusammengefasst. Man möchte sich dadurch (wie auch bei jedem anderen Programmierparadigma) der Realität annähern. In der Welt kann man beinahe alles in Objekten wahrnehmen. Ein Auto wurde von Ford produziert, hat 200 PS und kann beschleunigen und bremsen. Ein Objekt besteht aus einer Identität (zB Name, Produzent), seinen Eigenschaften (zB Leistung, Anzahl der Sitze) und den Methoden (zB Bremsen, Beschleunigen, Einparken). Nun genießt man durch diese Auffassung verschiedene Vorteile. Wir können ein Grundmodell definieren und davon besondere Typen ableiten ("Vererbung"). Ein LKW ist auch ein Auto, besitzt aber mehr Leistung und bekommt noch zusätzlich die Methode Anhänger abkoppeln. Ebenso muss das Objekt sich selbst in kWs oder PS angeben können. Es muss ihm egal sein welchen Datentyp es zu verarbeiten hat ("Polymorphismus"). Und als letztes darf nicht verraten werden, dass es sich um einen Geldtransporter handelt. Vor allem darf niemand erfahren, wieviel Geld transportiert wird ("Kapselung").

Logische Einheit bestehen aus ...

- Identität (identity)
- Eigenschaften / Attribute (property)
- Methodes (methods, functions, procedures)

Folgende Features werden dadurch vereinfacht:

•

8 Rekursion

Um Rekursion zu veranschaulichen gibt es viele Möglichkeiten. Mein Liebling ist die Definition aus einem Wörterbuch:

> **Rekursion** (Re | kur | sion; engl. recursion) siehe Rekursion

Oder ein anderer Ansatz ist die Tatsache, dass man Rekursion verstehen muss, muss Rekursion zu verstehen.

Jetzt mal im Ernst... unter Rekursion versteht man eine Funktion, die sich selbst aufruft. Wichtig ist dabei eine saubere Termination der Funktion, da das Programm sonst unendlich weiterlaufen würde und den Hauptspeicher vollschreibt ("Stack Overflow"). Alle Probleme – die rekursiv formuliert werden können – können auch iterativ (sich Schritt für Schritt der Lösung des Problems nähernd) formuliert werden. Rekursion lässt sich auch auf Graphiken anwenden. Als Resultat erhält man Gebilde wie den Pythagoreischen Baum oder die Mandelbrotmenge.

Der Vorteil gegenüber der Iteration liegt in der leichteren Lesbarkeit. Ob ein rekursives oder iteratives Programm ein Problem effizienter löst, hängt von der Programmierung des Compilers und der Implementierung ab. Man unterscheidet verschiedene Arten von Rekursion. Der Satz "Dieser Satz ist unwahr!" wendet eine **direkte** Rekursion (die eigene Komponente wird direkt angesprochen) an. Der Satz "a ist unwahr. Die vorige Aussage ist unwahr" ist **indirekt** rekursiv. Bei der **Endrekursion** wird nach dem Aufruf der Rekursion kein weiterer Befehl angegeben, sondern nur mehr das Ergebnis zurückgegeben. Endrekursive Algorithmen sind wesentlicher effizienter als sonstige Rekursionen, da beim Abstieg (also von der Ausgangsfunktion in Richtung Ende der Rekursion) keine Variablen (oder sonstige Werte) aufgehoben werden, die beim Aufstieg (vom Ende zur Ausgangsfunktion) wiederverwendet werden. Bei der **multiplen** Rekursion werden mehrere Rekursionausrufe innerhalb einer Funktion getätigt. In einem Beispiel sieht man ein Quadrat, das an jeder Ecke wiederum ein Quadrat besitzt. Dies wird mit einer multiplen (bzw. 4-fachen) Rekursion ermöglicht.

Klassische Beispiele für die rekursive Problemlösung sind der Euklidische Algorithmus, die Türme von Hanoi und die Berechnung der Fakultät. Wird ein Problem iterativ statt rekursiv formuliert, braucht man meist eine while-Schleife (Schleife mit unbekannter Laufzeit). Im folgenden Beispiel wird die Fakultät iterativ wie auch rekursiv berechnet.

```
#!/usr/bin/env python

# iterative solution
def fac(num):
```

```
    counter = 1
    factorial = 1
    # the typical while loop
    while counter <= num:
        factorial = factorial * counter
        counter = counter + 1
    return factorial

# recursive solution
def fac_rec(num):
    if num == 1:
        return 1
    else:
        return fac_rec(num - 1) * num

# print 1*2*3*4*5 iterative and recursive
print fac(5)
print fac_rec(5)
```

9 Excel

9.1 Notation

Excel eignet sich zur Tabellenkalkulation und erlaubt Befehle, um Verweise und Berechnung zu ermöglichen. Wenn wir das Feld `B1` mit `=A1` definieren, beginnt das Gleichheitszeichen einen Befehl und `A1` ist die Notation um auf das Feld `A1` hinzuweißen. Also wird der Wert des Feldes `A1` in `B1` geschrieben. Um auf Tabellen (also zusammengehörige Felder, die in reckteckiger Form platziert sind) zu verweisen, können wir den Doppelpunkt verwenden. Wir nennen die zwei Eckfelder der Tabelle und trennen sie mit einem Strichpunkt. Zum Beispiel `A1:C3` bezeichnet die Felder `A1`, `A2`, `A3`, `B1`, `B2`, `B3`, `C1`, `C2` und `C3`.

Für den Ottonormal-Anwender wird es genügen das Gleichheitszeichen zu notieren und mit der Maus auf das gewünschte Feld zu klicken oder eine Tabelle aufzuziehen. Excel ermittelt die "Koordinaten" selbst. Genauso praktisch ist es, einen Befehl in Zelle `A1` definieren zu können und diesen dann mit dem kleinen Quadrat (wird im rechten unteren Eck der markierten Zelle angezeigt) nach unten aufziehen zu können. Sämtliche Verweise werden spalten- bzw. reihenweise (je nachdem in welche Richtung man den Befehl aufzieht) weitergezählt. Manchmal möchte man jedoch gar nicht weiterzählen, da sich einer Zelle (als Beispiel) ein Fixwert befindet, der in eine andere Währung umrechnet. In diesem Fall kann man die Notation $Spalte$Zeile verwenden. Beim Aufziehen verändert sich die Zellennotation nicht. Die folgende Tabelle illustriert alle verschiedenen besprochenen

	A	B	C
1	5	=A1	=(A1+$A$4)
2	2	=B1	=(A2+$A$4)
3	1	=C1	=(A3+$A$4)
4	10		

	A	B	C
1	5	1	=(SVERWEIS(B1; $A$5:$B$6; 2)*A1)
2	3	2	=(SVERWEIS(B2; $A$5:$B$6; 2)*A2)
3	4	2	=(SVERWEIS(B3; $A$5:$B$6; 2)*A3)
4			
5	1	10	
6	2	12	

Notationen.

9.2 Der SVERWEIS

Unter **SVERWEIS** versteht man den Befehl SuchVerweis, der auf Basis eines Wertes einen anderen Wert nimmt, der in einer fremden Tabelle definiert. Der SVERWEIS braucht 3 Argumente. Das erste Argument (Basis) bezeichnet das Feld aus dem der Wert entnommen werden soll. In der Tabelle, auf die in Argument 2 verwiesen wird, wird nach diesem Wert gesucht. Wird er gefunden, gibt der Befehl den Wert der Spalte Nummer (Argument 3) zurück.

In unserem Beispiel oben: In C1 sucht er das Feld B1. B1 enthält den Wert 1. In der Tabelle A5 bis B6 wird nach dem Wert 1 gesucht. In A5 befindet er sich. Die Tabelle A5:B6 hat als 2. Spalte die Spalte B definiert. Also geht er vom Wert A5 aus und geht dann in die 2. Spalte (also ins Feld B5). Diesen Wert gibt er zurück (10). Der Befehl in der Zelle C1 geht jedoch weiter. Dieser Wert 10 wird mit dem Inhalt des Feld A1 multipliziert. $10 \cdot 5 = 50$. Im Feld C1 befindet sich das Wert 50.

Die anderen Felder kann man sich selbst überlegen. Die richtigen Lösungen lauten 36 und 48.

10 SQL

Die Structured Query Language (SQL) ist eine bekannte Sprache, um Daten in Datenbanken zu verändern, zu löschen, zu bearbeiten oder hinzuzufügen. SQL dient somit meist zur Kommunikation zwischen einem Datenbankserver und einem Programm. SQL

findet bei relationalen Datenbanken Anwendung, wobei es verschiedene Dialekte (kleine Änderungen an den Funktionen) gibt.

SELECT fragt Datensätze ab. Nach dem SELECT folgt der Asterisk (*) zur Auswahl aller Spalten oder die jeweiligen Spaltennamen getrennt durch ein Kommata. Nach dem Schlüsselwort FROM (zur Angabe der Tabelle) folgt optional ein WHERE-Klausel, der eine Aussortierung von Datensätzen erlaubt (durch Prüfung der Datensätze auf Werte). Ein Beispiel wäre:
`SELECT name, work FROM computer_scientists WHERE name = 'Linus Torvalds'`

INSERT (INTO) Fügt einen zusätzlichen Datensatz an

UPDATE Verändert einen Datensatz. Ein Beispiel wäre:
`UPDATE computer_scientists SET salary = salary*2 WHERE name = 'FLOSS'`

DELETE (FROM) Löscht einen Datensatz

CREATE (DATABASE, TABLE) Erzeugt eine neue Tabelle oder Datenbank

DROP (DATABASE, TABLE) Lässt eine Tabelle oder Datenbank fallen (Löschen)

TRUNCATE (DATABASE, TABLE) "Leert" eine Tabelle oder Datenbank (entfernt Datensätze)

ALTER Ändert die Konfiguration einer Tabelle (zB fügt Spalten hinzu)

11 Protokolle

TCP Transmission Transfer Protocol – definiert wie Daten verbindungsorientiert (wenn Pakete nicht beim Ziel ankommen, werden sie nochmals gesandt, etc.) übertragen werden sollen

UDP User Datagram Protocol – verbindungslose Alternative zu TCP

IP Internet Protocol – ermöglicht es das Internet in kleinere Subnetze einzuteilen und identifiziert jeden Teilnehmer mit einer Adresse (IP-Adresse, Subnetzmaske, IPv4, IPv6)

DNS Domain Name Service – löst Adressen (zB google.com) in IP-Adressen auf

HTTP(S) Hypertext Transfer Protocol (Secure) – ist für die Übertragung von Hypertext (hauptsächlich Webseiten) zuständig

(S)FTP (Secure) File Transfer Protocol – einfacher Dateitransfer

SMTP Simple Mail Transfer Protocol – Versand von E-mails

POP(3) Post Office Protocol (Version 3) – Zugriff und Verwalten von E-mails am Mailserver

IMAP Internet Message Access Protocol – Zugriff auf E-mails

XMPP Extensible Message and Presence Protocol – freies Protokoll für Instant Messaging

Telnet Zugriff auf entferntem Rechner mittels eines terminals (remote terminal)

SSH Secure Shell – verschlüsselter Zugriff auf entferntem Rechner

ARP Address Resolution Protocol – Merkt sich Assoziation IP-Adressen zu Mac-Adressen

12 Dieses Dokument

Zur Erinnerung an unseren 4 schönen Informatikjahre
Frei zur Verteilung und Bearbeitung.
Auch ohne Autorangabe.
Written with LaTeX 2_ε

Lukas Prokop 2009