3 Basiswissen

Programmieren lernen
ISBN 3-8273-1951-X

Sie lernen in diesem Kapitel:

- wie ein Computer grundsätzlich arbeitet und welche Bedeutung das Betriebssystem, das BIOS und der Arbeitsspeicher für Programmierer besitzen,
- auf welche Weise Daten auf Speichermedien und im Arbeitsspeicher gespeichert werden,
- wie Programme geschrieben und für den Computer übersetzt werden,
- welche Anwendungsarchitekturen es gibt,
- was die wichtigsten Programmiersprachen voneinander unterscheidet,
- welche Bedeutung Algorithmen für die Programmierung haben,
- welche Strukturelemente in Algorithmen verwendet werden und
- welche Bedeutung Schleifen und Verzweigungen für die Darstellung eines Algorithmus haben.

Nachdem Sie in Kapitel 2 Ihre ersten Erfolge beim Programmieren hatten, behandelt dieses Kapitel einige Basisfragen, die für das Verständnis des Buchs wichtig sind. Es zeigt, wie ein Computer und dessen Programme grundsätzlich funktionieren, was ein Programm aus der Sicht des Computers eigentlich ist, unter welchen Architekturen und mit welchen Programmiersprachen Sie heutzutage Software entwickeln können und was ein Algorithmus ist.

3.1 Wie arbeitet ein Computer?

Dieses Buch ist kein Buch über Computer an sich. Also keine Bedenken: Ich gehe nicht im Detail auf die Arbeitsweise eines Computers ein. Ein Programmierer muss aber dessen Funktionsweise wenigstens grundlegend kennen. Deswegen beschreibe ich, was die Aufgabe eines Betriebssystems ist, wie ein Programm prinzipiell arbeitet und welche Rolle der Arbeitsspeicher dabei spielt. Ich gehe in diesem Abschnitt übrigens da-

von aus, dass Sie den grundsätzlichen Aufbau des Computers kennen (also wissen, was die Begriffe CPU, Arbeitsspeicher, Festplatte, CD-ROM-Laufwerk etc. bedeuten).

3.1.1 Das Grundprinzip: EVA

Das Grundprinzip eines Computers wird als EVA bezeichnet. EVA bedeutet „Eingabe, Verarbeitung und Ausgabe". Daten werden in verschiedenen Formen eingegeben, verarbeitet (berechnet) und wieder ausgegeben. Eingabegeräte sind z. B. die Tastatur, die Maus, die Festplatte, ein Scanner oder ein (DSL-)Modem. Ausgabegeräte sind z. B. der Monitor, ein Drucker, die Festplatte oder wieder ein (DSL-)Modem.

Wenn Sie in einer Textverarbeitung einen Text eingeben, werden Ihre Tastenbetätigungen (die Eingabe) vom Programm verarbeitet (analysiert) und auf dem Monitor und im Druck in Form von Zeichen ausgegeben. Wenn Sie die Datei speichern, führt Ihr Befehl (die Eingabe) dazu, dass die im Arbeitsspeicher gespeicherten Daten in eine Datei geschrieben werden. Öffnen Sie in derselben Textverarbeitung eine Datei, wird diese vom Datenträger gelesen (Eingabe), verarbeitet (korrekt interpretiert) und als Dokument auf dem Bildschirm ausgegeben. Immer wieder ist EVA im Spiel. EVA bedeutet also, dass Daten von einem Eingabemedium gelesen, verarbeitet und wieder ausgegeben werden.

EVAs überall

Obwohl EVA das Grundprinzip des Computers ist, müssen Sie sich bei der Programmierung allerdings kaum Gedanken um dieses Prinzip machen: Jedes Programm besteht automatisch aus mindestens einem, meist aber ziemlich vielen EVAs. Die einfache Konsolen-Nettoberechnung aus Kapitel 2 besitzt beispielsweise nur ein EVA (Bruttobetrag und Steuerwert eingeben, Nettobetrag ausrechnen, Nettobetrag ausgeben). Die Fenster-Variante besitzt aber schon zwei: eines, das auf die Betätigung des RECHNEN-Schalters reagiert, und eines für den BEENDEN-Schalter. Ein komplexeres Rechen-Programm wie der Windows-Taschenrechner besteht aus recht vielen einzelnen EVAs, die Ihre einzelnen Tastatur- oder Mauseingaben auswerten.

Normalerweise wissen Sie recht sicher, woher die Daten stammen, die verarbeitet und ausgegeben werden sollen, und wohin die Daten ausgegeben werden sollen, weil die Aufgabenstellung Ihres Programms dies so festlegt. Ein einfaches Programm zur Berechnung eines Bruttobetrags erwartet die Eingabe z. B. entweder direkt über die Tastatur oder in einem Steuerelement und gibt das Ergebnis an der Konsole oder in einem weiteren Steuerelement aus. Ein Programm zur statistischen Auswertung der Umsätze eines Unternehmens wird die Basisdaten aus einer Datenbank oder einer Datei auslesen und wahrscheinlich in grafischer

Form auf dem Drucker ausgeben. Die Frage ist dabei lediglich, wie Sie die Daten einlesen und wie Sie das Ergebnis ausgeben. Jede Programmiersprache bietet Ihnen dazu verschiedene Hilfsmittel, deren Handhabung zwar erlernt werden muss, die dann aber mit wenig Aufwand zu schnellen Ergebnissen führen.

3.1.2 Die CPU, Maschinensprache und Controller

Bei der Verarbeitung von Programmen spielt die CPU (Central Processing Unit, der Prozessor des Rechners) die wichtigste Rolle. Die CPU kennt einige hundert fest verdrahtete einfache Befehle, über die grundlegende mathematische Operationen wie Additionen, Subtraktionen, Multiplikationen, Divisionen und das Ansprechen der Speichermedien und der Ausgabemedien möglich sind. Die Summe dieser Befehle wird als *Maschinensprache* bezeichnet.

Maschinensprache

Jedes Programm führt auf der untersten Ebene meist mehrere Millionen Befehle in der CPU aus. Ein CPU-Befehl besteht aus einer Folge von Nullen und Einsen. Ein typischer Befehl sieht beispielsweise so aus: 00000000010000001000000001100100. Ein Teil der Folge steht für den Befehl selbst, ein weiterer Teil für variable Argumente des Befehls. Bei einer Multiplikation enthält die Befehlsfolge beispielsweise den Multiplikations-Befehl und die Adressen der zu multiplizierenden Speicherbereiche. Dass ein Maschinensprache-Befehl aus einzelnen Nullen und Einsen zusammengesetzt ist, liegt daran, dass die CPU im Prinzip aus sehr vielen kleinen elektronischen Schaltern besteht, die Strom entweder ein- oder ausschalten. Eine Null in einem Befehl repräsentiert einen ausgeschalteten, eine Eins einen eingeschalteten Schalter. Wenn die CPU einen Befehl ausführt, werden die für die Ausführung von Befehlen zuständigen Schalter entsprechend dem Befehl ein- bzw. ausgeschaltet, womit der Strom in spezielle Bahnen gelenkt wird. Was dabei wirklich in der CPU passiert, ist ziemlich komplex. Im Prinzip geht es hier nur darum, dass Sie wissen, warum Maschinensprache aus Nullen und Einsen besteht und warum Maschinensprache für den Menschen sehr schwer verständlich ist.

Assembler

In den Anfangstagen des Computers wurden Programme oft in Maschinensprache geschrieben, weil zu dieser Zeit noch keine höheren Sprachen zur Verfügung standen. Die Programmierung sah damals allerdings (als die Computer noch kaum leistungsfähig waren) so aus, dass entweder Schalter auf einfachen Schalttafeln umgelegt oder Lochkarten gestanzt wurden (die dann in den Computer eingelesen wurden). Sie können auch heute noch in dieser Sprache entwickeln. Allerdings wird dazu meist eine Abstraktion der Maschinensprache verwendet. Diese Abstraktion wird als *Assemblersprache* bezeichnet. Eine Assemblerspra-

che stellt die CPU-Befehle in einer für Menschen einfacher lesbaren Form dar, wie Sie noch auf Seite 142 sehen. Ein spezielles Programm, ein *Assembler*, übersetzt diese Befehle dann in Maschinensprache. Die Entwicklung eines Assemblerprogramms ist allerdings keine einfache Aufgabe und heutzutage, wo meist grafische Oberflächen für Programme verwendet werden und Programme sehr komplex sind, eigentlich auch unsinnig. Schließlich gibt es moderne höhere Programmiersprachen, die die Programmentwicklung erheblich vereinfachen, wie Sie ja bereits in Kapitel 2 gesehen haben.

Controller

Moderne Computer überlassen nicht nur der CPU die gesamte Arbeit. Verschiedene Controller übernehmen wichtige Teilaufgaben. Die Tastatur und eine PS/2-Maus werden beispielsweise vom Tastatur-Controller verarbeitet, ein spezieller I/O-Chip übernimmt das Handling der seriellen und parallelen Schnittstellen, der Diskettenlaufwerke und in einigen Fällen auch der Festplatten. Diese Controller sind meist Bestandteil des Motherboards[1]. Andere Controller sind Teil einer Erweiterungskarte. Moderne Festplatten, Grafik- und SCSI-Karten enthalten beispielsweise eigene Controller. Insgesamt wird die CPU damit von vielen Aufgaben entlastet und die Performance des Gesamtsystems erhöht.

Taktrate

Bei der Bewertung der Leistung einer CPU spielt neben der Anzahl und Mächtigkeit der Befehle (die bei neueren CPUs immer wieder erhöht wird) die Taktrate eine wichtige Rolle. Die Taktrate wird in Hertz gemessen. Hertz bedeutet „Takte pro Sekunde“. Eine CPU mit 2000 MHz ist also in der Lage 2000 Millionen Takte pro Sekunde auszuführen. Ein Takt ist im Prinzip das einmalige Ein- und Ausschalten des Stroms in der CPU. Ein CPU-(Maschinensprache-)Befehl benötigt eine festgelegte Anzahl an Takten. Ein einfacher Befehl wird beispielsweise in zwei Takten ausgeführt, komplexere Befehle benötigen mehr Takte. Daraus ergibt sich, dass eine CPU, die eine hohe Taktrate besitzt, mehr Befehle in einem gegebenen Zeitrahmen ausführen kann als eine CPU mit einer niedrigen Taktrate (also „schneller“ ist). Die Gesamt-Performance des Systems ergibt sich aber nicht nur aus der CPU-Taktrate, sondern auch aus der Anzahl der CPUs (mit speziellen Motherboards ist es auch möglich, mehrere CPUs gleichzeitig zu betreiben), der Geschwindigkeit des System-Bus (der für die Datenübertragung zwischen CPU, Arbeitsspeicher, Grafikkarte, Festplatte und anderen Medien verantwortlich ist), der Geschwindigkeit der Grafikkarte, der Größe des Arbeitsspeichers, der Geschwindigkeit der Festplatte und anderen Faktoren.

1. Ein Motherboard ist eine große Platine mit vielen Schaltkreisen, die definierte Aufgaben im Computer übernehmen, und mit standardisierten Steckplätzen für die CPU, den Arbeitsspeicher, die Grafikkarte etc. Alle Bestandteile des Computers sind mehr oder weniger direkt mit dem Motherboard verbunden.

3.1.3 Das BIOS und das Betriebssystem

Das BIOS

Die Befehle der CPU ermöglichen lediglich einfachste Operationen wie Additionen, Multiplikationen und das Kopieren von Daten zwischen Speicherbereichen. Programme, die lediglich CPU-Befehle nutzen könnten, wären sehr aufwändig. Wenn ein Programm direkt über CPU-Befehle beispielsweise eine Datei von der Festplatte lesen wollte, müsste dieses Programm den genauen Aufbau der Festplatte kennen um die richtigen Speicherbereiche der Festplatte ansprechen zu können. Problematisch dabei ist, dass jeder Computer unterschiedliche Hardware besitzen kann, die auf verschiedene Weise angesprochen werden muss. Da die CPU nur grundlegende Befehle beherrscht, müssten Programme jede Hardware-Variante berücksichtigen, wenn sie nur auf der CPU aufsetzen würden.

Mächtigere Befehle

Um dieses Problem zu lösen besitzt jeder (normale) Computer ein so genanntes BIOS (Basic In and Out System, das Basis-Ein-und-Ausgabe-System). Das BIOS ist ein Programm, das im ROM (Read Only Memory) des Computers auf dem Motherboard gespeichert ist. Ein ROM ist ein Speicher, der einmal dauerhaft beschrieben wurde und mit normalen Mitteln dann nur noch gelesen werden kann. Das BIOS enthält spezielle Befehle, die jeweils mehrere CPU-Befehle aufrufen und damit wesentlich mächtiger sind. Eine Besonderheit des BIOS ist, dass dieses die verschiedenen Hardware-Varianten berücksichtigt, also für den Zugriff auf verschiedene Varianten dieselben Befehle zur Verfügung stellt.

Hardware-Einstellungen

Aus diesem Grund verwaltet das BIOS auch Einstellungen zur Hardware des Systems. Diese Einstellungen können Sie erforschen und verändern, wenn Sie beim Booten des Systems eine bestimmte Tastenkombination betätigen (die kurz nach dem Start des Rechners auf dem Bildschirm angezeigt wird). Diese Einstellungen werden in einem speziellen überschreibbaren Speicher auf dem Motherboard gespeichert. Damit die so gespeicherten Konfigurationsdaten nicht verloren gehen, besitzt das Motherboard eine kleine Batterie, die ständig Strom für die Erhaltung des BIOS-Datenspeichers liefert.

Booten

Das BIOS ist daneben auch noch dafür zuständig, beim Booten des Rechners das Betriebssystem auf einem festgelegten Bereich der Festplatte, der CD bzw. eines anderen bootfähigen Speichermediums zu suchen, in den Arbeitsspeicher zu laden und auszuführen.

Das Betriebssystem und Hardware-Treiber

Das Betriebssystem, das vom BIOS beim Bootvorgang in den Arbeitsspeicher geladen wird, ermöglicht erst die Arbeit mit dem Computer. Es bietet dem Anwender eine Infrastruktur, über die Programme ausgeführt und Dateien verwaltet werden können.

Treiber

Für den Anwender unsichtbar verwaltet das Betriebssystem normalerweise verschiedene Treiber, die den Zugriff auf spezielle oder erweiterte Funktionen der Hardware ermöglichen. Diese Treiber sprechen die Hardware häufig direkt an, unter Umgehung der CPU. So ist es beispielsweise für Programme über das Betriebssystem möglich, die 3D-Funktionen einer Grafikkarte zu nutzen oder die grafischen Features eines Druckers anzusprechen. Der Zugriff über das Betriebssystem geschieht dabei immer auf die gleiche Weise, unabhängig von der Art der Hardware. Der Treiber, der normalerweise vom Hersteller der Hardware entwickelt wurde, übernimmt die physische Steuerung der Hardware, die bei verschiedener Hardware ganz unterschiedlich sein kann. Ein Drucker erwartet beispielsweise bestimmte Steuerungsbefehle zur Formatierung von Zeichen oder zum Wechsel der aktuellen Seite, verschiedene Drucker arbeiten mit vollkommen unterschiedlichen Befehlssätzen.

Für ein Windows-Programm ist es prinzipiell unwichtig, welcher Drucker angeschlossen ist, es nutzt lediglich die zum Zugriff auf den Drucker vorgesehenen Betriebssystemfunktionen. Die eigentliche Steuerung des Druckers übernimmt der Druckertreiber, der vom Betriebssystem mit den Aufgaben betreut wird, die das Programm ursprünglich an das Betriebssystem übergeben hat.

Betriebssystem-Funktionen

Das Betriebssystem enthält aber noch wesentlich mehr Funktionalität. Moderne Betriebssysteme bieten Programmen zusätzlich eine große Anzahl einfach anzuwendender Funktionen für allgemeine Aufgaben. Programme müssen nicht mehr auf die komplizierten BIOS- und CPU-Befehle zugreifen oder die Hardware (über den Treiber) direkt ansprechen, sondern können die wesentlich einfacher anzuwendenden Betriebssystem-Funktionen nutzen. Das Einlesen einer Datei erfordert unter Windows beispielsweise nur den Aufruf einer einzigen Funktion. Der Ausdruck von Text- oder Grafikdaten ist ebenfalls für Programme recht einfach über die dafür zuständigen Betriebssystemfunktionen. Grafische Betriebssysteme bieten Programmen zudem spezielle Fenster und Steuerelemente, die genutzt werden, um die Oberfläche einer Anwendung zu erzeugen. Daneben stehen einem Programm auch Funktionen zum Zeichnen von grafischen Elementen wie einfachen Rechtecken, Kreisen, aber auch zur Ausgabe von Bildern auf dem Bildschirm zur Verfügung.

Kapselung des Hardware-Zugriffs

Viele Betriebssysteme wie Windows und Linux kapseln den Zugriff auf die Hardware komplett. Ein Programm muss nicht mehr direkt auf die CPU, das BIOS oder auf die Hardware zugreifen, sondern kann dazu Betriebssystemfunktionen nutzen. Bei „sicheren" Betriebssystemen wie Windows 2000 und Windows XP besteht sogar gar keine Möglichkeit, direkt auf die Hardware zuzugreifen. Weil dieser Zugriff nur über Betriebssystemfunktionen möglich ist, kann das Betriebssystem sicherstellen, dass (möglichst) keine Hardware-Funktionen aufgerufen werden, die Probleme verursachen könnten (wie z. B. das versehentliche Überschreiben eines Festplattenbereichs).

Die Zusammenarbeit

Abbildung 3.1 zeigt die Zusammenarbeit zwischen Anwendungen, dem Betriebssystem, dem BIOS und der Computer-Hardware (wozu auch die CPU gehört).

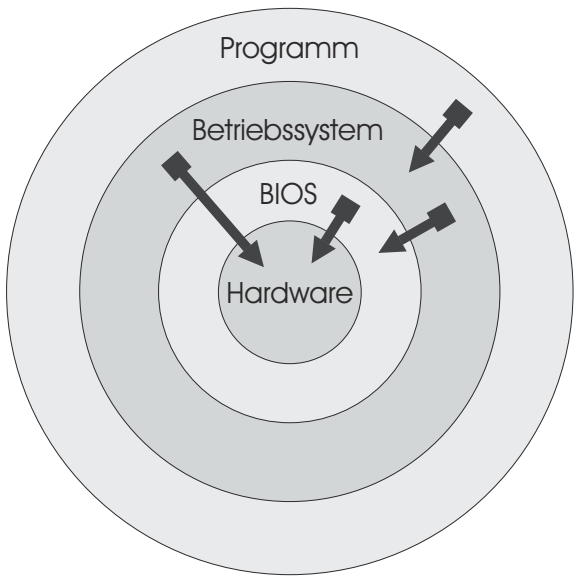

Abbildung 3.1: Die Zusammenarbeit von Programmen, dem Betriebssystem und dem BIOS

Einfache Betriebssysteme wie z. B. DOS liefern ihren Anwendungen natürlich wesentlich weniger Funktionen als komplexe Betriebssysteme wie Windows oder Linux. Wenn Sie schon mal mit einem einfachen DOS-Programm gearbeitet haben, werden Sie wissen, dass Ein- und Ausgaben unter DOS normalerweise direkt an der Konsole erfolgen. Komplexere DOS-Programme, die dem Anwender Windows-ähnliche Fenster anbieten, müssen diese Funktionalität selbst implementieren,

können dabei also nicht auf das Betriebssystem zurückgreifen. Windows und das X Window-System von Linux bieten Programmen dagegen eine Vielzahl von Funktionen für Ein- und Ausgaben. Jedes Fenster einer grafischen Anwendung und alle darin enthaltenen Elemente werden mit Hilfe von Betriebssystemfunktionen auf dem Bildschirm ausgegeben[2], jeder Ausdruck auf dem Drucker erfolgt über Betriebssystemfunktionen usw. Den Ablauf der Erzeugung einer Programmoberfläche (in Form eines Fensters) illustriert Abbildung 3.2. Abbildung 3.3 zeigt, wie ein Programm prinzipiell Daten auf dem Drucker ausgibt.

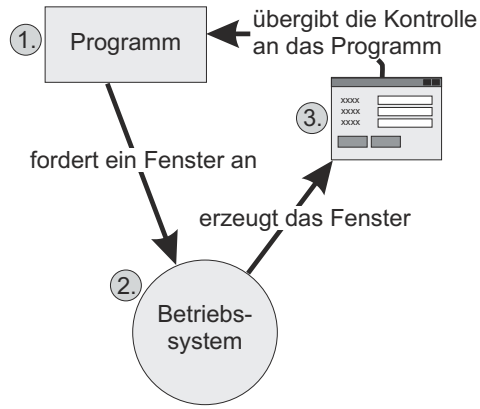

Abbildung 3.2: Ein Programm verwendet Betriebssystemfunktionen zum Erzeugen von Fenstern.

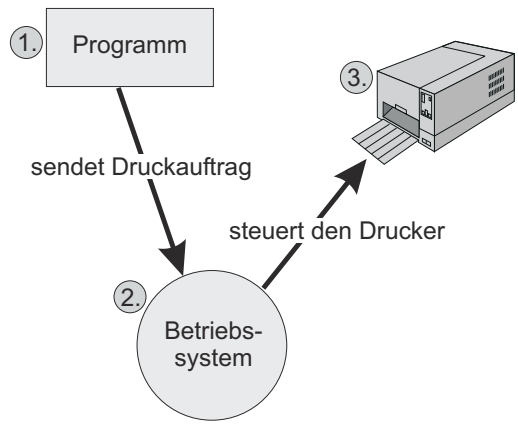

Abbildung 3.3: Ein Programm verwendet Betriebssystemfunktionen zum Drucken von Daten.

2. Was bei Linux nicht so ganz korrekt ist, denn hier stellt nicht das Betriebssystem, sondern ein spezielles Programm (meist das *X Window-System*) die grundlegende Funktionalität für grafische Oberflächen zur Verfügung. Auf diesem Programm bauen Fenstermanager wie die KDE und GNOME auf, die einen Desktop zur Verfügung stellen und Fenster ganz individuell darstellen.

3.1.4 Was ist ein Programm?

Ein Programm ist eine gespeicherte Folge von Anweisungen. Diese Anweisungen können Aktionen bewirken (wie Eingaben entgegennehmen oder Dateien einlesen), Berechnungen ausführen, Daten ausgeben, aber auch dafür sorgen, dass ein Teil des Programms wiederholt oder abhängig von einer Bedingung der eine oder andere Teil des Programms ausgeführt wird. Beim Einlesen einer Textdatei werden beispielsweise häufig die einzelnen Zeilen in einer Schleife eingelesen, verarbeitet und ausgegeben, bis das Ende der Datei erreicht ist. Diese bei der Programmierung sehr wichtige Strukturierung eines Programms behandelt das Buch ausführlich in Kapitel 5.

Ein einfaches Programm zur Berechnung einer mathematischen Formel enthält z. B. Anweisungen, die die Eingaben des Anwenders entgegennehmen, Anweisungen, die ein Ergebnis berechnen, und Anweisungen, die das Ergebnis auf dem Bildschirm oder dem Drucker ausgeben. Solch ein Programm haben Sie in Kapitel 2 schon selbst geschrieben.

Da Programme immer von Betriebssystemen ausgeführt werden, kann ein Programm aus Betriebssystem-Funktionen und Struktur-Anweisungen (Schleifen, Verzweigungen) bestehen. Das Betriebssystem sorgt bei der Ausführung des Programms dafür, dass die Betriebssystem-Funktionen in passende CPU- oder BIOS-Befehle und Treiber-Funktionsaufrufe umgesetzt werden. Programme können, wenn es das Betriebssystem zulässt, aber auch direkte CPU- oder BIOS-Befehle enthalten. Einfache Programme können sogar nur aus CPU- oder BIOS-Befehlen bestehen.

3.1.5 Die Rolle des Arbeitsspeichers

Programme bestehen also aus einzelnen Anweisungen. Diese Anweisungen sind zusammenhängend in Form einer Datei auf einem Speichermedium wie z. B. der Festplatte gespeichert. Eine Datei ist eine zusammengehörige Folge von Daten, die vom Betriebssystem als Ganzes eingelesen und verwendet werden kann. Im Falle eines Programms besteht eine Datei aus mehreren einzelnen Anweisungen. Solche Dateien besitzen unter Windows die Endung *.exe*, was für „executable“, also „ausführbar“ steht. Linux verwendet keine Dateiendungen für ausführbare Dateien, sondern erkennt den Typ der Daten an Informationen, die versteckt in der Datei gespeichert werden.

Schneller Speicher für Programme

Wenn Sie eine Programmdatei ausführen (in Windows z. B. über einen Doppelklick auf diese Datei oder über die Auswahl der Datei über das Startmenü), wird die Datei zunächst als Ganzes von der Festplatte in den Arbeitsspeicher gelesen. Windows erkennt an der Dateiendung,

dass es sich um eine ausführbare Datei handelt, Linux erkennt dies an der Datei selbst. Das Betriebssystem interpretiert die gespeicherten Daten also als Programm-Anweisungen, liest diese einzeln aus dem Arbeitsspeicher und führt Anweisung für Anweisung aus.

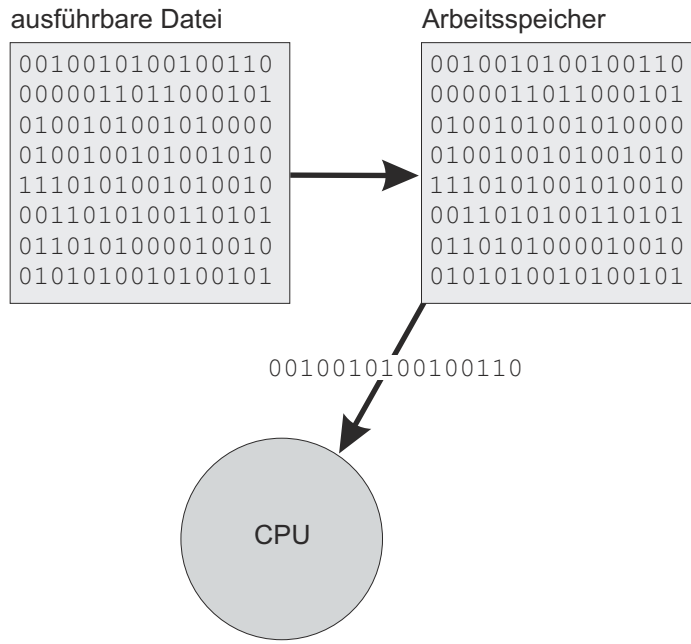

Abbildung 3.4: Eine ausführbare Datei wird in den Arbeitsspeicher geladen und dann Anweisung für Anweisung von der CPU ausgeführt.

Dass die Datei dazu in den Arbeitsspeicher des Computers geladen wird, hat einen Grund: Fast alle Programme werden nicht einfach nur von oben nach unten abgearbeitet, sondern enthalten an vielen Stellen Rücksprünge zu vorhergehenden oder Sprünge zu später folgenden Anweisungen. Damit erreicht man beim Programmieren, dass bestimmte Programmteile mehrfach wiederholt und andere bedingungsabhängig ausgeführt werden können. Die dazu verwendeten Techniken werden in Kapitel 5 behandelt. Würde die CPU die einzelnen Anweisungen immer wieder von der im Vergleich zum Arbeitsspeicher sehr langsamen Festplatte lesen müssen, würden Programme nur sehr schleppend ausgeführt werden. Um die Ausführung zu beschleunigen, wird das gesamte Programm vor der Ausführung also in den Arbeitsspeicher geladen.

Speichern von Daten

Der Arbeitsspeicher wird aber nicht nur zur Speicherung ausführbarer Programme verwendet. Viele Schritte zur Verarbeitung eingegebener Daten sind so komplex, dass ein Programm Zwischenergebnisse berechnen muss, die in späteren Anweisungen weiterverwendet werden. Diese