LUGS – Linux User Group Switzerland

FLUGS

Vereinsnachrichten

1-2/95

Terminliste

April

1.	9:00…16:00		Computerbörse GO TO, Kongresshaus Zürich	
2.	10:00…16:00		Computerbörse GO TO, Kongresshaus Zürich	
6.	19:00	Norbert Kümin	LUGS Treff, HTML – Hyper Text Markup Language (WWW)	
20.	19:00		LUGS Treff	

Mai

4.	19:00		LUGS Generalversammlung, Rest. Palmhof, Zürich	S. 6
18.	19:00		LUGS Treff	

Juni

1.	19:00		LUGS Treff	
10.	9:00…16:00		Computerbörse GO TO, Kongresshaus Zürich	
15.	19:00		LUGS Treff	

Juli

6.	19:00		LUGS Treff	
20.	19:00		LUGS Treff	

Die Themen für die noch offenen LUGS Treffs werden nach der Generalversammlung bekanntgegeben.

Hoi zäme

Liebe LUGS Mitstreiter

schon ist beinahe ein Jahr vergangen. Die Linux-Usergroup ist langsam gewachsen, und wir alle suchen immer noch die Form der weiteren Entwicklung. Nun, ich fühle mich als Präsident dazu berechtigt, der Entwicklung beizuwohnen, ohne sie forcieren zu wollen. Das Bestehen einer Linux-Gruppe ist nicht so einfach und populär wie das eines anderen Computerclubs oder eines anderen Betriebsystem-Vereins.

LINUX, und UNIX im generellen, ist im Heimbereich eine unerschöpfliche Philosophie. Eigentlich kann das Betriebsystem niemand brauchen. Es gibt bis jetzt keine Textverarbeitung, auf die es sich zu warten lohnt, keine Tabellenkalkulation, die noch mehr Funktionalität als unnütze, aber teure Erweiterung in das eigene Computersystem eingliedert, es gibt keinen König, den man anbeten kann, und dessen Reich verteidigt werden muss. Linux entwickelt sich einfach so dahin. Und doch spürt jeder, der sich damit beschäftigt, dass sich da etwas entwickelt, das unverständlich, tief und weit ist.

Wollte ich früher noch einen Computer, um die neusten Textverarbeitungen erlernen zu können, um die raffiniertesten Spiele spielen zu können, um bei einer Entwicklung dabei zu sein, so hat sich vieles geändert. Der Computer ist zum Meta-Mittel geworden. Es ist gerade so interessant geworden, über Computer und Betriebsysteme in einem Verein zu sprechen, als mit dem Computer etwas direkt zu machen.

Linux gibt die Möglichkeit, saubere Entwicklungen zu machen. Noch besser, Linux erlaubt, aktiv an einem Netzwerk mitbeteiligt zu sein. Linux wird zur Kommunikationsmaschine für jedermann. Da diskutieren die grossen sieben (G7) über das Recht, das jedem zusteht, auf der Datenautobahn mitzufahren, und Verträge werden ausgehandelt, wer damit das grosse Geld verdienen darf. Mit Linux wird dies bereits praktiziert. Es ist stabil und schnell genug, eine Datenbank für jedermann anzubieten, Files weltweit zu offerieren. Ungeahnte Kommunikationsgruppen bilden sich weltweit. Menschen diskutieren in Gruppen.

Das Internet ist die mir erste bekannte Möglichkeit, die es zulässt, dass viele Menschen zur gleichen Zeit an einem Thema diskutieren. So als ob viele Steinhauer zusammen an einem Stein hauen, um eine Skulptur zu erschaffen. Linux selbst ist so eine Skulptur. Der Erfolg ist unerwartet gross. Kleine Firmen, die wohl auch Grosses leisteten, Firmen wie Coherent, werden vom Markt verdrängt. Individualismus ist einmal stärker als Profit.

Grosse Firmen haben es in der gleichen Zeit nicht fertig gebracht, ähnlich stabile Werkzeuge auf den Markt zu bringen. Die Verunsicherung bei vielen Leuten bleibt aber trotzdem, erzeugt durch einen enormen Werbeaufwand. Die Unterstützung für LINUX von Softwareherstelleren ist noch nicht besonders gross, obwohl die Produkte oft nur geringe Anpassungen brauchen. Fürchtet man sich hier vor Repressalien?

Was sollen wir mit unserem neuen Werkzeug beginnen? Wir suchen immer noch in unserem alten Denken. Ich höre immer wieder, dass wir unseren alten PC mit einem stabilen Betriebsystem aufwerten wollen. Einige finden, dass man sich endlich von der Unterdrückung einiger Softwarehersteller befreien muss. Trotzdem. Ich persönlich brauche kein LINUX, um das Windows zu ersetzen. Dies kann ich nicht einmal mit dem neuen Windows-Emulator NT. Die Software, die ich auf dem Windows benutze, läuft stabil und genügend gut. Ich habe überhaupt kein Verlangen, noch komplizierte Systeme bedienen zu müssen, wenn ich einen einfachen Brief oder gar ein Buch schreibe. Die Bedürfnisse, die ich da hätte, kann die Software schon lange nicht mehr lösen. Ich brauchte eher einen flacheren, gesünderen Bildschirm oder einen kleineren, moduläreren Computer.

Und LINUX? Der Wege, den LINUX zeigt, hat viele offene Enden. Als günstiges Entwicklungs-System für professionelle Entwickler hat LINUX bereits seine Berechtigung verdient. Mailserver, FTP-Server, Datenbankanbieter, WWW-Server, Fire-Wall-System sind bereits gängige Anwendungen unter LINUX. Doch ich glaube, dass LINUX nur eine Entwicklungsstufe zu einem allumfassenden Kommunikationssystem ist. Und die Folgen eines solchen Systems können wir bis jetzt kaum einschätzen.

Es werden sich wohl positive wie negative Strömungen mit diesem neuen Element beschäftigen. Schon jetzt zeigen beide Richtungen massive Einflüsse. Die Computerkommunikation ist bereits jetzt ein gefürchteter Gegner in der modernen Kriegführung.

Was wir machen können, ist das faire Umgehen mit dieser Komponente zu erlernen, und offen zu sein gegenüber diesem neuen Medium. Die Kommunikation gehört vielleicht bereits der nächsten Generation.

Kahlil Gibran sagt über die Kinder: Ihr dürft ihnen euere Liebe geben, doch nicht eure Gedanken. Denn sie haben ihre eigenen Gedanken.

Wie soll es mit unserem Verein weitergehen? Ich denke, dass es wichtig ist, mit dem Werkzeug LINUX umgehen zu lernen. Dies können wir erreichen, indem wir uns gemeinsam mit einzelnen Teilen des Betriebsystems beschäftigen. Dies verlangt mehr Disziplin und Einsatz von Einzelnen. Der Club lebt von den Aktivitäten von Einzelnen.

Wir müssen die Mitgliederzahl des Vereins erhöhen. Ein kleiner Verein ist in seinen Möglichkeiten eingeschränkt. So denke ich, dass wir uns nach Aussen öffnen müssen, dem Anfänger Hilfen bieten, sowie dem Profi eine interessante Diskussionsumgebung ermöglichen.

Die Präsenz an Ausstellungen soll nach Möglichkeit verstärkt werden. Dies kann aber nur auf Basis von freiwilligen Aktivitäten erreicht werden.

Ich wünsche Euch weiterhin viel Spass mit LINUX, mit LUGS, und hoffe, Euch alle an der ersten Generalversammlung am 4. Mai im Restaurant Palmhof begrüssen zu dürfen.

Ralph Steigrad

Einladung zur ordentlichen Generalversammlung der LUGS

Donnerstag 4. Mai 1995, 19 Uhr 00
Restaurant Palmhof
Universitätsstr. 23
8006 Zürich
Tel 261 69 90

Traktanden:

1. (a) Begrüssung und Mitteilungen
   (b) Wahl der Stimmenzähler
   (c) Wahl des Protokollschreibers
2. (a) Jahresbericht des Präsidenten
3. (a) Jahresrechnung 1994
   (b) Abnahme der Jahresrechnung 1994 und Entlastung der Vorsteherschaft
4. Wahlen
   (a) Wahl der Vorsteherschaft
   Der Vorstand schlägt das alte Team zur Wiederwahl vor
   (b) Wahl der Revisoren
5. Festlegung der Beiträge 1995
   (a) Vorschlag Vorsteherschaft
   (b) Diskussion
   (c) Abstimmung
   (d) Budget 1995
   (e) Abnahme des Budget 1995
6. Aufnahmeritual neuer Bewerber
   (a) Vorschlag Präsident
   (b) Diskussion
   (c) Abstimmung
7. Aktivitäten 1995
   (a) Vergrösserung Verein
   (b) Raum LUGS
   (c) Aktivitäten nach Aussen
8. Antrag Daniel Etter
   Spende an Linux Entwicklergruppen
9. Varia

Nach der GV gemütliches Zusammensein.

Ralph Steigrad

Erweiterte Redaktion — Neue Konzepte?

“Diese Heft enthält Hinweise für Autoren”, …

…steht in manchen Heften, die auch von den Lesern mitgestaltet werden. Von den Lesern mitgestaltet werden viele technische und wissenschaftliche Hefte. Die Leserschaft und die Autorinnen und Autoren sind die gleichen. Sie veröffentlichen die Ergebnisse der Arbeit, die sie geleistet haben oder berichten über die Erfahrungen, die sie gemacht haben.

In dieser Weise verstehen wir auch FLUGS. Nicht, dass nur Artikel von wissenschaftlichem Gehalt hier erscheinen sollten. Nein, wir denken, dass diese Hefte uns eine Hilfe beim Erfahrungsaustausch über den Computer sein sollen. So sind Artikel über Technik, Wissenschaft und besonders über Computer sehr erwünscht. Gemäss unseren Statuten müssen wir jedoch auf religiöse und politische Themen verzichten.

Neue Konzepte? Das Heft stammt aus mehreren Computern! Das wirklich Neue am FLUGS des Jahres 1995 ist die Teamarbeit. Roland Alder, Patrick Seemann und Martin Wunderli werden das Heft mit Erich Stüssi als verantwortlichem Redaktor gestalten (sprich beim Setzen der Artikel helfen und manchmal auch selber etwas schreiben). Dabei wird gegen aussen nichts von dieser Zusammenarbeit ersichtlich sein, denn ein ausgeklügeltes System wird die Arbeit auf das Team verteilen und die PTT reicher machen[1]. “The Making of FLUGS” wird sicher in diesem Jahr in einen kurzen Magazinbeitrag einfliessen!

Wie bringe ich nun meinen Beitrag in das Heft? Also zuerst muss jeder wissen, dass das Heft in LaTeX gesetzt wird. Denn die Voraussetzung dafür, dass das Heft dezentral produziert werden kann, ist ein System, dass die verteilte Produktion nicht nur zulässt, sondern auch unterstützt und welches alle Beteiligten (möglichst) kostenlos bekommen können. Das Verschicken der Artikel per Email braucht mit LaTeX auch keine Handstände wie uuencode/tar/compress oder Ähnliches.

Die Texte sollten also wenn möglich unter Verwendung der Vorlagen aus `lugsleer.tgz`[2] in LaTeX gesetzt werden. Eine exzellente LaTeX-Anleitung ist als DVI oder PS-File ebenfalls erhältlich. Fehlerhafte Sourcen und “HalbgeTeXtes” verursachen leider mehr Schaden als Nutzen[3]. Wer als Linuxer noch keine TeX-Installation hat, dem sei die teTeX Distribution sehr

[1] vor allem das erste
[2] siehe Artikel “TeX– Kann man das essen?” ab Seite 9
[3] Nach Möglichkeit Fehlerfreiheit durch Übersetzen testen

empfohlen (z.B. von ftp.switch.ch oder sunsite.unc.edu per ftp downloadbar). Absolut einfach zu handhaben und mit extrem gutem Installationsscript und Installationsanleitung.

Wer sich nun nicht zum TEX-Spezialisten mausern will, der kann einfach ganz normale ASCII-Dateien einsenden. Abschnitte sollen dabei durch zweimal Return erzeugt werden, Punkte von Aufzählungen ebenfalls. Bilder können als Postscript Pixel- oder Vektordateien beigefügt werden. Dabei wird im ASCII-Text in einem eigenen Abschnitt einfach "hier Bild Datei.xxx einfügen" vermerken, im LATEX Text bitte den Style epsf zur Einbindung benutzen. Bilder sind schön, doch sie verzögern auch die Produktion, weil die Dateien unter Umständen nicht mehr per e-Mail transportiert werden können, sondern mit der Schneckenpost[4] reisen oder speziell bearbeitet müssen! Wer die Pixelbilder in einem Nicht-Postscript Format hat, dem sei das pbmplus Paket ans Herz gelegt. Zur Not können aber auch wir die Konvertierung erledigen.

Wie kommen die Artikel nun in die Redaktion? Per Internet oder per Post mit Diskette (frühzeitig aufgeben!!) an den Redaktor. Er verteilt die Arbeit dann. Sicher kann man die Dateien auch an einem Clubabend mitgeben. Oder man übergibt sie einem Kollegen, der im Internet aktiv ist!

Im nächsten Heft, 3/95, sollten alle Termine mit den Vorschauen bis und mit dem 31. Oktober enthalten sein. Vorschauen? Ja, das soll etwas Neues werden. Soweit wie möglich sollen Abende mit geplanten Themen durch einen (1/2-seitigen oder grösseren) Artikel vorbereitet werden. In der folgenden Tabelle seht Ihr, wie die weiteren Redaktionschlüsse geplant sind (3-4 pro Jahr). Man muss aber anmerken, dass es bei gesteigerter Nachfrage auch mehr Ausgaben werden können. Insbesondere wenn die Treff-Aktivität zunehmen würde. Aber diese Änderungen würden dann in einem der folgenden Hefte angekündigt.

Nr.	Red. schluss	Versand bis	Termine bis
3/95	16.6.95	14.7.95	31.10.95
4/95	15.9.95	13.10.95	31.1.96
1/96	15.12.95	12.1.96	30.4.96

Anmerkung: Ehemalige ACC-Mitglieder (Atari Computer Club Zürich) werden das Layout und auch diesen Artikel in Grundzügen wiedererkennen. Das kommt nicht von ungefähr. Zum einen wurden die LATEX Definitionsdateien des FLUGS auf der Basis des ACCtuell erstellt. Zum anderen sind 3 von 4 Mitgliedern des jetzigen Redaktionsteams ehemalige Mitglieder des ACCtuell Redaktionsteams. "Und das ACCtuell?" werden jetzt vielleicht einige fragen. Gibt es nicht mehr. Die ACC Vereinsnachrichten heissen jetzt 'ACC im Aufwand' und sind eine unregelmässig erscheinende Lose-Blatt-Sammlung…

Martin Wunderli

[4]= Briefpost

TEX: Kann man das essen?

Dieser kleine Artikel soll kurz umreissen, was TEX einem Benutzer an Möglichkeiten bieten kann. Er soll kein Kurs sein, da dies den Rahmen dieses Artikels sprengen würde. Wer sich weiter interessiert, kann sich in die am Ende des Artikels angegebene Literatur vertiefen.

Allgemeines

TEX (gesprochen [tech]) ist ein Computer*satz*programm. Es wurde von Professor Knuth zum Setzen seiner Buchreihe "The art of computer programming' entwickelt und ist Freeware (Das Geld, das man also eventuell für eine TEX-Implementation zahlt, ist 'nur' für die Portierung und Anpassung an einen bestimmten Computertyp). Heute arbeitet aber praktisch niemand mehr direkt mit TEX, sondern mit einem von Leslie Lamport und anderen entwickelten Makropaket namens LATEX, dass dem Benutzer eine grosse Anzahl mächtiger und einfach zu benutzender Befehle anbietet und auf TEX aufbaut. Wenn in Zukunft also von TEX gesprochen wird, dann ist LATEX gemeint. Das nackte TEX wird als 'plain TEX' speziell gekennzeichnet. Die aktuelle Version von LATEX ist 2ϵ vom 1.12.1994. Die nächste wird übrigens wieder 2ϵ sein, aber vom 1.6.1995.

TEX ist keine Textverarbeitung. Und auch kein DTP. TEX ist sowas wie ein Compiler für Texte. Womit wir schon beim grössten Nachteil für viele Anwender wären: Kein WYSIWYG, TEX-Texte sind reine ASCII-Texte mit (ASCII-)Steuerbefehlen (Kleine Ausnahme: Es gibt die Möglichkeit, 8 Bit Zeichen zu benutzen. Halte ich aber für sehr schlecht, weil nicht mehr beliebig zwischen allen Rechnern portabel).

Aufbau eines TEX-Textes

Reiner ASCII-Text heisst: Keine system- oder programmspezifischen Steuerzeichen. TEX-Texte können also beliebig zwischen verschiedenen Computern und Betriebssystemen hin- und hergeschoben werden[5]. Und zwar ohne irgendwelche Konvertierungsprogramme. Ergo sind mit TEX geschriebene Texte "zukunftssicher" (Das heisst, dass ich diese Clubzeitschrift vermutlich noch ausdrucken kann, wenn es den Club schon lange nicht mehr gibt. Beeindruckend, gell? `:-)`). Vor allem: TEX gibt es nun auf wirklich `jedem` einigermassen verbreiteten Computer. Als da wären: UNIX-Systeme, VAX-VMS, ATARI, Mac, Amiga, Archimedes, DOSen usw. (ach ja, auf dem NeXT gibt es natürlich auch ein TEX…).

Um den TEX-Compiler mitzuteilen, dass man einen Text in einer bestimm-

[5] Dieser Text wurde vor mehr als 3 Jahren auf einem ATARI TT erstellt. Danach 'wanderte' er auf eine SUN (SunOS 4.1.3) und hatte ein Gastspiel auf einem Mac PowerBook. Nach einem Abstecher auf eine weitere SUN (Solaris 2.3) ist er nun auf einem Linux PC (1.2.1) beheimatet.

ten Art und Weise formatiert haben möchte (Überschriften, Aufzählungen, Beschreibungen, Tabellen usw.), gibt es Steuerbefehle, die (fast) alle mit einem Backslash '\' anfangen. Es ist aber zum Glück so, dass man nicht allzuviel verschiedene Befehle kennen muss, um ein einfaches Dokument schreiben. Aber genau wie man halt ein Handbuch zu einer Textverarbeitung lesen muss, muss man mindestens ein paar Seiten aus [1] inhalieren, bis es klappt. Nichtsdestotrotz ist das Standard TEX Buch [3] immer noch deutlich dünner als das MS Word Handbuch…

Ein minimaler (deutscher) TEX-Text sieht dann etwa so aus:

```
\documentclass{article}
\usepackage{german}
\begin{document}
Hier kommen die Abs"atze, welche
durch Leerzeilen getrennt sind.
\end{document}
```

Mit `\documentclass` wird der Dokumenttyp festgelegt (z.B. ob es Kapitel gibt oder nicht). Danach kommen Zusatzpakete, die nicht im Standard LATEX enthalten sind, in diesem Fall Zusatzbefehle für deutsche Texte, z.B. Umlaute. Mit `\begin{document}` zeigt man an, dass es jetzt richtig los geht und mit `\end{document}`, dass der Text fertig ist.

Wie entsteht ein TEX-Dokument?

Da TEX ein *Compiler* für Texte ist, ähnelt auch die Benutzung dem eines 'normalen' Compilers für Programmiersprachen. Das heisst natürlich auch, dass Leute, die ab und zu programmieren, sich am Anfang einfacher zurechtfinden als solche, die Programmieren nur vom Hörensagen kennen. Mit der Zeit (und der Übung) klappt es aber bei jedem ganz gut. Und wenn eine komfortable Umgebung wie das Emacs Makropaket AucTeX einem die mühsamen Dinge abnimmt, dann vereinfacht sich die Bedienung nochmals.

Das Erstellen eines Dokumentes, vom Tippen bis zum Ansehen oder Ausdrucken, geht mit TEX etwa so vor sich.

1. Zuerst den Text in einem Texteditoreditor schreiben (Emacs, Vi, Joe, nedit etc.). Es entsteht eine Datei mit der Endung `tex`.

2. Mit dem TEX-Compiler übersetzen (und solange wieder zu 1. bis kein Fehler mehr auftritt: Denn in den Satzbefehlen kann es ja Fehler haben!).

3. Das (fast) fertige Produkt (Eine Datei mit der Endung `dvi`, das ist die Abkürzung für *de*vice *i*ndependent, also geräteunabhängig) kann nun auf dem Bildschirm betrachtet (xdvi) oder ausgedruckt werden. Wenn das Ergebnis nicht passt, dann gehe über 1 und ziehe keine 4000.- ein…

Warum aber TEX? Wo doch alles nur schwieriger wird, man den Text nicht direkt in seiner endgültigen Form sieht und der Rechner bis an seine Leistungsgrenze beansprucht wird?

Nun, es gibt neben der Textportabilität tatsächlich ein paar nicht zu unterschätzende Vorteile:

- TEX verteilt die Leerstellen zwischen den Wörtern im Blocksatz nicht zeilen- sondern absatzweise. Das bedeutet, dass es nicht passieren kann, dass z.B. zwei Zeilen 'vollgepackt' sind aber die nächsten 'ausgedünnt' erscheinen. Ein Absatz hat also einen einheitlichen Schwärzungsgrad. (Bei einem WYSIWYG-System genügt die Rechenleistung eines durchschnittlichen Computers für so ein Feature nicht).

- Genauso wie der Platz zwischen den Wörtern gleichmässig über einen Absatz verteilt wird, wird auch der Platz zwischen Absätzen selbst über eine Seite gleichmässig verteilt. Das Ergebnis ist eine sauber gefüllte Seite.

- Durch das (Mehrpass-)Compilerkonzept und den Satzbefehlen fallen Dinge wie Inhaltsverzeichnis, Abbildungsverzeichnis, Index, Glossar sowie Kapitel und Seitenreferenzen 'so nebenbei' an. Dass Seitennummern und Kapitelnumerierung automatisch erzeugt und bei Bedarf angepasst werden, ist ja klar…

- Die Programmierung von neuen Makros (LATEX-Befehlen) ermöglicht selbst dem Laien die einfache Zusammenfassung oft benötigter Arbeitsschritte und ermöglicht dem Profi die Gestaltung neuer Features, die eigentlich nicht in TEX enthalten sind. Herausragende Beispiele sind der Mehrspalten-Seitenstil sowie Makros num Setzen von Musiknoten und chemischen Formeln.

- Zum Erzeugen von Standardtexten (Artikel, Berichte, Bücher, Briefe) stehen vordefinierte Seitenstile (Zusatzmakropakete zu LATEX) zur Verfügung, die einem mit wenig Lernaufwand die Erzeugung qualitativ hochwertiger Texte ermöglichen. So haben wir mit dem Setzen von Artikeln fürs FLUGS relativ wenig Mühe, nachdem der Stil für das Clubheft einmal definiert wurde.

- Die Möglichkeit, hochkomplizierte (d.h. beliebig verschachtelte) Formeln und umfangreiche Tabellen zu setzen, erleichtert das Schreiben von technischen und wissenschaftlichen Texten enorm.

- Durch das Konzept der DVI-Dateien sind die übersetzten TEX-Texte nicht an ein bestimmtes Ausgabegerät gebunden. Man kann also zuhause einen Text schreiben, übersetzen und den Text auf einem Nadeldrucker oder Laser ausdrucken. Die gleich DVI-Datei, die man für den 9-Nadler Ausdruck verwendet hat, kann man nun aber auch einer Druckerei geben, die sie dann z.B. auf einer 2400dpi Linotype belichten kann.

- Die freie Verfügbarkeit von TEX ermöglicht nicht nur den preiswerten Erwerb des Satzsystems an sich, sondern sie bewirkt auch, dass viele Zusatzprogramme geschrieben werden, die ebenfalls in der Regel Freeware sind. Beispiele sind das Programm BibTEX das Literaturdatenbanken verwaltet, das Programm MakeIndex, das bei der Indexerstellung hilfreich ist (diese gehören heutzutage praktisch zu jeder TEX-Installation), sowie diverse Tools, die die Ausgabe von TEX-Texten auf Postscriptdruckern ermöglichen.

Natürlich hat TEX auch Nachteile, die einen manchmal hart treffen können. Mich persönlich haben sie bis jetzt nicht besonders gestört, aber sonst hätte ich natürlich auch nicht TEX zu meiner Lieblingstextverarbeitung erkoren (na ja, die Bezeichnung Textverarbeitung stimmt ja nicht so ganz…). Andere Leute haben da andere Ansichten.

Die Nachteile:

- Die Seitenstile sind zwar schön und in den meisten Fällen ausreichend, aber wenn man etwas Extravagantes machen will, dann dauert es seine Zeit. Da merkt man dann, dass TEX kein DTP ist, sondern ein Satzsystem ‘herkömmlicher’ Art: Ich würde z.B. nie ein Plakat mit TEX gestalten…
- Grafik und TEX ist auch so ein Thema. Aber inzwischen ist das Einbinden von Postscript kein Problem mehr und so steht einem eine Unzahl guter Tools (idraw, xfig, tgif etc.) zur Verfügung.

Die obigen Punkte waren natürlich nur ein kleiner Ausschnitt aus dem Leistungsspektrum von TEX. Wer sich mehr dafür interessiert, dem empfehle ich teTEX. teTEX ist eine TEX-Distribution, die wirklich kinderleicht zu Installieren ist. Ein paar Megabytes (ca. 35) freier Plattenplatz sind aber schon die Voraussetzung….

Da es zu TEX/LATEX-Installationen in der Regel nur eine Installationsdokumentation gibt, ist man für TEX an sich auf externe Literatur angewiesen. Für praktisch alle reinen Anwender genügt aber [3], zum (preisgünstigen) Einstieg sogar eine Kopie von [1].

[3] ist im Prinzip das Handbuch zu LATEX und es ersetzt damit die englische Originalausgabe von Leslie Lamport (welche nie ins Deutsche übersetzt wurde und auch stilistisch ziemlich mühsam zu lesen ist. Es gibt aber eine neue Auflage für LATEX2ϵ). [4] behandelt in erster Linie eine ganze Anzahl der zu LATEX existierenden Styles und Zusatzprogramme (Es ist damit die deutsche Übersetzung und Erweiterung der Manuals und fürs ‘Drumherum’), widmet sich aber auch der Erstellung eigener Seitenstile. Als sehr gute Alternative zu den beiden Kopka Bänden wäre [1] zusammen mit dem LaTeX Begleiter [2] zu nennen, die sich vielleicht etwas mehr an die Schnelleinsteiger und extrem Wissensdurstige richtet.

Wenn man sich ans Eingemachte (d.h. plain TEX) heranwagt, dann steht einem neben der (ebenfalls nicht übersetzen) englischen Originalversion des TEX-Books von Knuth, die Einführung von Schwarz ([5]) zur Verfügung.

Und die Zeichensätze?

Zum Schluss noch ein paar Worte zu den Zeichensätzen. TEX-Zeichensätze sind in Ihrer ursprünglichen Form Vektorzeichensätze, deren Definition als ASCII-Text vorliegt. Diese Vektorzeichensätze werden von einem Programm namens Metafont zu Pixelzeichensätzen verwandelt, sodass sie in vernünftiger Zeit zu Papier gebracht werden können.

Allerdings gibt es nicht nur einen Vektorfont für alle Fontgrössen eines Zeichensatzes, sondern die standardmässig mitgelieferten Zeichensätze liegen in verschiedenen Entwurfsgrössen vor. Der Grund liegt darin, dass es schlichtweg keinen Algorithmus gibt, der mit Hilfe einer Vektorinformation für *jede* Auflösung den *perfekten* Zeichensatz generiert (Auch wenn es einige DTP-Hersteller einem weiswachen wollen!). Perfekt heisst in diesem Fall nicht mathematisch perfekt (denn das ist durchaus machbar), sondern für das menschliche Auge perfekt. Und das sieht die Proportionen der Strichdicken bei zwei Zeichengrössen eventuell verschieden, auch wenn sie mathematisch gleich sind. Wie man sieht, hat Knuth auch bei den Zeichensätzen versucht, eine möglichst perfekte Arbeit abzuliefern!

Zu Metafont gibt es leider (noch) keine umfassende deutsche Literatur, abgesehen von dem Kapitel in [4]. Man ist hier auf das englische Originalwerk von Knuth angewiesen, wenn man selbst eigene Zeichensätze generieren will. Von dem ist aber sowieso eher abzuraten, denn wer von uns kann schon behaupten, dass er eine Typographenausbildung hinter sich hat?

Das einzige, wozu man Metafont ab und zu braucht, ist die Generierung von Pixelzeichensätzen in nicht vorhandenen Grössen. Denn obwohl es für die gebräuchlichsten Zeichengrössen Vektordateien plus dazugehörige Pixeldarstellung gibt, möchte man vielleicht doch mal eine Grösse, die davon abweicht. Hierzu kann Metafont mit Vergrösserungsfaktoren den Zeichensatz (sogar recht schön) generieren. Bei einer Schrift zwischen 11 und 30 Punkten merkt man bei einer 10 Punkt Ausgangsschrift den Unterschied auch nicht (wer hat schon ein Typographenauge!). Kritisch bezüglich der Perfektion sind nur die ganz grossen und die kleinen Zeichensätze (die für Plakate und die für den Mikrofilm! :-)).

In einer guten Arbeitsumgebung werden die benötigten Zeichensätze aber sowieso automatisch generiert, sodass man sich nicht darum zu kümmern braucht.

So, das wars zur Einführung. Ich hoffe, dass das Interesse etwas geweckt wurde.

Martin Wunderli

Literatur

[1] Partl, Schlegl, Hyna. *LATEX2ε Kurzeinführung*, LATEX Quelldatei, DVI Datei und Postscript Datei in Deutsch (lkurz) und Englisch (lshort) frei erhältlich.

[2] Michel Goossens, Frank Mittelbach und Alexander Samarin. *Der LATEX–Begleiter*, Addison-Wesley, 1995.

[3] Helmut Kopka. *LATEX–Einführung (Band 1)*, Addison-Wesley.

[4] Helmut Kopka. *LATEX–Erweiterungen (Band 2)*, Addison-Wesley.

[5] Norbert Schwarz. *Einführung in TEX*, Addison-Wesley.

Shells

Die Bedienung von Unixrechnern[6] erfolgt auch heute in den meisten Fällen noch mittels Shells, egal ob diese nun auf der Console (im Textmodus) oder in einem xterm (unter X) laufen.

Besonders von Windowsbenutzern hört man immer wieder wie rückständig man sei, wenn man die Befehle noch via Tastatur und nicht mittels Maus gibt. Die Aussage dieser Benutzergruppe ist auch verständlich, da sich diese nur an die Standardshell von MSDOS[7] gewohnt sind. Was Windowsbenutzer aber nicht wissen: Shells unter Unix bieten *viel* mehr Komfort und Möglichkeiten als das was sie kennen. Aber auch unter Unix gibt es Unterschiede. Shells gibt es in rauhen Mengen, so das eigentlich jeder eine finden sollte, welche seinen Wünschen entspricht. Es gibt fast keinen Buchstaben der in Kombination mit `sh` nicht als Name einer Shell herhalten musste. Die wohl bekanntesten sind `sh`, `bash`[8], `csh`, `tcsh`, `ksh`, `ash` und `zsh`. Sollte keine der Shells den eigenen Wünschen entsprechen, lassen sich die meisten noch in weiten Bereichen konfigurieren.

Einige Features welche die Shells untereinander wie auch gegenüber der MS-DOS-Eingabeaufforderung unterscheiden:

- Job control — Jobs (Progamme) welche gestartet wurden können verwaltet werden (starten, stoppen, killen u.s.w.).
- Completion — Wird während der Eingabe (z.B. eines Kommandos) `TAB` gedrückt, stellt die Shell eine Liste von Möglichkeiten zur Verfügung. Dies sind zum Beispiel File- und Directorynamen, Usernamen, Hostnamen e.t.c. Einie Shells bieten auch die Möglichkeit diese Completion frei zu Programmieren. So kann man Konfigurieren was bei welchem Kommando an welcher Stelle in der Kommandozeile zu was vervollständigt wird.
- Command History — In den letzten Eingaben kann geblättert und diese auch verändert werden. Gewisse Shells beherrschen auch History-Completion. Damit wird z.B. die Eingabe von `!ema` und anschliessendem Tab zum letzten Emacs-Aufruf e.t.c. vervollständigt.
- Scriptsprache — Sämtliche Shells können die Befehle auch aus Files lesen und haben eingebaute Befehle und Kontrollmechanismen mit welchen sich auch kleinere Tools realisieren lassen. Diese Scriptsprache ist um einiges mächtiger als die Batchfiles unter MSDOS.
- Shellfunktionen — Wie in „normalen“ Programmiersprachen lassen

[6] darunter fällt natürlich auch Linux
[7] `C:>`
[8] Die Defaultshell unter Linux

sich bei manchen Shells in Scripts — und auch in der Shell selber — Funktionen definieren welche wie normale Befehle und Programme aufgerufen werden.

- Ein-/Ausgabenumleitung — Ausgaben von Programmen lassen sich in eine Datei oder an ein Programm (mit bestimmten Shells (zsh) auch in/an mehrere) umleiten. Das gleiche ist natürlich auch mit den Eingaben möglich. So werden unter Unix viele Dinge erledigt indem man die Daten durch mehrere Programme "durchpiped".
- Rechtschreibkorrektur — Die Shell erkennt gewisse Vertipper und macht Verbesserungsvorschläge. Gibt man aus Versehen `ecit` ein, bekommt man den Vorschlag `exit` und wird gefragt, ob dies gewünscht ist.

Die meiner Meinung nach interessanteste Shell ist die Z-Shell (zsh). Diese kann wohl als „eierlegende Wollmilchsau" unter den Shells bezeichnet werden. In ihr ist (fast) alles eingebaut was eine Shell bieten kann. Und ist in weiten Bereichen konfigurier- und erweiterbar.

Natuerlich sind auch Shells sehr vom Geschmack, den Ansprüchen und Bedürfnissen des Benutzers abhängig, so das jeder für sich entscheiden muss, welche er benutzen will. Mir persönlich gefällt die zsh bisher am besten.

Die Tabelle am Ende dieses Artikel zeigt die wichtigsten Unterschiede einiger Shells. Ein leerer Eintrag bedeutet das dies nicht möglich ist, ein ∘ bedeuted Optional (mit Erweiterungen) und ein • bedeutet vorhanden.

Wer mehr über die verschiedenen Shells wissen will, besorgt sich von Vorteil die Faq[9] *UNIX shell differences and how to change your shell* welche monatlich in der Newsgroup `news.answers` gepostet wird. Aus dieser stammt auch die Tabelle in diesem Artikel.

Quellen: Die üblichen Shells compiliert für Linux, befinden sich auf den ftp-Servern Sunsite und tsx-11 bzw. dessen Mirrors auf anderen FTP-Servern, sowie auf den üblichen Linux CD's.
Personen mit Internetzugang bemühen zum Auffinden der Shells am besten Archie.

Die zsh befindet sich als Binary in den neueren Slackware Distributionen, sowie auf tsx-11 und dessen Mirrors. Das Originalarchiv (Source) liegt auf `ftp.math.gatech.edu` im Directory `/pub/zsh`.

Roland Alder

[9] Frequently Asked Questions — Meistgestellte Fragen

	sh	csh	ksh	bash	tcsh	zsh
Job control		●	●	●	●	●
Aliases		●	●	●	●	●
Shell functions	○		●	●		●
„Sensible“ Input/Output redirection	●		●	●		●
Directory stack		●	●	●	●	●
Command history		●	●	●	●	●
Command line editing			●	●	●	●
Vi Command line editing			●	●	○	●
Emacs Command line editing			●	●	●	●
Rebindable Command line editing				●	●	●
User name look up		●	●	●	●	●
Login/Logout watching					●	●
Filename completion		○	●	●	●	●
Username completion		○	●	●	●	●
Hostname completion		○	●	●	●	●
History completion				●	●	●
Fully programmable Completion					●	●
Mh Mailbox completion				○	○	○
Co Processes			●			●
Builtin artithmetic evaluation		●	●	●	●	●
Can follow symbolic links			●	●	●	●
Periodic command execution					●	●
Custom Prompt (easily)			●	●	●	●
Sun Keyboard Hack				●		●
Spelling Correction					●	●
Process Substitution				○		●
Underlying Syntax	sh	csh	sh	sh	csh	sh
Freely Available			○	●	●	●
Checks Mailbox		●	●	●	●	●
Tty Sanity Checking					●	●
Can cope with large argument lists	●		●	●	●	●
Has non-interactive startup file		●	○	○	●	●
Has non-login startup file		●	○	●	●	●
Can avoid user startup files		●		●		●
Can specify startup file			●	●		
List Variables		●	●		●	●
Full signal trap handling	●		●	●		●
File no clobber ability		●	●	●	●	●

Apsfilter: Drucken von Postscript

Grundsätzlich gibt es nur zwei Möglichkeiten Postscript auszudrucken:

1. auf einen Postcriptdrucker (no problem!)
2. auf einen nomalen Drucker mit einem Postscriptinterpreter, der Postscript in die druckereigene Sprache umsetzt.

Unter Unix hat man die Möglichkeit, dem Druckerdämon diesen Postscriptinterpreter vor die Nase zu setzten, so dass sich der User nicht um diese Feinheit kümmern muss. Ja, die Filter sind so genial geschrieben, dass sie erkennen, ob eine Postscript-, Ascii- oder eine Bilddatei (GIF, TIFF, TGA (Targa), u.s.w.) ausgedruckt wird.

Genau dies macht das Programm `APSFILTER`! Mit Hilfe der in der Datei /etc/magic gespeicherten Fileformate, erkennt Apsfilter was die zu druckende Datei für ein Format hat. Mit dieser Information entscheidet sich dann der weitere Verlauf innerhalb des Filters:

- Erkennt er eine Postscript-Datei wird diese automatisch mit Ghostscript in die jeweilige Druckersprache umgeschrieben und ausgedruckt.
- Wird aber eine Ascii-Datei entdeckt, wird bei Apsfilter nicht wie man denken würde die Ascii Informationen direkt an den Drucker gegeben, sondern mit einem mitgelieferten Tool namens `a2ps` (ASCII to PS) in Postscript umgewandelt und dann wiederum mit Ghostscript ausgedruckt. Diese zusätzliche Umformatierung wird für spezielle Ausgebamglichkeiten, z.B. 2 to 1 oder mit Filename als Haeder, gebraucht. Es ist klar, das dies natürlich zusätzliche Rechnerleistung braucht, aber nach meinen Erfahrungen sind diese unter Linux kaum zu bemerken, falls die Datei nicht mehrere MB gross ist.

Wenn es besonders schnell gehen muss, hat man aber immer noch die Möglichkeit die Ascii Datei mit `lpr -Praw <filename>` auszudrucken. Dabei schickt der Druckerdaemon die Datei wie sie ist, ohne jegliche zusätzliche Formatierung, an den Drucker. Wie schon oben erwähnt erkennt das Programm Apsfilter noch etliche Bildformate, deren Ausdrucken jedoch zustzliche Tools erfordern. Diese sind im Paket Pbmplus enthalten und müssen zuerst kompiliert werden. Leider hatte ich mit Apsfilter auf diesem Sektor weniger Erfolg, d.h. Apsfilter hat sich geweigert lugs.gif auszudrucken (Pfui!).

So bin ich also bei den Problemen angelangt, die ich hatte:

1. Meinen Epson Stylus Color habe ich leider noch nicht zum Farbendruck

bewegen können. Ich vermute jedoch, dass dies nicht am Filter selber sondern eher am Ghostscript liegt, der die Farberweiterung von escp2 noch nicht kennt.

2. Ich brachte es nicht fertig, eine Postscriptdatei in der Mitte des A4 Blattes zu postieren.
3. Weiteren, wenn auch viel kleineren, Ärger bescherte mir die Tatsache, dass Apsfilter schlecht dokumentiert ist.

Fazit: Wer ein bisschen Spass hat beim Ausprobieren sollte sich unbedingt Apsfilter (mindestens Version 4.71) anschauen. Nich perfekt, aber nahe daran.

Philip Markwalder

Privtool — Das Mail Tool für Openwindows

Privtool steht für Privacy Tool und ist ein Email Frontend für Openwindow. Wer bis anhin mit viel Mühe seine Email mit pgp verschlüsselt hat, kann mit Privtool unter konfortabler Oberfläche per „Knopfdruck“ und Kennsatz seine Email verschlüsseln und/oder signieren. Dieses Tool wurde von Mark Grant (mark@unicorn.com) geschrieben und sollte eigentlich das von Sun’s Openwindow gewohnte Mailtool, erweitert mit pgp Fähikeiten, ersetzten. Obwohl für dieses Mailtool die Sourcen *frei* verfügbar sind, ist es mir bis jetzt nicht gelungen auch die neuste Beta Version 0.83 unter Linux zu kompilieren. Die Version 0.81 brachte man mit ein paar Tricks[10] zum kompilieren. Nach diesen anfänglichen Schwierigkeiten sind kleinere Defekte aufgefallen, mit denen man aber leben kann[11].

Mein Tip für sicherheitsbewusste Emailer, mit Privtool schickt ihr in Zukunft leichter eure Eurocard Nummer über’s Netz.

Philip Markwalder

[10] Auschalten einiger der SYSV Switches in der Source
[11] Vielleicht kommen diese Defekte von meinen Modifikationen :–)

UUCP Installation

Diese Anleitung soll dem Sysadmin eine Hilfe zur Installation von UUCP sein. Es werden die Compilierung, Installation und die Konfigurierung besprochen.

Copyleft by David Frey
`<david@eos.uu.ch>`

Bitte Fragen und Ergänzungen an obige Adresse.

Was ist UUCP überhaupt?

UUCP steht für **U**nix to **U**nix **C**o**P**y und erlaubt es auf einfachste Weise, sprich mit minimalem Aufwand, Unix-Rechnern untereinander zu vernetzen. Die Ursprünge liegen bei den AT&T Bell Laboratories, wo Mike Lesk 1976 ein erstes UUCP-System entwickelte [1].

Was leistet UUCP? Mit UUCP kann man/frau folgendes tun:

- Files von einem Unix-Rechner auf einen anderen transferieren (dies erledigt das Programm `uucp`, das dem ganzen Paket den Namen gab),
- ausgewählte Kommandos auf der Partnermaschine auszuführen,
- und — last but not least — Mail und News der Partnermaschine zusenden.

Was braucht man, um UUCP betreiben zu können?

Folgende Dinge sind nötig, um UUCP betreiben zu können:

1. Hardware
   (a) ein (Hayes kompatibles) Modem. Der Typ spielt keine Rolle; man/frau wählt natürlich das schnellste Modem, das das Budget (resp. die Normen[12] zulässt[13].
   (b) ein funktionfähiges Linux-System, das versteht sich aber von selbst.
2. Software

   Es gibt verschiedene UUCP-Packages. Am bekanntesten sind das BNU (Berkeley Network Utilities), auch bekannt unter dem Namen *HoneyDanBer* (manchmal auch mit *HDB* abgekürzt) (nach den Erfindern Peter *Honey*man, *D*avid *A*. *N*ovitz and *B*rian *E*. *R*edman) und mittlerweile auch das *Taylor*-UUCP von *Ian Lance Taylor*. Daneben existiert noch das alte *V2*-UUCP (das ich in der Einleitung erwähnte).

   Auf Linux-Maschinen wird defaultmässig das Taylor-UUCP mitgeliefert, da es

[12] die 28.8kbps-Modems (bit pro Sekunde, bitte nicht mit Baud verwechseln!!) sind gerade erst standardisiert und dementsprechend unsicher

[13] im Prinzip müsste das Modem nicht einmal Hayes kompatibel sein, der Besitzer muss jedoch nur wissen, wie man es anspricht…

- der **GPL** (GNU Public License) untersteht, und damit frei verfügbar ist, und, darüberhinaus,
- die meisten Möglichkeiten bietet.

Der springende Punkt ist nun der, dass man Taylor-UUCP in verschiedenen Modi betreiben kann, und dass es dementsprechend verschiedene Konfigurationsdateien verwendet:

(a) V2-UUCP-Emulation (obsolet)

(b) HDB-Emulation (weitverbreitet, dazu existiert am meisten Literatur)

(c) Taylor-Modus (am einfachsten einzurichten und am mächtigsten)

Ich werde im folgenden davon ausgehen, dass der Sysadmin das Taylor-UUCP im **Taylor**-Modus installieren will, da dies das modernste, leistungsfähigste und — meiner Ansicht nach — am Einfachsten zu konfigurierende UUCP-System ergibt. Will man Taylor-UUCP im HDB-Modus betreiben, so kann man auf schon bestehende Literatur zurückgreifen, insbesondere [1]. Aus obigen Satz zu schliessen, das dieses Dokumentchen die einzige Literatur zu Taylor-UUCP sei, ist allerdings schlichtweg falsch. Zum Taylor-UUCP wird eine sehr gute TEXInfo-Online-Dokumentation[14] mitgeliefert, die es zu lesen lohnt [2].

3. einen UUCP-Provider
Was nützt einem ein UUCP ohne Gegenstelle? man/frau muss also jemanden finden, der bereit ist, für einen die Services, die man wünscht, zu erbringen. So jemand wird im Jargon ein "Provider" genannt und ist häufig ein halb-kommerzieller Anbieter. Zu erwähnen wären da *SWITCH*, *Eunet*, *Relog AG* und andere. Es können allerdings auch engagierte Private sein, z.B. Daniel Etter.

Man könnte auch, wie zu den Anfangszeiten vom Internet, unter den LUGS-Mitgliedern die Maschinennamen, Telephonnummern, Anrufzeiten (= wann die Maschine läuft und das Modem eingeschaltet ist) austauschen und so ein Mininetz aufbauen.

Compilierung

Ich gehe davon aus, dass der Sysadm das Taylor-UUCP-Package im Source-Code vor sich hat und nun es nun compilieren will. Das Vorgehen ist — GNU üblich – einfach und ich skizziere es hier nur kurz.

Makefile editieren

Mit `./configure` ein `config.h` und das `Makefile` erzeugen. Beide anschauen, da man automatisch erzeugtem Code aus Erfahrung nicht trauen sollte (z.B. wollen die GNU-Utilities ständig eine weitere Version von `getopt` kompilieren und mit den Programmen linken, obwohl auf

[14] das Default Dokumentationsformat von GNU-Programmen

Linux-Systemen diese sowieso standardmässig in der (GNU)-C-Libraries eingebunden sind. Diese Idiotie stammt daher, dass sich die GNU-Utilities auch auf anderen, nicht so gut ausgestatteten Systemen kompilieren lassen müssen). Ich habe folgende Zeilen im `Makesfile` angepasst:

```
# Prefix directory for installation directories.
prefix = /usr

# The user name that should own the resulting executables, several of
# which are suid.
owner = uucp

# Where to install uucico, uuxqt, uuchk and uuconv.  If you use this
# definition, $(prefix)/lib must exist.
sbindir = $(prefix)/lib/uucp

# Where to install uucp, uux, uustat, uuname, uulog, uuto, uupick and
# cu.
bindir = $(prefix)/bin

# Where to install man pages.
# Section 1 for user programs, 8 for daemons.
man1dir = $(prefix)/man/man1
man1ext = .1
man8dir = $(prefix)/man/man8
man8ext = .8

# Where to install the info files.
infodir = $(prefix)/info

# The directory to look in for new style configuration files
# (when using HAVE_TAYLOR_CONFIG).  Note that by default this
# is different from sbindir, unlike traditional UUCP packages.
newconfigdir = /etc/uucp


...

# Define programs and flags
CC = gcc
CFLAGS = -O6 -Wall
LDFLAGS = LIBS
```

`policy.h` anpassen

Nun kommt der Punkt: die Datei `policy.h` ist nach den lokalen Gegebenheiten und/oder Politik des Systemadministrators zu editieren. Auf meinem System habe ich folgendes eingestellt:

```
#define HAVE_POSIX_TERMIOS 1
```

```
/* Set MAIL_PROGRAM to a program which can be used to send mail.  It
will be used for mail to both local and remote users.  Set
MAIL_PROGRAM_TO_BODY to 1 if the recipient should be specified as a
To: line in the body of the message; otherwise, the recipient will be
provided as an argument to MAIL_PROGRAM.  Set
MAIL_PROGRAM_SUBJECT_BODY if the subject should be specified as a
Subject: line in the body of the message; otherwise, the subject will
be provided using the -s option to MAIL_PROGRAM (if your mail program
does not support the -s option, you must set MAIL_PROGRAM_SUBJECT_BODY
to 1).  If your system uses sendmail, use the sendmail choice below.
Otherwise, select one of the other choices as appropriate.  */
#if 1
#define MAIL_PROGRAM "/usr/lib/sendmail -t"
#define MAIL_PROGRAM_TO_BODY 1
#define MAIL_PROGRAM_SUBJECT_BODY 1
#endif
```

Nachtrag: Beim mir — und bei ca. 95% der anderen Installationen — ist `sendmail` ein link auf `smail`.

```
#if 1
#define PS_PROGRAM "/bin/ps -lp"
#define HAVE_PS_MULTIPLE 0
#endif
```

Ich selbst benutze Branko Leicester's ps, das auf dem /proc-Filesystem aufsetzt und BSD-Kommandozeilenparameter hat (`ps -aux`). Sollte irgendjemand ein anderes `ps` benutzen, was ich aber bezweifle, so wäre dies anzupassen.

```
#define LOCKDIR "/var/lock/uucp"
```

Die Pfade, die ich hier - und nachfolgend - angegeben habe, sind FSSTND (Linux Filesystem Standard) konform[15].

```
/* You must also specify the format of the lock files by setting
   exactly one of the following macros to 1.  Check an existing lock
   file to decide which of these choices is more appropriate.

   The HDB style is to write the locking process ID in ASCII, passed
   to ten characters, followed by a newline.

   The V2 style is to write the locking process ID as four binary
   bytes in the host byte order.  Many BSD derived systems use this
   type of lock file, including the NeXT.

   SCO lock files are similar to HDB lock files, but always lock the
   lowercase version of the tty (i.e., LCK..tty2a is created if you
   are locking tty2A).  They are appropriate if you are using Taylor
   UUCP on an SCO Unix, SCO Xenix, or SCO Open Desktop system.

   SVR4 lock files are also similar to HDB lock files, but they use a
```

[15] ok, das Subdirectory *uucp* ist meine Erfindung, aber im */var/lock*-Directory sollten die Lock-Files schon liegen.

```
   different naming convention.  The filenames are LK.xxx.yyy.zzz,
   where xxx is the major device number of the device holding the
   special device file, yyy is the major device number of the port
   device itself, and zzz is the minor device number of the port
   device.

   Sequent DYNIX/ptx (but perhaps not Dynix 3.x) uses yet another
   naming convention.  The lock file for /dev/ttyXA/XAAP is named
   LCK..ttyXAAP.

   Coherent use a completely different method of terminal locking.
   See unix/cohtty for details.  For locks other than for terminals,
   HDB type lock files are used.

   QNX lock files are similar to HDB lock files except that the node
   ID must be stored in addition to the process ID and for serial
   devices the node ID must be included in the lock file name.  QNX
   boxes are generally used in bunches, and all of them behave like
   one big machine to some extent.  Thus, processes on different
   machines will be sharing the files in the spool directory.  To
   detect if a process has died and a lock is thus stale, you need the
   node ID of the process as well as the process ID.  The process ID
   is stored as a number written using ASCII digits padded to 10
   characters, followed by a space, followed by the node ID written
   using ASCII digits padded to 10 characters, followed by a newline.
   The format for QNX lock files was made up just for Taylor UUCP.
   QNX doesn't come with a version of UUCP.  */
#define HAVE_V2_LOCKFILES 0
#define HAVE_HDB_LOCKFILES 1
```

Über Format und Art der Lock-Files wurde schon viel diskutiert. Der "normale" Weg ist, dass die Process-ID in ASCII in das Lock-File geschrieben werden. Sollten danach andere Programme (in Frage kommen da: kermit, minicom und seyon, insbesondere kermit) die Lock-Files nicht verstehen, so sind *diese* zu rekompilieren, da dies klar unüblich ist. Getties verlassen sich darauf, die PIDs in ASCII lesen zu können[16].

```
/* To compile in use of the new style of configuration files described
   in the documentation, set HAVE_TAYLOR_CONFIG to 1.  */
#define HAVE_TAYLOR_CONFIG 1

/* To compile in use of V2 style configuration files (L.sys, L-devices
   and so on), set HAVE_V2_CONFIG to 1.  To compile in use of HDB
   style configuration files (Systems, Devices and so on) set
   HAVE_HDB_CONFIG to 1.  The files will be looked up in the
   oldconfigdir directory as defined in the Makefile.

   You may set any or all of HAVE_TAYLOR_CONFIG, HAVE_V2_CONFIG and
   HAVE_HDB_CONFIG to 1 (you must set at least one of the macros).
```

[16] mgetty lässt sich allerdings bei der Compilation konfigurieren.

```
   When looking something up (a system, a port, etc.) the new style
   configuration files will be read first, followed by the V2
   configuration files, followed by the HDB configuration files.  */
#define HAVE_V2_CONFIG 0
#define HAVE_HDB_CONFIG 0

#define SPOOLDIR_V2 0
#define SPOOLDIR_BSD42 0
#define SPOOLDIR_BSD43 0
#define SPOOLDIR_HDB 0
#define SPOOLDIR_ULTRIX 0
#define SPOOLDIR_SVR4 0
#define SPOOLDIR_TAYLOR 1

#define USE_TRADITIONAL_STATUS (SPOOLDIR_HDB || SPOOLDIR_SVR4)

#define HAVE_TAYLOR_LOGGING 1
#define HAVE_V2_LOGGING 0
#define HAVE_HDB_LOGGING 0
```

Wir wollen alles im Taylor-Format haben.

```
/* The name of the default spool directory.  If HAVE_TAYLOR_CONFIG is
   set to 1, this may be overridden by the ``spool'' command in the
   configuration file.  */
/* #define SPOOLDIR "/usr/spool/uucp" */
#define SPOOLDIR "/var/spool/uucp"

/* The name of the default public directory.  If HAVE_TAYLOR_CONFIG is
   set to 1, this may be overridden by the ``pubdir'' command in the
   configuration file.  Also, a particular system may be given a
   specific public directory by using the ``pubdir'' command in the
   system file.  */
/* #define PUBDIR "/usr/spool/uucppublic" */
#define PUBDIR "/var/spool/uucppublic"

/* The default command path.  This is a space separated list of
   directories.  Remote command executions requested by uux are looked
   up using this path.  If you are using HAVE_TAYLOR_CONFIG, the
   command path may be overridden for a particular system.  For most
   systems, you should just make sure that the programs rmail and
   rnews can be found using this path.  */
#define CMDPATH "/bin /usr/bin /usr/local/bin"
```

Den Pfad könnte man auch auf `/bin /usr/bin` einstellen, je nach Sicherheitsbedürfnis.

Dieser Default hängt ganz from verfügbaren Plattenplatz ab.

```
#define DEFAULT_FREE_SPACE (50000)
```

Compilation und Installation der Binaries

Mit `make` Taylor-UUCP generieren und mit `make install` installieren. Vorher kann man noch `uuchk` - wie im README empfohlen - laufen lassen, falls man/frau schon vorher eine UUCP-Installation hatte.

Installation

`make install` installiert Taylor-UUCP in den oben konfigurierten Pfaden mit den “richtigen” Permsissions.

Permissions

Auf meiner Maschine sehen die Permissions folgendermassen aus:

`/usr/lib/uucp`-Directory:

```
total 309
dr-xr-xr-x   2 uucp     uucp         1024 Dec 18 22:56 ./
drwxr-xr-x  15 bin      users        3072 Jan 22 23:04 ../
-r-x--x--x   1 uucp     uucp        50180 Nov 20 01:40 uuchk*
-r-s--x--x   1 uucp     uucp       177156 Nov 20 01:40 uucico*
-r-x------   1 uucp     uucp          341 Dec 18 22:53 uuclean.daily*
-r-x------   1 uucp     uucp          569 Dec 18 22:56 uuclean.monthly*
-r-x------   1 uucp     uucp          794 Dec 18 22:54 uuclean.weekly*
-r-x--x--x   1 uucp     uucp          322 Nov 22 20:25 uusched*
-r-s--x--x   1 uucp     uucp        74756 Nov 20 01:40 uuxqt*
```

Die `uuclean.*`-Files sind Skripts die täglich/wöchentlich/monatlich in den `/var/spool/uucp/*`-Directories aufräumen.
`/etc/uucp`-Directory:

```
total 7
drwx------   2 uucp     uucp         1024 Jan  5 18:57 ./
drwxr-xr-x   8 root     root         2048 Jan 26 23:01 ../
-rw-------   1 uucp     uucp           56 Dec 21 23:17 call
-rw-------   1 uucp     uucp          194 Dec 19 21:27 dial
-rw-------   1 uucp     uucp          154 Dec 21 22:58 port
-rw-------   1 uucp     uucp          466 Dec 29 21:02 sys
```

Zu diesen Files, sage ich weiter unten, im Kapitel Konfiguration, noch mehr. Zu den Permissions ist folgendes zu sagen:

Da die Datei `call` *sensitive* Informationen, nämlich die Namen der/des Nachbarrechne(r) und die Passwörter zu deren UUCP-Accounts, ist (mindestens) gegen unbefugtes Lesen zu schützen! Die Datei `sys` enthält zudem die Telephonnummern der neighbors; man kann sich also überlegen, dieser nicht auch gleich Group/Other-Leseberechtigung zu entziehen.

Konfiguration

Ich beschreibe nachfolgend die vorgeschlagene Standard-Konfigurationen, es ist jedem freigestellt z.B. alles in ein File zu packen. Allerdings sollte man/frau vorher die info-Pages gelesen haben! Diese Namen lehnen sich vom Inhalt her an die BNU-Benennung an.

`call`-File

Im call File stehen die Namen der Nachbarrechner, die Namen der Login-Accounts deren dazugehörige Passwörter hierzu. Das Format dieses Files sieht folgendermassen aus:

pname	llogin	lpasswd
…	…	…

`pname` steht für den Hostnamen des Providers, `llogin` für den Loginnamen, den der lokale Rechner beim Provider benutzen soll und `lpasswd` ist das dazugehörige Password. Beispiel: mein Rechner heisst *apple* und der dazugehörige Provider heisst *banana*. Es hat sich eingebürgert, den Login-Namen für UUCPmit `u`*`lname`* zu verwenden, in unserem Beispiel also `uapple`. Als Password wählte der Sysadmin von apple *green*. Somit sieht die vollständige `call`-Zeile folgendermassen aus:

banana	uapple	green

`banana` hat in seinem `/etc/passwd`-File den folgenden Eintrag:

```
uapple:Password:500:14:Apple,,,:/var/spool/uucp:/usr/lib/uucico
```

500 ist die Usernummer und 14 die Group ID.

`dial`-File

Im Dial-File steht was für einen Dialer[17] man verwendet. In ihm stehen, was man als Übertragungsmedium brauchen will, also z.B. Modem, oder Direkt. Es hindert aber niemanden, hier auch TCP/IP einzutragen (ob dies sinnvoll ist, steht auf einem anderen Blatt Papier). Verwendet man ein Hayes-kompatibles Modem und den Dialer anzurufen, so steht in etwa folgendes drin:

```
# dial file for Taylor UUCP 1.05
dialer hayes
carrier true
dtr-toggle true
chat "" \dAT\r OK\r \EATDT\D\r CONNECT
fail BUSY
fail NO\sCARRIER
complete \d\d+++\d\dATH\r\c
abort \d\d+++\d\dATH\r\c
```

[17] Der Begriff Dialer ist ein historisches Relikt als noch die Wähleinrichtung von der Codier-/Decodierschaltung getrennt waren. Heutzutage würde man keinen Unterschied mehr machen.

Man sieht unter anderem, dass ein Hayes-kompatibles Modem angehängt ist (`dialer hayes`) und dass auf einen carrier gewartet wird. Mit `dtr-toggle true` wird die DTR-Leitung vor dem Gebrauch getoggelt, was üblicherweise (zumindest unter Unix) das Modem zurücksetzt. Die *Chat-Sequenz* beschreibt in der Form *expected – send* was im fehlerfreien Fall ans Modem gesendet resp. vom Modem erwartet wird. Übersetzt heisst das: Erwarte nichts, sondern sende ein `AT`. Als Antwort erscheint ein `OK` (das Modem ist richtig angeschlossen) und man kann die Nummer wählen (`ATDT` *`telephon-nr`*). Ist der Provider frei, und erhält man einen Carrier, so meldet das Modem `CONNECT`. Im Gegensatz zu anderen UUCPs erkennt Taylor-UUCP, falls die Leitung besetzt ist, oder kein Carrier vorhanden ist. BNU UUCP z.B. wartet unintelligenterweise auf einen Timeout.

`port`-File

Im `port` stehen welche Geräte an welchen Ports angeschlossen sind und man kann da auch die Protokollparameter vorgeben (die pro Port unterschiedlich sein können).

```
# Port file for Taylor UUCP 1.05
port ACU
type modem
device modem
speed 38400
dialer hayes

protocol-parameter g timeout 5
protocol-parameter g retries 3

port DIRECT
type direct
device modem
speed 38400

protocol g
protocol-parameter g timeout 5
protocol-parameter g retries 3
```

Einerseits stehen da die üblichen Parameter (Bitrate, welchen Dialer man verwenden möchte, und – nicht zu vergessen – welches Device). Die Namen ACU und DIRECT wurden vom Benutzer vergeben (ACU steht übrigens für Automatic Calling Unit [auch ein AT&T-Relikt]). Die Typen "`direct`" und "`modem`" sind übrigens im Taylor-UUCPeingebaut.

`sys`-File

Im `sys`-File kommen die oben beschriebenen Files – anscheinend wild verstreut – zusammen. Die Files wurden extra so aufgeteilt, um das Anfügen von weiteren Providern zu erleichtern. Im Prinzip könnte man alle Einträge im `sys`-File "hartverdrahten"[18] und die anderen Files vergessen. Das `sys`-File sieht folgendermassen aus:

```
# sys file for Taylor UUCP 1.05
#
# Global

call-login *
```

[18] Sorry für die Beleidigung, Hardware-Entwickler!

```
call-password *

remote-send /var/spool/uucppublic
local-receive /
remote-receive /var/spool/uucppublic

# for apple
system apple
phone 123456789
time Wk1705-1855,Wk2105-0755,SaSu0000-2359

chat "" \r\r ogin: \L ssword: \P

# for pear
system pear
phone 987654321
time Wk1705-1855,Wk2105-0755,SaSu0000-2359

chat "" \r\r ogin: \L ssword: \P

# for plum
system plum
port DIRECT
time ANY

chat "" \r\r ogin: \L ssword: \P
```

Die Parameter sind ziemlich selbsterklärend. `\L` resp. `\P` stehen für den Login resp. das Passwort die aus dem Call-File extrahiert wurden. Dies wird durch die Angabe von * im `call-login` und `call-password` bewerkstelligt. Durch setzen von Zeitlimiten setzen, kann man zum Beispiel Niedertarifzeiten erzwingen.[19]

Modem-Konfiguration

Die Modem-Konfiguration ist auch bei Hayes-kompatiblen Modems unterschiedlich, man sollte also das Modemhandbuch griffbereit haben. Genaugenommen sind drei verschiedene Konfigurationden zu unterscheiden, die aber miteinander verzahnt sind:

1. Hardware-Konfiguration des Modems, die *EEPROM* liegt.
2. Konfiguration des `getty`-Programms, `getty` überschreibt evtl. Einstellungen, die im EEPROM liegen.
3. *Chat-Skript*. Es ist möglich, aber ungebräuchlich, dass auch im Chat-Skript noch Einstellungen neu eingestellt werden könnten.

EEPROM-Konfiguration im Modem selber

Dies ist die wichtigste Konfiguration, wenn diese nicht stimmt, wird kein Datentransfer funktionieren. Folgendes ist

[19] Man beachte, dass die Niedertarifzeiten um 5 Minuten eingegrenzt wurden, um allfällige Ungenauigkeiten der Systemuhr zu kompensieren!

zu beachten:
Die plain-vanilla MS-LOSS Konfiguration ist auf UNIX-Systemen unbrauchbar.

Wieso? Viele voreingestellte Konfigurationen gehen davon aus, dass das Terminalprogramm die Signallleitungen DTR[20],DSR[21] und CD[22] nicht beachtet werden und setzt sie permanent high. Zuallerletzt ist praktisch immer auch der Autoanswermode eingeschaltet, so dass das Modem auch abnehmen wird, wenn der Computer ausgeschaltet ist! Meine Konfiguration beispielweise sieht folgendermassen aus (ein `FM-144Vbis+/R1 V3.1p`-Modem):

Befehl	**Beschreibung**	**Bemerkungen**
`B0`	variable link rate CCITT compability	*
`E1`	commands are echoed	
`L1`	low speaker volume	Geschmackssache
`M0`	speaker off	Geschmackssache
`Q0`	send response codes (OK, ERROR…)	**wichtig!** *
`V1`	responses in textual form	
`X4`	both dialtone and busytone detection	
`Y0`	Long space disabled	Geschmackssache *
`&C1`	DCD Signal follows modem CD signal	**wichtig!** *
`&D3`	Reset when DTR high-to-low transition	**wichtig!**
`&G0`	guard tone disabled	*
`&L0`	dialup operation	*
`&M0`	asynchronous only	*
`&O0`	perform like a normal modem	*
`&R0`	CTS follows RTS	**wichtig!**
`&S1`	DSR follows RS-232C specification	**wichtig!***
`&X0`	synchronous clock provided by modem	egal
`&Y0`	profile 0 will be retrieved when power on	Geschackssache
`%D1`	send Clear-Down enabled	*
`%L011`	set transmit level -11 dBm	*
`%A000`	set auto-reliable character to 0x0	*
`%C1`	enable data compression (MNP 5/V42bis)	**wichtig!**

[20] DTR=Data Terminal Ready
[21] DSR=Data Set Ready
[22] CD=Carrier Detected

Befehl	Beschreibung	Bemerkungen
\A3	256 characters max. MNP block size	tune-bar
\C0	do not buffer data (MNP, auto-reliable mode)	*
\E0	do not echo data during a normal link	
\G0	disable DCE flow control	MNP ignoriert \G
\H0	HP ENQ/ACK protocol disabled	
\J0	turn off DTE baud rate adjustment	fixe Bitrate beim PC wichtig für MNP
\K5	queued	*
\N6	auto-reliable V.42 or MNP data link	Fallback andernfalls
\Q3	RTS/CTS flow control	**wichtig!** das einzige Protokoll
\T000	inactivity timer	**wichtig!** Verbindungen sind manuell zu trennen.
\V1	send extended responses	
\X1	Process XON/XOFF pass through	Einstellung egal, da XON/XOFF in den `getty`-Parametern sowieso ausgeschaltet (*protokollabhängig*)!

Register	Beschreibung	Bemerkungen
S00:000	auto answer, # of rings	**wichtig! auf 0** setzen (abschalten), da `getty` abnehmen soll.
S07:045	waiting period for carrier	$45s$
S08:002	delay for “,”	$2s$
S09:006	carrier must stable for this delay, until DCD will be signalled	$0.6s$
S10:014	max. time span for carrier interruption	$1.4s$
S11:095	length of a digit when (tone dialling)	$0.95s$
S12:050	min. pause after an escape character	$1s$

Man sieht also, dass die Default-Einstellungen (mit Stern markiert) bei diesem Modem recht gut eingestellt wurden (obwohl ich das Modem bei Vobis gekauft hatte). Ich bin mir bewusst, das andere Modems unter Umständen ganz andere Kommandi benutzen, um die oben dargestellten Werte bekanntzugeben (Zyxel, Telebit u.a.), aber als Startpunkt für Modemsitzer ohne Handbuch sollte obige Tabelle sicherlich ausreichen.

mgetty-Konfiguration

Mit obiger Konfiguration ist man soweit, dass man mit einem ATZ, d.h. einem Modem-Reset, bereits eine vernünftige Standardeinstellung erhält. Man muss nun seinem benutzten getty (ich empfehle hier übrigens entweder ps_getty 2.0.7e mit Security-Patch[23] oder mgetty, auch mit Security-Patch, das überdies auch noch faxen kann.) noch die Portparameter klarmachen. Das sieht beim ps_uugetty folgendermassen aus:

```
# /etc/gettydefs
#
# This file contains the startup and final flags for the
# tty lines.  Each line starts with a SPEED value; this is
# the same SPEED that you pass to [uu]getty.  Note that it
# does not have to correspond to any real speed.
#
# the flags are the same flags you would pass to the stty program.
#
# Format: <speed># <init flags> # <final flags> #\
<login string>#<next-speed>

# Virtual Console entry
VC# B38400 SANE CLOCAL # B38400 SANE -ISTRIP CLOCAL #@S login: #VC

# 14400 baud Modem/direct line entries
38400# B38400 CS8 # B38400 SANE -ISTRIP HUPCL CRTSCTS -CLOCAL #\
@S login: #19200

19200# B19200 CS8 # B19200 SANE -ISTRIP HUPCL CRTSCTS -CLOCAL #\
@S login: # 9600

 9600# B9600  CS8 # B960   SANE -ISTRIP HUPCL CRTSCTS -CLOCAL #\
@S login: # 9600
```

Wichtig ist bei obiger Konfiguration CS8, -CLOCAL und CRTSCTS. Dies entspricht der bekannten 8N1-Einstellung mit dem Zusatz “dies ist ein Modem und nicht ein lokales Terminal”. (Bem: alle 38400..., 19200..., ...-Zeilen sind auf einer Zeile zu schreiben, ohne \.)

getty selbst wird im inittab mit folgenden Zeilen gestartet:

```
S1:3:respawn:/sbin/mgetty -n 5 modem 38400
s1:4:respawn:/sbin/uugetty -h -r0 modem 38400
```

[23] –froot-“Loginnamen” sollten abgewiesen werden

Taylor-UUCP-Chat-Skript

Das Chat-Skript ist oben abgedruckt und ist je nach Provider etwas anders. Es entspricht einer *send – expect*-Sequenz. Im Normalfall kommt man mit einer

```
"" \dAT\r OK\r \EATDT\D\r CONNECT
```

und einer

```
"" \r\r ogin: \L ssword: \P
```

Sequenz aus. Dies bedeutet:

`""`		erwarte gar nichts
	`\dAT\r`	warte (`\d`) sende ein AT und einen Carriage Return, um zu schauen, ob ein Modem angeschlossen ist.
`OK\r`		Modem meldet mit "OK"
	`\EATDT\D\r`	Wähle die Nummer, aus dem `sys`-File, entnommen, im Dialtone-Verfahren.
CONNECT		Wir sind verbunden!

Hier folgt nun der Provider spezifische Teil. Ich nehme an, dass ein Unix-Rechner direkt am Modem hängt und nicht irgendwelche spezielle Router-Befehle einzugeben sind.

`""`		erwarte gar nichts
	`\r\r`	sende danach zwei Carriage Returns
ogin:		darauf erscheint der `login:`-Prompt.
	`\L`	sende das im `call`-File vereinbarte Login
ssword:		darauf erscheint der `Password:`-Prompt.
	`\P`	sende das im `call`-File vereinbarte Password.

Literatur

[1] Grace Todino Tim O'Reilly. *Managing UUCP and Usenet*. O'Reilly & Associates, Inc., 103 Morris Street, Suite A Sebastopol, CA 95472, tenth edition, January 1992.

[2] Ian Lance Taylor. *TeXInfo Documentation Taylor UUCP 1.05*. 675 Mass Ave, Cambridge, MA 02139, USA. /usr/info/uucp.info-[1-4].

David Frey

Anmerkung:
Aus Gründen des besseren Satzes habe ich einige Kommentare in den abgedruckten Files entfernt. Diese können aus den beim UUCP beiliegenden Files entnommen werden. (RA)

Adressen

Präsident:
Ralph Steigrad
Frohbühlstrasse 16
8052 Zürich
Tel. P: 01/303 06 41
Tel. G: 01/257 52 38
Email: steigrad@geo.umnw.ethz.ch

Vizepräsident:
Philip Markwalder
Summerauweg 1
8623 Kempten
Tel. P: 01/930 30 38
Email: pgmarkwa@stud.ee.ethz.ch

Redaktor:
Erich Stüssi
Oberdorfstrasse 3B
5213 Villnachern
Tel. P: 056/42 21 08
Tel. G: 01/844 19 19
Email: est@etter.uu.ch

Kassier:
Daniel Etter
Ringstrasse 2
8107 Buchs ZH
Tel. P: 01/844 58 25
Email: daniel@etter.uu.ch

Aktuar:
Paul Bosshard
Hulmenweg
8352 Ricketwil
Tel. P: 052/252 50 63
Email: paul.bosshard@squirrel.ch

Vereinslokal:
ETH Zürich
HG F33.5
ETH Zentrum
8092 Zürich

Vereinsadresse:
Linux User Group Switzerland
c/o Daniel Etter
Ringstrasse 2
8107 Buchs ZH

Bankverbindung:
Postcheque-Konto
80-54319-6

Mitgliederbeitrag: Nicht Verdienende: Fr 60.-, Juristische Personen: Fr 200.-, Andere: Fr 100.-

Impressum

Auflage:
200 Exemplare

Herausgeber:
Linux User Group
Switzerland
c/o Daniel Etter
Ringstrasse 2
8107 Buchs ZH

Redaktion:
Erich Stüssi

Druck:
Universitätsdruckerei
Uni Irchel, Zürich

Inseratenpreise:

1/1 Heftrückseite	Fr.	100.-
1/1 Innenseite	Fr.	75.-
1/2 Innenseite	Fr.	50.-

Artikel:
Artikel und Inserate müssen bis zum Redaktionsschluss dem Redaktor vorliegen. Fragen zu Format usw. werden gerne beantwortet.

Erscheinungsweise:
4 mal jährlich

Inhaltsverzeichnis

Nächster Redaktionsschluss: 16.6.1995