Harald Hillgärtner

Protected Mode

Zur Lesbarkeit des Computers

Sehr geehrte Damen und Herren, liebe Kolleginnen und Kollegen, der Titel des heutigen Vortrages „Protected Mode" muss zwar reißerisch erscheinen, ist jedoch recht eindeutig: Bezeichnet wird mit Protected Mode ein Konzept in der Architektur von Prozessoren, also der – ganz im wörtlichen Sinne – „Schaltzentrale" heutiger Computer. Der Protected Mode erlaubt es, den einzelnen laufenden Programmen einen Speicherbereich zuzuweisen, ohne dass diese Prozesse in den Speicherbereich anderer Prozesse schreiben können. Bekanntermaßen liegt ja die Software einträchtig neben den Daten auf der nichtflüchtigen, aber vergleichsweise trägen Festplatte. Soll ein Programm ausgeführt werden, dann wird es zuvor in den Hauptspeicher geladen, auf welchen ein wesentlich schnellerer Zugriff von Seiten des Prozessors erfolgen kann. Über den Zugriff auf diesen Hauptspeicher zu wachen ist nun unter anderem Aufgabe des Prozessors im Protected Mode. Wichtig ist dies vor allen Dingen beim so genannten „Multi-Tasking", bei dem ja prinzipiell die parallele Nutzung ein und desselben Computers durch mehrere Nutzerinnen möglich wird, ohne dass man voneinander überhaupt etwas bemerkt. Die drastischste Form eines solchen Bemerkens wäre, wenn ein Nutzer ein Programm startet, das – ungenügsam mit dem zur Verfügung stehenden Speicher – sich in den Speicherbereich eines anderen Prozesses eines anderen Benutzers ausbreitet. Die Folge eines solchen Ausbreitens ist regelmäßig der Absturz des Programms. Die meisten von Ihnen werden diesen Effekt noch aus der Zeit von Windows3.11 bis zu WindowsME kennen, wenn der Absturz eines Programms auch ohne weiteres den Absturz des ganzen Computers verursachte, einfach aus dem Grund, da bis Windows2000 das Betriebssystem nicht in der Lage war, den von Seiten des Prozessors zur Verfügung gestellten Protected Mode effektiv auszunutzen. Geschuldet war dies wiederum der Kompatibilität zum, wie es Friedrich Kittler charmant formuliert, „dümmsten aller Betriebssysteme"[1], nämlich MS-DOS. So

1 Friedrich Kittler: Es gibt keine Software, in: Hans Ulrich Gumbrecht, K. Ludwig Pfeiffer (Hg.):

hatte man unter allen zu DOS kompatiblen Windows-Versionen zwar meist die Ehre, mit der Warnmeldung über eine „Allgemeine Schutzverletzung" darauf hingewiesen zu werden, dass just eine solche Übertretung des Speicherbereiches eines Prozesses eingetreten war, doch in aller Regel war es dann schon zu spät. Ohne eine effektive Unterscheidung zwischen Anwenderprozessen und Betriebssystemprozessen war Letzteren auch nicht mehr zu helfen. Für die auf die Schutzverletzung folgende Meldung von Windows hat sich die sprechende Bezeichnung „Blue Screen of Death" eingebürgert. Moderne Betriebssysteme nun unterstützten ausnahmslos den Protected Mode und werden eigentlich erst damit fähig, mehrere laufende Programme gleichzeitig zu verwalten, etwas, an das sich wohl inzwischen jeder gewöhnt haben wird.

"Lesen, was nie geschrieben wurde"

Was im Lichte dieser kurzen Erörterung als Verbesserung erscheinen muss, und vom technischen Standpunkt her auch ist, veranlasste Friedrich Kittler zu Beginn der 90er Jahre zum Verfassen eines Aufsatzes unter dem klingenden Titel *Protected Mode*. Allerdings wurde hier in der Ersetzung des „Real Mode", wie er vorher in den Prozessoren herrschte, durch den Protected Mode eine weitgehende Entmündigung des Computerbenutzers vermutet. Unwiderruflich vorbei war es nun „mit der guten alten Zeit, als Mikroprozessorpins noch groß genug für schlichte Lötkolben waren" und „auch Literaturwissenschaftler mit dem Intel-Prozessor 8086 anstellen" konnten, „was sie wollten."[2] Schwerer noch, als dass die Prozessoren zu klein wurden, um ihnen mit dem Lötkolben zu Leibe zu rücken, muss wiegen, dass selbige Literaturwissenschaftler durch den Protected Mode nun auch noch außerstande gesetzt werden, die Befehle auszuführen, die sie ausführen möchten. Denn der Protected Mode verwaltet den Speicher, indem er über die Privilegien der Prozesse wacht, damit diese sich nicht doch hinterrücks das Recht einräumen, in die Speicherbereiche anderer Prozesse einzugreifen. Dieser interne Satz von Privilegien und Zugriffsbeschränkungen veranlasste Kittler dann auch, im Protected Mode nichts anderes als die „ins

---

Schrift, München 1993, S. 367-378, hier S. 371.

2 Friedrich Kittler: Protected Mode, in: Norbert Bolz, Friedrich Kittler, Georg-Christoph Tholen (Hg.): Computer als Medium, München 1994, S. 209-220, hier S. 209.

Silizium versenkte US-Bürokratie"[3] zu sehen. Über diese Segregation auf Seiten der Hardware, von der der Nutzer schließlich nichts bemerken soll, aber wird dem gleichen Nutzer, wie Kittler befürchtet, die ganze Maschine entzogen. Verdammt ist er, diesseits der Benutzeroberfläche sein Dasein zu fristen und sich damit abzufinden, dass er ein „Untertan der Microsoft Corporation"[4] geworden ist.

Jedoch, Kittler wäre nicht Kittler, würde er nicht gerade hieraus einen Imperativ ableiten. Die technische Eleganz, die es dem lötkolbenbewehrten Literaturwissenschaftler nachgerade verunmöglichen soll, Befehle abzusetzen ohne über die entsprechenden Privilegien zu verfügen, wird nämlich unterminiert durch die Kompatibilität zu älterer Software. Dass Windows bis zur Einführung von Windows2000 zwar die Fähigkeit zum Multitasking versprach, dieses Versprechen aber nicht einlösen konnte, lag an genau dieser Kompatibilität zu älterer Software. So gibt es Befehle, die zwar nicht mehr dokumentiert werden, die jedoch von einer Systemgeneration zur nächsten mitgeschleift werden und damit ihre Wirkmächtigkeit behalten. Nebeneffekt dieser Praxis ist, dass die Codes, wie Kittler sagt, auf „dieselbe Opazität wie Alltagssprachen hinauslaufen"[5]. Hieraus leitet er dann ab, dass eine der Diskursanalyse verwandte Praxis, er nennt sie gegen Ende des Textes „Kryptoanalyse", angeraten sei, um, Kittler zitiert hier Hugo von Hofmannsthal, „lesen zu können, was nie geschrieben wurde".[6] Der Imperativ wäre, eben genau dies zu versuchen, etwas Verborgenes zu lesen um mit diesem Wissen wider den „Protected Mode" zu handeln.

Zur Lesbarkeit des Computers

Damit aber wären wir beim eigentlichen Thema des Vortrags, denn versteht man „Lesen" zunächst ganz trivial als Lesen von Text, wie ist es dann um die Lesbarkeit des Computers bestellt? Neben dem Protected Mode, der auf Seiten der Hardware Privilegien verwaltet, ist die Computerwelt aufgespalten in Benutzer und Administratorinnen. Gewissermaßen spiegelt sich hier der Protected Mode im Sinne von „privilegiert" und „nicht-privilegiert" wider. Werden dort

3 Ebd., S. 214.
4 Ebd., S. 209.
5 Ebd., S. 218.
6 Ebd., S. 219.

Speicherbereiche verwaltet, so werden hier Rechte an Dateien und Programmen festgelegt, wobei fein abgestimmt werden kann, wer eine Datei lesen, wer in diese Datei schreiben, oder wer eine Datei gar ausführen darf. Verfügt man aber als Nicht-Administrator nicht über ausreichende Privilegien um ein Programm zu starten, dann ist sogar der magischen Allmacht des Wortes am Computer ein Ende gesetzt, wie sie Kittler recht pointiert beschreibt: „Das Kürzel WP nämlich tut, was es sagt. Im Unterschied nicht nur zum Wort WordPerfect, sondern auch zu leeren alteuropäischen Wörtern wie Geist oder Wort umfassen ausführbare Dateien alle Routinen und Daten, die zu ihrer Realisierung notwendig sind. [...] Solche Triumphe gewährt Software."[7] Doch die Unterscheidung in Hinsicht auf Privilegien geht noch weiter, oder könnte noch weiter getrieben werden: Machtlos sind selbst Administratoren, oder „Superuser", wie es auch manchmal unter Unix heißt, in Angesicht der so genannten „Binarys", dem maschinenlesbaren Code, zu dem nur die Programmiererinnen den Schlüssel haben.

Nun weiß ich nicht, wie vertraut Sie mit dem Konzept von Programmiersprachen sind. Glücklicherweise lässt sich das für diesen Vortrag Wichtigste in einigen wenigen Sätzen zusammenfassen. Die erste grobe Unterscheidung muss zwischen der ganz basalen Maschinensprache, die im Grunde für Menschen unlesbar ist, und den höheren Programmiersprachen getroffen werden. Maschinensprache kann in Hexadezimalwerten notiert werden. Das Hexadezimalsystem beinhaltet die Ziffern 0-9 des Dezimalsystems und zusätzlich die Buchstaben a-f des lateinischen Alphabets, so dass es im Hexadezimalsystem 16 verschiedene Ziffern gibt. Für die Maschinensprache gibt es zudem noch ein lesbares Pendant, die Assemblersprache. Diese ist im Grunde eine Eins-zu-eins-Übersetzung von Befehlen in Maschinensprache, die der Prozessor direkt verarbeiten kann. Befehle, die mittels Hexadezimalwerten oder in Assembler erteilt werden können, sind sehr einfach strukturiert und klingen etwa folgendermaßen: Lade eine Zahl von Speicherstelle x in Register y, zähle den Inhalt des Register y um eins hoch, addiere den Inhalt von Register y zum Inhalt des Registers z, usw. usf. Dass nun ein Prozessor bestimmte Befehle direkt verarbei-

7 Kittler: Es gibt keine Software, a.a.O., S. 370. Ebenso pointiert ist auch folgende Formulierung: ,,Die ENTER-Taste hat eine Macht erlangt, die den Wortsinn von Poesie, nämlich machen, im Unterschied zu aller Poesie oder Literatur der Geschichte erstmals einlöst." (Friedrich Kittler: Daten - Zahlen - Codes, Leipzig 1998, S. 21.)

ten kann, bedeutet nichts anderes, als dass diese Befehle prozessorspezifisch sind und von diesem intern gewissermaßen als eine endlose Litanei von Nullen und Einsen interpretiert werden. Nullen und Einsen, und das ist sicherlich für niemanden ein Geheimnis, sind letztendlich die Basis, auf der die Prozessoren rechnen, oder besser gesagt: komputieren.

Assemblersprachen werden in aller Regel nur noch in maschinennahen Bereichen eingesetzt, bei denen es darauf ankommt, möglichst speicher- und geschwindigkeitseffizient zu arbeiten. Mit Assemblerbefehlen zu programmieren ist, wie sie sich denken können, ähnlich mühsam wie die direkte Notation von Hexadezimalwerten. Höhere Programmiersprachen hingegen bieten der Programmiererin ein je nach Sprache sehr ausgefeiltes Set an Befehlen und Prozeduren, die immer wiederkehrende Aufgaben zu komplexeren Befehlen zusammenfassen. Hierauf will ich aber an dieser Stelle schon gar nicht mehr eingehen. Jedoch wäre noch eine weitere Unterscheidung zu treffen: Es gibt nämlich eine ganze Reihe von sogenannten Skript-Sprachen wie Perl oder Python, die erst, wenn sie gestartet werden, von mitgelieferten Interpretern in Maschinensprache übersetzt und dem Prozessor übergeben werden. Konkret bedeutet dies, dass eine in einer solchen Skript-Sprache verfasste Software jederzeit von einer Programmiererin, die entsprechenden Sprachkenntnisse vorausgesetzt, umgeschrieben, erweitert, vereinfacht oder einfach nur gelesen und verstanden werden kann. Ganz anders verhält es sich bei den Sprachen wie C oder dessen Erweiterung C++, in denen gegenwärtig die meiste Software geschrieben wird. Hier wird eine Schrift verfasst, der „Quellcode", der nach Fertigstellen des Programms von einer anderen Software, dem Compiler, in Maschinencode übersetzt wird und bei kommerzieller Software nur in dieser, für Menschen unlesbaren Form, distribuiert wird. Diese „Binarys" können eben nicht gelesen, nicht erweitert, nicht verbessert, nicht verstanden werden. Zwar gibt es die Möglichkeit des „reverse-engineering", jedoch ist dies zum einen in den Lizenzen der Software so gut wie immer ausdrücklich untersagt und zum anderen sehr fehlerbehaftet. In anderen Worten: mit Binarys kann man schwerlich etwas anderes anfangen, als sie auszuführen. Zwar liebt es Friedrich Kittler darauf hinzuweisen, dass der Erfinder der Universellen Maschine, Alan Turing, es vorgezo-

gen haben soll, Programme in Form von Nullen und Einsen zu lesen[8], jedoch ist ein solches Unterfangen in der Gegenwart einigermaßen umständlich, wenn nicht gar illusionär. Jedenfalls ist es so für die Hersteller erfolgreicher Software durchaus möglich, ihre Software zu vertreiben, ohne dass befürchtet werden muss, dass jemand „abschreibt".

Geheimniskrämerei

Da nun der Quellcode so bekannter Software wie MS-Windows zu den bestgehütetsten Geheimnissen der Welt gehören, und selbstverständlich ähnliches für das Gros der kommerziellen Software gilt, also der Software, mit der über 95% aller Anwender in der Welt arbeiten, gibt es im Grunde im Computer nichts zu lesen. Erst recht gilt dies, da auf Skript-Sprachen basierende Software eine mehr als geringe Rolle spielen. Dies allerdings nicht aus Gründen der Geheimniskrämerei, sondern weil solche Programme, bevor sie den Prozessor erreichen, zunächst einmal in Maschinencode übersetzt werden müssen, was schlicht und ergreifend mehr Zeit braucht als das direkte Ausführen von Binarys. Hiermit wäre aber die Frage nach der Lesbarkeit des Computers bereits beantwortet, denn Binarys sind nicht zum Lesen bestimmt. Zwar bleibt auf der Ebene der Administration noch einiges an Dokumentation oder in Konfigurationsdateien zu lesen, jedoch wäre hier zu fragen, wie interessant dies ist, wenn es hierbei im Wesentlichen nur noch darum geht, etwas zu verwalten, das hinter diesen Verwaltungsvorschriften gänzlich opak ist. Diese Opazität auszunutzen, um – mit Kittler gesprochen – zu lesen, was nirgends geschrieben steht, ist sicherlich das Geschäft der so genannten Hacker und Virenschreiberinnen. Hiermit wären wir aber wieder bei Kittlers Imperativ, der sich nicht zuletzt aus seinem Steckenpferd, alle Medientechnologie sei letztendlich Kriegstechnologie, ableitet: Die Betriebssysteme, die Software muss überlistet werden, um mit ihr machen zu können, was man machen möchte. Ähnlich einem Hacker muss man versuchen, das Geheimnis der Software zu ergründen, denn es liegt ja gerade nicht zutage. Dies heißt aber auch nichts anderes, als abseits der von der Software vorgegebenen Pfade kreativ zu werden.

8 Vgl.: Friedrich Kittler: Protected Mode, a.a.O., S. 210.

Nun ist es ja gerade das Versprechen der konsequent auf grafische Benutzerführung ausgerichteten Betriebssysteme, dem Nutzer eben aufgrund des Verbergens aller Kommandozeilen den größtmöglichen Freiraum zur Kreativität zu ermöglichen. Wenn man sich keine Befehle merken muss, sondern stattdessen aus Menüs, Submenüs und Subsubmenüs heraus quasi intuitiv die gewünschte Funktion herausfischt, dann wäre die Software für den Nutzer transparent. Der andere, noch konsequentere Weg, mittels Icons, wie einem Pinsel oder einem Radiergummi, bei jeder gängigen Bildbearbeitungssoftware Werkzeuge auszuwählen, die wie die Befehle Kittlers zwar nicht tun, was sie sagen, sondern vielmehr tun, was sie zeigen, beinhaltet erst recht das Versprechen, alle Konzentration des Nutzers für den kreativen Arbeitsprozess freizusetzen. Was hiermit aber erzeugt wird, sind Idioten in der ursprünglichen Bedeutung des Wortes, nämlich unkundige Laien, im Falle der grafischen Benutzeroberfläche schreib- und leseunkundige Idioten.[9] Erfüllt wäre hiermit, polemisch formuliert, Vilém Flussers Utopie vom Ende der Schrift und der Heraufkunft eines neuen Zeitalters technischer Bilder. Diese Utopie beschreibt insofern tatsächlich eine erstrebenswerte Zukunft, da darin das Versprechen vollkommener Freiheit des Menschen liegt, sich ausschließlich schöpferisch zu betätigen. Flusser selbst nennt diese Zukunft einen fortwährenden „Zerebralorgasmus".[10] Der Imperativ Flussers ist in dieser Hinsicht ein ganz anderer als der Kittlers, denn nicht hinter die Oberfläche wäre zu schauen, sondern es gilt, auf der Oberfläche zu bleiben. Im Zeitalter nach der Schrift gibt es schlicht nichts mehr zu lesen, das Illiteratentum ist nicht regressiv, sondern progressiv.

Computer als Textmaschine

In einer entscheidenden Hinsicht jedoch muss Flussers Schwanengesang, sein Abgesang auf die Schrift, für die Gegenwart entschieden zurückgewiesen werden. Im Gegensatz dazu, wie sich die auf grafische Benutzerführung ausgelegten Betriebssysteme darstellen, indem sie versuchen auf Kommandozeilen zu verzichten, obwohl sie doch über eine verfügen, ist der Computer eine Textmaschine. Zwar ist es meines Erachtens nicht so, wie etwa Florian Cramer be-

9 Vgl.: Stefan Heidenreich: Icons: Bilder für User und Idioten, in: Birgit Richard, Robert Klanten, Stefan Heidenreich (Hg.): Icons, Berlin 1998, S. 82-86.

hauptet, dass im Computer nichts anderes als Text vorfindlich sei[11], vielmehr handelt es sich, wie Kittler betont, um „eine Schrift zugleich vor jeder Schrift und nach jeder Schrift"[12]. Die Interna des Computers sind Schrift vor der Schrift, da sie die Komplexität jedes alltagssprachlichen Schriftsystems bei weitem unterbieten. Unterbieten bedeutet, dass Computer nach dem fundamentalen Prinzip der Turing-Maschine funktionieren, eine Turing-Maschine aber nichts anderes vollbringen kann, als auf einem Endlospapierband nachzuschauen, ob ein Feld leer ist oder eben nicht, wobei sie je nach leer oder nicht-leer ihren Zustand ändert und ein Feld markiert oder eben diese Markierung löscht. In diesem Sinne handelt es sich beim Prozessieren um eine endlose Litanei von Schreib- und Leseakten. Zwar klingt dies so, als würde es sich beim Gelesenen und beim Geschriebenen um Text handeln, aber wenn überhaupt, dann handelt es sich um einen Text von Daten. Denn, so das Argument, muss sich jeder Code, den man als Text bezeichnen könnte, vor seiner Verarbeitung im Inneren des Prozessors in Daten verwandeln, gleichsam in eine Markierung auf dem Papierband. Dies ist die Voraussetzung dafür, dass die Von-Neumann-Architektur heutiger Rechner einen gemeinsamen Speicher für Daten und Instruktionen vorsieht und es so theoretisch möglich wird, dass sich ein Programm selbst modifizieren, sich selbst schreiben kann. Gewissermaßen wird nicht aus den Daten Text, wie Florian Cramer der Überzeugung ist, sondern viel eher werden aus Programmtexten Daten. Sowohl aus digitalisierten Fotografien wie aus Programmcode werden Transistorzustände. Aus diesem Grunde ist Schrift im Computer wiederum eine Schrift nach jeder Schrift: Die Turing-Maschine wäre, mit Kittler gesprochen, die Vollendung der Schriftgeschichte, indem der Computer das Digitale der Schrift zum fundamentalen Prinzip erhebt. Das Zeitalter der Digitalisierung begann mit der Erfindung des Alphabets, dem „Prototyp alles Diskreten"[13], und endet mit der Turing-Maschine.

---

10 Vgl.: Vilém Flusser: Ins Universum der technischen Bilder, Göttingen 1999, S. 143.

11 Vgl.: Florian Cramer: Für eine Textwissenschaft des Digitalen, 1.10.2001, http://userpage.fu-berlin.de/~cantsin/homepage/writings/theory/germanistentag_2001/textwissenschaft_des_digitalen.html (zuletzt besucht am 4.10.2004)

12 Kittler: Daten - Zahlen - Codes, a.a.O., S. 21.

13 Friedrich Kittler: Code oder wie sich etwas anderes Schreiben lässt, Ars Electronica Katalog-Archiv 2003, http://www.aec.at/de/archiv-files/20031/FE_2003_kittler_de.pdf, S. 18 (zuletzt

Trotzdem ist der Computer eine Textmaschine, denn die nach wie vor privilegierte Schnittstelle zum Computer ist Schrift. Dafür gibt es einen schlichten Grund: Mit ihr lassen sich grammatikalisch-komplexe und eindeutige Konstrukte bilden. Befehle können miteinander verknüpft werden, es kann auf das Eintreten oder auf das Nicht-Eintreten eines Falles reagiert werden, es können Ersetzungen vorgenommen werden, Schleifen an Bedingungen geknüpft werden etc., kurz: es kann gescriptet werden. Die Polyvalenz von Bildern und eben auch von Icons, die ja bekanntlich auf den grafischen Benutzeroberflächen oftmals zusätzlich beschriftet werden, lassen solche Operationen nur schwerlich zu. Hiermit soll nicht gesagt werden, dass dies auf ewig so bleiben muss, dass Schrift sozusagen die Essenz des Computers sei, jedoch bietet sie bei der Arbeit mit Computern entscheidende Vorteile.

Es soll nun nicht um eine Art Paragone gehen, nicht soll das Konzept grafischer Benutzerumgebungen zugunsten einer „protestantischen", also kargen schriftbasierten Kommandozeile depraviert werden. Aber es muss möglich sein, stupide, weil immer wiederkehrende Arbeit mit einem kurzen Skript zu delegieren, denn schließlich ist es eines der Versprechen des Computers, Arbeit zu automatisieren. Es sollte möglich sein, einen einfachen Text zu schreiben, mit dessen Hilfe man beispielsweise sein E-Mail-Konto abfragt, alle Beiträge etwa zu einem Publikationsprojekt herausfiltert, diese Dateianhänge dann gleich an eine Textverarbeitung weiterleitet, um bereits vorher definierte Formatvorlagen darauf anzuwenden, alte gegen neue deutsche Rechtschreibung auszutauschen, die unterschiedlichen Dateiformate anzupassen, diese Dateien mit einem Packprogramm zu Archiven zusammenzufassen, um die daraus resultierende Datei dann wieder mit dem Mail-Programm an die anderen Herausgeber der Publikation weiterzusenden. Hierfür wäre es aber notwendig, dass es zum einen eine voll funktionstüchtige Kommandozeile gibt, dass sich weiterhin die benötigten Programme mit den entsprechenden Parametern aus der Kommandozeile heraus bedienen lassen und dass dies alles auch noch dokumentiert wird.[14]

---

besucht am 8.10.2004).

14 Freilich braucht es keine Kommandozeile, denn einzelne Programme lassen sich auch über standardisierte Schnittstellen etwa zu einer ,,Office-Suite" zusammenfassen. Ebenso lassen sich ,,Macro-Rekorder" einbauen, die die einzelnen Schritte der Nutzer aufzeichnen und als Skript zur Verfügung stellen. Im Grunde handelt es sich hierbei lediglich um ein Verstecken der

Nun bin ich allerdings unversehens von der Frage nach der Lesbarkeit des Computers bei der Zumutung angelangt, dass der Nutzer bitte doch selbst beginnen möge, Programme zu schreiben. Nichts anderes sind nämlich Skripte, nur dass dies Programme sind, die sich anderer Programme bedienen, um zu tun, was sie tun. Im Grunde sind die Nutzer aber dieser Zumutung ohnehin immer schon ausgesetzt, bloß, dass sie programmiert werden, anstatt selbst zu programmieren. Als Nutzer muss man sich daran gewöhnen, wie die jeweilige Software funktioniert, um dann für die jeweiligen Aktionen immer ganz stupide die richtigen Schritte auszuführen. Mit einem Skript ließe sich der Spieß wieder umdrehen, man programmiert die Programme, damit sie gleich das tun, von dem man schon vorher weiß, was sie tun sollen. Um Kittler zu variieren: Solche Triumphe gewähren Skripte.

Um zu solchen Triumphen allerdings erst zu gelangen, wäre ein Literarisierung vonnöten, die überflüssig zu machen die modernen grafischen Benutzerumgebungen ja überhaupt erst angetreten sind. Um hiermit wieder zum eigentlichen Thema des Vortrags zurückzukehren: Gängige Betriebsysteme wie Microsoft Windows behaupten, dass es im Computer nichts zu lesen gibt. Dies geht noch darüber hinaus, dass der Quellcode zum Firmengeheimnis deklariert wird, denn auch die Bedienung der Programme soll nicht literat, mittels Befehlen, sondern intuitiv erfolgen. Statt einer „Geheimwissenschaft", bei der die Schriftkundigen eine Reihe unverständlicher Befehle als Kolonne vor einem schwarzen Bildschirmhintergrund eingeben, einer „schwarzen Kunst" gleich, ist es das Versprechen der Oberfläche, transparent zu sein. Allerdings konterkariert sich dieses Streben nach Transparenz gleich schon wieder selbst, indem etwa solche Softwareprojekte wie Microsoft Word versuchen, alle nur erdenklichen Funktionsweisen in seine Oberfläche zu integrieren. Sinn und Zweck scheint zu sein, dass man nicht anderer Software bedarf, deren eventuell andere Funktionsweise, oder gar bloß deren Namen sich zu merken eine Zumutung für den Nutzer sein könnte. Dies ließe sich ebenso lakonisch wie treffend kommentieren: „Seit seinen frühen Versionen ist Word wie eine ertrunkene und im Wasser treibende Kuh angeschwollen."[15] Dies ist die Alternative zur „schwarzen Kunst" des

---

Kommandozeile.

15 Matthew Fuller: Anscheinend wollen Sie einen Brief schreiben, Telepolis, 7.3.2001, http://www.telepolis.de/deutsch/special/med/7072/1.html (zuletzt besucht am 4.10. 2004).

Schreibens von Befehlen, dass man sich aufhält in einer Oberfläche, die es ebenso zu durchforsten gilt, wie die Befehlsreferenzen der Kommandos, die man schreibt, statt sie zu „klicken".

Unix as literature

Ich weiß nicht, wie geläufig der Begriff der „Freien Software" ist. Gemeint ist hiermit ein Lizenzmodell, bei dem Software zwar nicht notwendig kostenfrei angeboten wird, jedoch in der Lizenz ausdrücklich das Recht vorgesehen ist, von der entweder im Internet heruntergeladenen, oder von der im Laden auf CD-ROM gekauften Software so viele Kopien erstellen und weitergeben zu dürfen, wie man möchte. Gewissermaßen ist dies eine Umkehrung herkömmlicher Lizenzmodelle, bei der die Anzahl der Kopien eingeschränkt werden soll. Viel interessanter aber ist die Bestimmung, dass bei aller Software der Quellcode, also das, was bei proprietärer Software Firmengeheimnis ist, mit angeboten werden muss. Man kann also den Programmtext lesen, umschreiben und für eigene Programmtexte verwenden. Freie Software, deren bekanntestes Projekt als eigenständiges Computerbetriebssystem seit nunmehr etwas über zehn Jahren unter dem Namen „Linux" firmiert, versteht sich dementsprechend im Grunde als Software von Programmiererinnen für Programmierer. Hinter dem Konzept von Linux stecken die Prinzipien eines wesentlich älteren Betriebssystems, nämlich Unix. Unix aber hat nie den Sprung aus den großen Rechenzentren auf die PCs der inzwischen vielen Millionen Computerbenutzer geschafft. Einer der Gründe ist sicherlich, dass Unix als viel zu kompliziert gilt, zumal das Arbeiten auf der Kommandozeile bei einem effektiven Umgang mit Unix schlicht nicht wegzudenken ist. Ähnliches gilt nach wie vor für den Unixabkömmling Linux.

Nun ist es nicht so, dass es etwa unter Windows keine Kommandozeile mehr geben würde, und es ist auch nicht so, dass es für Linux keine grafischen Benutzeroberflächen gibt, aber anders herum wird ein Schuh daraus: Im Gegensatz zu Windows braucht es unter Linux keine Benutzeroberfläche, hingegen aber zwingend den Befehlszeileninterpreter, da das Starten der einzelnen Systembestandteile über Skripte abgewickelt wird. Dies hat, nebenbei bemerkt, den entscheidenden Vorteil, dass man jederzeit nachlesen kann, wie das System funktioniert. Hieraus resultiert aber auch, dass man sich

als Nutzerin geradezu dazu genötigt sieht, sich mit Skripting auseinander zusetzen, und das heißt: zu lesen. Sollte das System nicht mehr funktionieren, bleibt kein anderer Weg, als die zentralen Konfigurationsskripte zu lesen und solange zu suchen, bis man die Fehlerquelle gefunden und beseitigt hat. Allerdings muss ich mich gleich schon wieder korrigieren: Selbstverständlich gibt es einen anderen Weg, das System wieder zum Laufen zu bringen: Mit einer Neuinstallation aller Bestandteile. Dies wäre, polemisch formuliert, der grafische Weg.

Wie gesagt, es geht um keinen Paragone, aber es ließe sich zumindest festhalten, dass Linux weit mehr auf Schrift, auf Text beruht, als andere Betriebssysteme. Ob man dies nun als Vor- oder als Nachteil werten möchte, liegt sicherlich in den Vorlieben der jeweiligen Nutzer begründet. Es geht, grob gesprochen, um Bild vs. Text, und in dieser Hinsicht wäre auch die Beobachtung von Thomas Scoville in dem Aufsatz *Unix as Literature* zu verstehen, wenn dieser sich darüber wundert, dass viele seiner Kollegen, die statt mit Windows mit Unix arbeiten, schlicht gerne mit Wörtern arbeiten, ja ihnen sogar die „Unix-Sprache" zur zweiten Natur werde.[16] Das Arbeiten mit Unix sei ein „Tanz mit Worten". Begründet liegt dies in einer frühen Design-Entscheidung von Unix, nämlich weniger auf monolithische Strukturen zu setzen, sondern vielmehr auf eine ganze Reihe kleiner Utilities, auf Werkzeuge, deren Ausgabe mithilfe der so genannten Pipelines sogleich als Eingabe für ein anderes Programm verwendet werden kann.[17] So obliegt es der Kreativität der Nutzer, wie die Werkzeuge miteinander zu kombinieren sind, um aus ihnen eine neue, der Aufgabe genau angepasste Vorschrift, ein Programm zu schmieden.

Selbstverständlich muss eine Formulierung wie „Unix as Literature" als eine grobe Übertreibung erscheinen. Eine schwerlich zu überwindende Kluft liegt zwischen den Alltagssprachen und der Eingabe von Befehlskolonnen auf der Kommandozeile oder aber dem Verfassen von Algorithmen in einer der höheren Programmiersprachen. Die auch nur ansatzweise Gleichstellung formaler Sprachen mit der Alltagssprache führt meines Erachtens in die Irre, jedoch soll

16 Vgl.: Thomas Scoville: The Elements Of Style: UNIX As Literature, http://www.thomasscoville.com/PCarticle.html (zuletzt besucht am 11.10.2004).

17 Michael Hauben, Ronda Hauben: Netizens: An Anthology, Chapter 9, On the Early History and Impact of UNIX: Tools to Build the Tools for a New Millennium, http://www.columbia.edu/~rh120/ch106.x09 (zuletzt besucht am 11.10.2004).

auch nicht ausgeschlossen werden, dass in Zukunft Programmiersprachen entwickelt werden, die diese Kluft wenigstens teilweise überbrücken. Vielleicht lassen sich ja eines Tages die Maschinen im Dialog mit ihnen programmieren und die daraus resultierende Vorschrift ließe sich wie ein Gespräch lesen.

Die Zukunft des Lesens

Gegenwärtig gibt es hauptsächlich im Bereich der Freien Software außerordentlich viel zu lesen, da hier wie gesagt der Quellcode zur Verfügung steht, und entsprechendes Interesse vorausgesetzt, mag dies auch eine spannende Lektüre sein. Für die Programmierunkundigen bleiben jedoch nur die Dokumentationen, die allerdings, auch wieder bei der Freien Software, oftmals wirklich das lesenswert sind.

Die Frage wäre aber trotzdem die nach der Zukunft des Lesens, denn ich habe ja bereits angedeutet, dass es beim Computer nicht zuletzt auch um die Vorherrschaft von Bild oder Text geht, und zwar im Sinne von grafischer Benutzerumgebung vs. Kommandozeile. Der Abgesang auf die Schrift begann zwar mit Marshall McLuhans *Am Ende der Gutenberg-Galaxis* bereits gegen Ende der 60er Jahre, fand aber mit der massenhaften Verbreitung von Personal-Computern und erst recht mit der Durchsetzung des Internets und seiner grafischen Benutzeroberfläche, des WWW, dreißig Jahre später eine Art Renaissance. Einer der schillerndsten „Totengräber" der Schrift in Angesicht des Neuen Mediums Computer war sicherlich der 1991 bei einem Autounfall ums Leben gekommene Vilém Flusser. Bereits in seinem Essay *Ins Universum der technischen Bilder* wird die Vision einer Zukunft ausgebreitet, in der die Menschen gänzlich auf Lesen und Schreiben verzichten, um stattdessen mittels Bilder zu kommunizieren. Fast schon wie eine Erläuterung hierzu wirkt der darauf folgende Text *Die Schrift*, in dem Flusser im Medium der Schrift versucht darzulegen, warum Schrift an ihr Ende gekommen ist. Zwar ist es nicht so, dass Flusser die Schrift fröhlich verabschiedet, aber seine Diagnose steht zumindest für ihn fest. Hierbei führt er im Wesentlichen eine Art Teleologie der Medienentwicklung ins Feld, um diesen welthistorischen Zustand herzuleiten. Nun ist hier nicht der Raum, um Flussers durch und durch essayistisch vorgetragenen Gedanken noch einmal zu raffen und meinerseits vorzutragen, ohne dabei Gefahr zu laufen, dass meine Ausführungen über Flusser absurd erscheinen

müssen. Jedoch gibt es in dem Band *Die Schrift* einen Abschnitt mit Überlegungen zum Status „digitaler Codierungen", die im Zusammenhang mit diesem Vortrag recht interessant sind. Zum einen findet sich dort die Beobachtung, dass Rechenvorschriften, also Algorithmen, bildlich wahrgenommen werden. Hiermit stehen Algorithmen im Gegensatz zu alphabetischen Texten, die ihre Aussagen wie auf einer Perlenschnur aufreihen und strikt linear organisieren. Dieser Gegensatz beruht darauf, dass bei der Wahrnehmung eines Algorithmus das Auge zu kreisen beginne, wie bei der Betrachtung eines Bildes: „Die Zahleninseln im wissenschaftlichen Text sind als außerordentlich abstrakte, einem Diskurs unterworfene Bilder anzusehen."[18] Dem Auge stellt sich die Aufgabe, die Verbindungen zwischen den einzelnen Elementen eines Algorithmus' zu verfolgen. Gesetzt, man folgt der vorgeschlagenen Charakterisierung alphabetischer Texte als lineare Strukturen, dann wird recht schnell klar, dass es bereits beim Programmieren nicht mehr um das Verfassen von Texten geht. In der Tat liegt eine der wichtigsten Merkmale von Programmiersprachen in Rekursionen, das heißt, dass eine Funktion immer wieder auf sich selbst zurückweist, solange, bis eine Abbruchbedingung gegeben ist. Weiterhin stellen alle Programmiersprachen Befehle bereit, um Anweisungen erteilen zu können, wann was wie geändert werden soll, zudem muss es die Möglichkeit geben, mit Wenn-dann-Strukturen innerhalb eines Programms auf die unterschiedlichsten Ergebnisse reagieren zu können. Immer geht es darum, etwas zu machen, wenn etwas eintritt. Die Frage wäre nur, ob Algorithmen in der Tat dem Bild näher stehen als dem linearen Text, jedenfalls aber sind sie nicht umstandslos den Texten zuzuordnen.

Intern aber, abseits von allen Programmiersprachen, verwandeln sich die Algorithmen in Daten, in Bitfolgen, oder wie Flusser sagt, in „pickbare Haufen".[19] Diese Geflügelmetapher, die pickbaren Haufen, aus denen ein dummes Huhn sich sein Futter herausklaubt, ist von Flusser nicht willkürlich gewählt. Zum einen bezeichnet er damit in der Tat das, was bei der Analog-digital-Wandlung vor sich geht. Aus Bildern und Tönen werden ebenso wie aus Befehlen und Texten unterschiedslos Nullen und Einsen geformt, die sich im Prinzip ebenso unterschiedslos manipulieren, oder besser: komputieren lassen. Die Einschrän-

18 Vilém Flusser: Die Schrift, Göttingen 2002, S. 27.

kung liegt im Grunde dann nur noch darin, ob die manipulierten Daten nach ihrer Manipulation überhaupt noch einen Sinn ergeben. Sinn aber wäre in der von Flusser projizierten Zukunft ohnehin eine obsolete Kategorie, da man sich in Zukunft viel eher nach solchen Kategorien wie „informativ" oder „interessant" richten wird. Alles kann mit allem kombiniert werden, etwa das demokritsche Atommodell mit der einsteinschen Relativitätstheorie, ohne dass es wichtig wäre, welche Theorie stimmiger ist.[20] Insofern wäre Flussers Vision eines Universums der technischen Bilder davon gekennzeichnet, dass sich die in ihm lebenden Subjekte ebenso unterschiedslos am Kulturerbe bedienen. Das Kulturerbe selbst wird zu einem pickbaren Haufen, aus dem es gilt, rein nach Fragen des Informationsgehalts und nicht nach Sinn auszuwählen.

Die Zukunft Flussers ist nicht bestimmt von analytischen Subjekten, von Lesern, sondern von informationssynthetisierenden Subjekten, von Schreibern. Diese Schreiber wären ähnlich den „Skript Writern", womit Flusser die Autoren etwa von Drehbüchern bezeichnet, die etwas verfassen, das später in Bilder umgesetzt werden soll. Skript Writer schreiben schon nicht mehr an Menschen, sondern bereits an Apparate, wobei im Sinne Flussers etwa eine Filmproduktion schon als Apparat aufgefasst wird. Für ihn bilden die Skript Writer damit bereits den Übergang von der alphabetischen in die informatische Kultur.[21] In dieser Kultur ist dann jeder Mensch in die Lage versetzt, aus dem instantan zur Verfügung stehenden, digital umcodierten Material eigene Bilderreigen zu kreieren. Gedacht ist eine Zukunft grenzenloser, durch die Apparate ermöglichter Freiheit.

Kaum zu verhehlen ist, dass ich trotz des Themas „Lesen" immer wieder beim Schreiben angelangt bin. Mag sein, dass es an meiner mangelnden Disziplin gelegen hat, mag aber auch sein, dass es sehr wohl mit dem Gegenstand, der gelesen werden soll, dem Computer zu tun hat. Dies ist dann der letzte Punkt, auf den ich kurz hinaus will. So sehr Kittlers Texte, um den Beginn des Vortrags wieder aufzugreifen, von der Behauptung gekennzeichnet sind, dass alle Medientechnologie im Grunde Kriegstechnologie sei, so sehr steckt dahinter der heimliche Imperativ, ebendiese Kriegstechnologie für eigene Zwecke einzuset-

---

19 Ebd., S. 30

20 Vgl.: Vilém Flusser: Ins Universum der technischen Bilder, Göttingen 1999, S. 138.

21 Vgl. ebd., S. 128ff.

zen. So heißt einer von Kittlers Aufsätzen *Rockmusik – Ein Missbrauch von Heeresgerät*[22], beim dem es nur am Rande um Computer geht. Nichtsdestotrotz kulminiert der Aufsatz aber in der Formulierung, dass es darum gehe, die Geheimwaffen des Zweiten Weltkrieges, und gemeint ist hiermit der Computer, „für Decodierungen zu missbrauchen."[23] Decodieren heißt aber in diesem Zusammenhang nichts anderes als Hacken und mit dieser Forderung ist Kittler denkbar weit von Flussers Vision einer Zukunft der technischen Bilder entfernt. Wo Kittler das Öffnen der „Black-Box" des Computers fordert, predigt Flusser die Oberflächlichkeit im Sinne eines von allen technischen Obliegenheiten entlastetem Synthetisierens von Bildern.

Beide decken sich allerdings in der Hoffnung, man könne den Computer verwenden, um zu tun, was man tun möchte, wobei eben nicht differenziert werden soll in die, die schreiben, und in die, die nur lesen. Die alte Forderung, eine zentralistische Sender-Empfänger Struktur sei aufzugeben, um an ihrer Stelle alle Menschen zu Sendern zu machen, findet im Computer eine Reaktualisierung. Nun wäre dies, was doch bei Video und Fernsehen ebenso wenig funktionierte wie zuvor beim Radio, in Hinsicht auf den Computer leicht als ebenso naiv abzutun. Und es wäre in der Tat ebenso naiv, wenn nicht durch das Internet eine weitgehende Vernetzung der Computer eingesetzt hätte. Für die mit einem Rechner und Netzwerkanschluss ausgestatteten Menschen ist es so unmittelbar möglich geworden, zu einem Sender zu mutieren. Wohingegen das Betreiben einer Seite im WWW optional ist, wird man in den File-Sharing-Netzwerken sogar automatisch gleichzeitig zum Sender. Hier liegt aber auch gleichzeitig das Problem. Gerade durch den unkontrollierten Tausch von Dateien, also Computerprogrammen, Musik, Büchern, Bildern und Filmen setzt die Tendenz ein, dass die bisherigen zentralen Sender, also die, denen die Rechte an all diesen kulturellen Produkten gehören, das illegitime Senden unterbinden möchten. Unterm Stichwort vom „Digital Rights Management", an dem sich die meisten namhaften Anbieter von Musik oder Filmen sowie die Produzenten von Software und Hardware beteiligen, soll sogar weitergehend kontrolliert werden, wer wie oft ein Buch lesen, ein Musikstück hören, einen Film sehen darf. Digital

22 In: Theo Elm, Hans H. Hiebel (Hg.): Medien und Maschinen. Literatur im technischen Zeitalter, Freiburg 1991, S. 245-257.
23 Ebd., S. 257.

Rights Management soll effektiver, als es bisher möglich war, dafür Sorge tragen, dass auch technisch nicht mehr möglich ist, was rechtlich nicht erlaubt ist. Kulturgut soll nicht zu einem pickbaren Haufen werden, sondern Ware bleiben, allerdings eine Ware, auf die zurückzugreifen jedes Mal eine Lizenz erfordert und die nicht weitergegeben werden darf. Und wo wir schon mal beim Thema Lesen sind: Die Firma Adobe hatte in einer Lizenz für die Downloadversion von *Alice im Wunderland* das laute Vorlesen des Buches untersagt.[24] Zwar wurde diese Bestimmung später wieder gestrichen, doch dürfte an diesem Beispiel deutlich werden, in welche Richtung der Zug rollt.

Bliebe aber das Leben an der Oberfläche, das Flusser propagiert, durch Digital Rights Management technisch reglementiert, dann wäre nicht nur die US-Bürokratie „ins Silizium versenkt"[25], dann hieße Protected Mode die effektive Kontrolle über das so genannte „Geistige Eigentum". In diesem Falle bliebe der Imperativ Kittlers bestehen, die Geheimwaffen des Zweiten Weltkriegs für Decodierungen zu missbrauchen oder aber das Modell der Freien Software als „Creative Commons" auch auf die eigene geistige Produktion anzuwenden.[26]

24 Vgl.: Dietmar Kammerer: Rettet die Privatkopie, Junge Welt, Onlineausgabe vom 3.8.2002, http://www.jungewelt.de/2002/08-03/010.php (zuletzt besucht am 11.10.2004)

25 Kittler: Protected Mode, a.a.O., S. 214.

26 Vgl. zu alternativen Lizenzmodellen: http://www.creativecommons.org