Einfuehrung in UNIX (LinuX) fuer DOS-Umsteiger - Version 1.21 - 03.03.1998

Erstellt von DG2MIC (Hartwig Harm, Riesengebirgstr. 9, D-85540 Haar)

An dieser Gegenueberstellung von DOS- und UNIX-(LinuX)-Befehlen haben mitgearbeitet: DG2MNN, DL3MCW, DJ0ZY und viele andere. An dieser Stelle meinen herzlichen Dank an alle, auch die, die nicht extra genannt wurden!

Inhalt

1. Was ist anders bei Unix gegenueber DOS?

Unix ist ein Multitasking-Betriebssystem und kann mehrer Prozesse (so heissen die Programme bei Unix) quasi gleichzeitig ausfuehren. Hierzu laufen nach dem Start des Systems staendig einige Prozesse auf 'Sparflamme' (fuer den User unsichtbar), die bei Bedarf 'anspringen' und die erforderlichen Massnahmen treffen.

Ein solcher Prozess tritt auch in Aktion, wenn sich ein neuer User am System meldet. Er teilt dem User einen neuen Prozess zu (meist eine Bourne Shell), von dem aus der User seine Aktionen starten kann. Unter 'Aktionen' sind die Ausfuehrung von Befehlen und Programmen zu verstehen. Jeder User hat - anders als bei DOS - die Moeglichkeit, mehrere Programme zu starten, von denen alle bis auf einen (den sogenannten Vordergrund-Prozess) im Hintergrund ablaufen. Die Hintergrund-Prozesse bekommen eine Nummer, unter der sie vom System verwaltet werden und mit der sie auch wieder in den Vordergrund geholt und/oder abgebrochen werden koennen.

Der Anfaenger wird jedoch - wie von DOS gewohnt - zunaechst immer nur einen Prozess zur Zeit starten und ausfuehren.

Jeder User kann aber mit anderen Usern im System kommunizieren, sobald er die entsprechenden (im Hintergrund wartenden) Prozesse aktiviert. Hierzu gehoeren neben dem E-Mail-Versand (mit mail oder mailx) die Prozesse write, talk (nicht fuer AX.25-User), convers, lconvers und Mitteilungen an alle User durch den Befehl: echo <text> | wall

Filenamen werden bei Unix ueblicherweise mit kleinen Buchstaben geschrieben, Variablen dagegen mit Grossbuchstaben. Eine Zeichenfolge aus Grossbuchstaben, die mit einem $-Zeichen beginnt, wird als Umgebungsvariable betrachtet.

Variablennamen koennen (fast) beliebig lang sein. Sie duerfen - anders als bei DOS - den Punkt (.) an beliebiger Stelle enthalten. Extentions (wie z.B. .EXE

oder .BAT bei DOS) gibt es bei Unix nicht und man kann daher einem File-Namen nicht ansehen, ob es sich z.B. um ein ausfuehrbares Programm oder ein Text-File handelt. Ueber die Art eines Files kann man mit dem Befehl find <filename> eine meist zutreffende Aussage erhalten (es ist eine 'guess'-Aussage!). Ausfuehrbare Files muessen aber in den Zugriffs-Rechten das Execute-Bit gesetzt haben.

Der Befehl 'ls -al' listet z.B. alle Files des aktuellen Directories auf. Dabei sind ausser dem Owner, der Filelaenge und dem Erstellzeitpunkt am linken Rand die Zugriffsrechte aufgefuehrt. Ein '-' bedeutet, dass das Recht nicht eingeraeumt ist, die Buchstaben d,r,w und x bezeichnen ein Directory, das Nur-Leserecht (read), das Schreib- und Loeschrecht (write) und x kennzeichnet das Executerecht. Das erste Zeichen der Zugriffsrechte traegt das Directory-Kennzeichen. Die anschliessenden 3 Gruppen mit je 3 Stellen gelten (von links nach rechts) fuer den 'user', die 'group' (Nutzergruppe, der der User angehoert) und 'other', das sind alle anderen Nutzer im System. Zur Aenderung der Zugriffsrechte und 'owner' siehe die Befehle chmod, chown und chgrp.

2. Gegenueberstellung von DOS- und Unix-Befehlen

In der DOS-Spalte sind auch Befehle aufgefuehrt, die nur mit entsprechenden DOS-Erweiterungen oder nur in hoeheren DOS-Versionen verfuegbar sind.
Aus Platzgruenden (und um die Uebersichtlichkeit nicht zu beeintraechtigen) sind bei den meisten Befehlendie notwendigen Parameter NICHT aufgefuehrt.
Bitte ggf. mit dem HELP-Befehl (DOS) oder in den man-pages ('man'-Befehl unter Unix) nachschlagen.

Konventionen und Zeichen mit spezieller Bedeutung

DOS	Unix	Erklaerung/Bemerkungen
\	/	Trennzeichen in Pfadanweisungen
xyz=XYZ	xyz<>XYZ	Unix unterscheidet Gross/Kleinschreibg!
.	.	gegenwaertiges Directory
..	..	naechst hoeheres (Parent-)Directory
* und ?	* und ?	Platzhalter in Pfad- und File-Specs
|	|	Pipe-Zeichen in Befehls-Ketten
ctrl-c	ctrl-d und/oder ctrl-c	haeufigster Abbruch-Tastendruck
/	-	Standardzeichen fuer Switches/Optionen
""	"" und/oder ''	Zeichen, um Strings zu kennzeichnen
>,>>,<	>,>>,<	Umleitungszeichen
*.BAT	keine Festlegungen	ausfuehrbare Shell- bzw. Script-Datei
*.COM/EXE/DLL	keine Einschraenkungen	Dateinamen-Kennung ausf□hrbarer Dateien

Vergleichbare Befehle

DOS	Unix	Erklaerung/Bemerkungen
DIR	ls -l oder -al	-al zeigt auch verdeckte Dateien an
DIR /S	ls -l *	Directory incl. Sub-Directories
TYPE	cat <file> [|more]	Anzeigen des File-Inhaltes (seitenweise)
COPY	cp -iv <...> <...>	copy - Der Schalter -iv wird dringend empfohlen. Er steht fuer 'interakive' Ausfuehrung mit 'verbose'(=ausfuehrlichster) Hilfestellung bei Fehlern
RENAME	mv <...> <...>	mv (=move) kann in Unix noch mehr, s.u.

DEL, ERASE	rm -iv <file(s)>	remove:Schalter -iv unbedingt empfohlen!
MD, MKDIR	mkdir [-p] <name>	-p = inclusive ev. parent directories
RD, RMDIR	rmdir <name>	remove directory - Directory loeschen
CD	cd	(einer der wenigen identischen Befehle)
DATE	(time)	Datum anzeigen (bei DOS auch aendern)
TIME	time	Zeit (Unix: auch Datum) anz./(aendern)
SHELL	sh	Aufruf der (Bourne-)Shell
>	$	Prompt-Zeichen in der DOS/Bourne-Shell
HELP <cmd>	man <cmd>	ausfuehrliche Hilfe zu einem Befehl
<cmd> /h (/?)	<cmd> --help	kurze Hilfe zu Befehl (geht oft)
Rechner aus	exit (z.T.auch 'logout')	Beenden einer DOS- bzw. Unix-Sitzung
TOUCH	touch <file>	Datum/Zeit eines Files aktualisieren. In Unix auch Erzeugen eines leeren Files
DIR NUL	pwd	Anzeige des derzeit aktiven Pfades
PROMPT=pg	???	Anzeige des Pfades im Prompt
ATTRIB	chmod	Zugriffsrechte aendern. Beispiele:
	chmod a+r <file>	Alle erhalten/behalten Leserecht
	chmod u+x <file>	User erhaelt Executerecht
	chmod g-w <file>	Group verliert Schreibrecht
FREE	df -k	disk free - Freier Speicherplatz auf der/den HDD in 1024-byte-Bloecken
SET PATH	echo $PATH	Pfad-Umgebungsvariable anzeigen
SET	printenv	Alle Umgebungsvariablen anzeigen
WHERE <file>	find / -name <file> -print	Ermitteln wo sich 'file' befindet
PRINT <file>	cat <file> > /dev/printer	File auf Default-Printer ausdrucken
EDLIN <file>	ed [-p#] <file>	Zeileneditor [mit # als Promptzeichen]
EDIT	vi	Bildschirmorientierten Editor starten

3. Unix-Befehle, die es in DOS nicht gibt...

pwd	present working directory - wo bin ich (im Dateibaum)
whoami	wer bin ich (zeigt nur Einlogg-Namen, bzw. im Amateurfunk das Call)
who am i	wer bin ich (zeigt: Host!Call Interface Datum Zeit Protokoll)
who	wer ist eingeloggt an meinem Host (Format aehnlich wie 'who am i')
w	wie 'who', zusaetzlich idle-time, CPU-Auslastung u. letztem Befehl
mv	move - Datei umbewegen (in anderes Directory) oder umbenennen
df -k	disk free - freien Festplattenplatz anzeigen
free	informiert ueber Auslastung von ram und swap
more <file>	Zeigt Inhalt von 'file' seitenweise an
cat <datei>	datei anzeigen. Mit Zusatz '|more' seitenweise anhalten
head <file>	listet die ersten 10 Zeilen eines Text-Files (head|tail -30 <file>
tail <file>	listet die letzten 10 Zeilen eines Text-Files (... zeigt 30 Zeilen)
>	Standardausgabe umlenken z.B. cat > datei schreibt das File 'datei' mit den Eingaben von der Console (Beenden mit ctrl-d) (Achtung: wenn File datei vorhanden, wird Originalinhalt geloescht)
>>	Standardausgabe umlenken z.B. cat >> datei schreibt ins File 'datei' und haengt an bestehenden Inhalt an; Ende mit ctrl-d
touch <file>	erzeugt leeres File
chmod	aendert Zugriffsrechte
chown	aendert (Datei)-Besitzer
chgrp	aendert die Gruppenzugehoerigkeit eines Files
last -10	listet die letzten (juengsten) 10 Logins aller User
last -5 dg2mic	" " " 5 Logins von dg2mic
&	startet einen Prozess im Hintergrund, wenn es als Parameter am Ende der Befehlszeile steht
ctrl-z	ctrl-z schickt den aktuellen Prozess in den Hintergrund (so als ob

```
            gleich mit dem Befehl am Schluss das & gestanden haette)
fg #        holt den Prozess # aus dem Hintergrund in den Vordergrund. # ist
            die beim Start des Prozesses in [] angegebene Nummer, NICHT die
            mit dem Befehl 'ps' ermittelte PID
kill #      Loescht den Prozess #. # ist die PID (ermitteln mit Befehl 'ps')

ps          zeigt den Prozess-Status aller Prozesse des Users. Die Zahl am
            Zeilenanfang ist die Prozess-Nr., die man in 'kill #' benoetigt
write <user> Nach dem Absetzen dieses Befehles, gehen alle Eingaben als msg an
            den (eingeloggten) User. Beendigung mit ctrl-d
talk <user> Wie 'write <user>', aber nur mit TCP/IP-Usern, die einen ANSI-
            Bildschirm haben
echo "<text>" | wall
            Broadcast-msg an alle eingeloggten User (wall = Write ALL)
            Das Pipe-Zeichen "|" leitet die Ausgabe des links davon stehenden
            Prozesses (echo ...) an den rechts davon stehenden Prozess weiter
stty echo   Schaltet das Echo ueber Funk ein (ist normalerweise aus). Wieder
            ausschalten mit 'stty echo-'.
sleep #     Erzeugt einen Prozess, der sich nach # Sekunden wieder beendet
```

4. Handwerkszeug: Die EDITOREN ed und vi

Der Editor ed ist (wie der EDLIN bei MS-DOS) ein Relikt aus den Anfangstagen des Computerzeitalters. Fuer AX-25-User, die ueber Funk kein ANSI-faehiges Terminal anbieten koennen, ist er das wichtigste Arbeitsmittel zur Bearbeitung von Files im Host. TCP/IP-User koennen z.B. auf den vi ausweichen (s.u.).

Die Bedienung kann in der man-page des ed nachgelesen werden. AX.25-Usern ist zu empfehlen, die man-page zuerst am Host in ein File zu schreiben und das File dann zu uebertragen und als ASCII-File mitzuschreiben. Dadurch entgeht man den vielen stoerenden ANSI-Steuerzeichen, die eine direkte Auswertung der Hilfe am Bildschirm fast unmoeglich machen.

Wer es dennoch probieren will, sollte folgendes wissen und beherzigen:

a) mit dem 'Space'-Zeichen gefolgt von ctrl-j (LineFeed) und <CR> wird die folgende Seite angefordert. Achtung: Reihenfolge einhalten!
b) mit <CR> wird die naechste Zeile angefordert
c) mit q<CR> wird die Ausgabe abgebrochen.

Das 'downgeloadete' File laesst sich off-line muehelos ausdrucken.
Die empfohlene Befehlsfolge sieht fuer ein BayCom-Terminal so aus:

```
cd $HOME                        ins eigene home-Directory wechseln
man ed > ed.man                 man-page in file $HOME/ed.man schreiben. Dabei
                                werden NUR die Hervorhebungen FETT und UNTER-
                                STRICHEN verwendet (jeweils durch BackSpace
                                vom Nutzzeichen getrennt), was ein Drucker
                                verarbeiten koennen sollte.
:WRITE C:\TEMP\ED.MAN           Oeffnet File ED.MAN im (existierenden!)
                                Directory C:\TEMP (der Befehl ist der von
                                BayCom-Terminal und muss ev. angepasst werden)
cat ed.man                      Liest und uebertraegt das File ed.man
rm -iv ed.man                   File auf Host loeschen (Bitte nicht vergessen,
                                Fragen des Host mit y beantworten, wenn alles
                                stimmt!)
:WRITE OFF                      File am eigenen Terminal schliessen (BayCom)
:OSHELL                         Wechsel von BayCom-Terminal in eine DOS-Shell
```

```
PRINT C:\TEMP\ED.MAN            File drucken
DEL C:\TEMP\ED.MAN              eigenen Plattenplatz schonen...., hi.
EXIT                            zu BayCom-Terminal zurueckkehren
```

Ein neues File kann statt mit ed auch mit dem cat-Befehl erstellt werden. Der Befehl 'cat > neuesfile' informiert Unix, dass alle folgenden Zeichen von der Console in das File 'neuesfile' geschrieben werden sollen, bis ein ctrl-d kommt. Achtung: Falls das File 'neuesfile' schon existiert, wird der Inhalt durch den Befehl cat > neuesfile gnadenlos geloescht! Bitte auch sicherstellen, dass das eigene Terminal-Programm ein ctrl-d senden kann! Manche Terminal-Programme koennen kein ctrl-d senden (z.B. GP), da sie das Zeichen selbst fuer interne Zwecke verwenden. Dann bitte im eigenen Rechner ein File mit dem Namen CTRL-D. erstellen und nur ein ctrl-d (hex 04) hineinschreiben. Das File kann mit dem READ-Befehl des Terminals gelesen und damit das Zeichen gesendet werden. Aehnlich sollte man es bei BayCom mit dem ESC, ctrl-c und Doppelpunkt (COLON) halten.

Bei der nachfolgenden Befehlsauswahl werden alle ed-Befehle ohne optionale Parameter verwendet, um die Bedienung des ed nicht unnoetig zu verkomplizieren.

ed datei	Standard-Aufruf, laedt ed und das zu editierende File 'datei' Nach Ausgabe der Filelaenge wartet der Editor im Command-Mode auf den ersten Befehl
ed -p# datei	wie vorstehend, aber User wird zu Eingaben durch das Prompt # am Zeilenanfang aufgefordert. Das ist vor allem ueber Funk und fuer ungeuebte Benutzer sinnvoll, da man so z.B. nach einem a-Befehl sicher sein kann, dass eingegebenen alle Zeichen so lange als Texteingabe behandelt werden, wie KEIN neuer Prompt erscheint.

Nach dem Laden des Files steht der Zeilenpointer auf der LETZTEN Zeile des Files. Zum Verschieben des Pointers dienen folgende Befehle:

+ oder +n	eine Zeile bzw. n Zeilen nach unten (hinten), n ist eine Zahl
- oder -n	Pointer eine Zeile bzw. n Zeilen nach oben (vorne)
$	Pointer auf letzten Zeile stellen
n	springen zur Zeile n (z.B. mit der Eingabe 1<CR> zur Zeile 1)

Als Reaktion wir der Inhalt der jeweils neuen Zeile ausgegeben. Durch einen Punkt (gefolgt von der Eingabetaste) wird ebenfalls die aktuelle Zeile ausgegeben. Damit kann man kontrollieren, ob man sich in der gewuenschten Zeile befindet!!! Falls man eine Leerzeile erwischt hat, zeigt zumindest der #-Prompt (falls eingestellt!) an, dass ueberhaupt eine Reaktion erfolgte.
Nach dem Start des ed und dem Laden des zu bearbeitenden Files befindet man sich im Command-Mode. Auch nach den meisten Befehlen landet man automatisch wieder im Command-Mode (erkennbar am Prompt, falls aktiviert).
Ausnahmen sind die Befehle a (append) und c (change). Bei diesen muss die Texteingabe durch einen Punkt am Zeilenanfang, gefolgt von Carriage Return (nachfolgend mit <CR> bezeichnet) beendet werden.

d<CR>	delete - loescht die aktuelle Zeile. Der Pointer geht auf die Zeile NACH der geloeschten (falls verhanden, sonst auf die davorliegende). Loeschen aufeinanderfolgender Zeilen also zweckmaessig von oben nach unten!
a<CR>	append - fuegt den anschliessend eingegebenen Text NACH der aktuellen Zeile ein. Die Eingabe muss mit einem .<CR> am Anfang einer Zeile beendet werden. Der Pointer steht auf der letzten eingegebenen Zeile (Kontrolle durch Eingabe eines weiteren .<CR>, was die aktuelle Zeile ausgibt)
c<CR>	change - ersetzt die aktuelle Zeile durch den nachfolgend

```
                eingegebenen Text. Beenden der Eingabe mit .<CR>
s/alt/neu/<CR>  substitute - ersetzt die Zeichenfolge 'alt' in der aktuellen
                Zeile durch die Zeichenfolge 'neu'. Der Pointer bleibt auf
                der geaenderten Zeile. Kontrolle durch .<CR> empfohlen!
                Falls die Zeichenfolge 'alt' in der Zeile mehrfach auftaucht,
                wird nur die erste Folge ersetzt.
u<CR>           undo - macht das letzte Kommando rueckgaengig
P<CR>           Prompt on/off. Schaltet einen definierten Prompt aus (oder
                wieder ein, dabei dient allerdings immer der * als Prompt)
w datei<CR>     write into file 'datei'. Schreibt das Arbeitsergebnis ins
                File 'datei'
wq<CR>          write & quit - schreibt das geaenderte File unter dem Default-
                Namen auf die Platte und verlaesst den ed.
q<CR>           quit - Verlassen des ed. Falls nicht vorher alles gespeichert
                wurde, erfolgt eine Warnung.
Q<CR>           Quit - Verlassen des ed OHNE zu speichern und OHNE Warnung.

-------------------------------------------------------------------------------

Der Editor vi (ein weiteres grausames Kapitel der EDV-Steinzeit) kennt
3 Eingabe-Modi und benoetigt ein ANSI-faehiges Terminal, das im allgemeinen
ueber eine AX.25-Amateurfunk-Verbindung nicht realisierbar ist. Zumindest
werden einige der benoetigten Tasten (ESC, Doppelpunkt, Ctrl-C) fuer die
oertliche Terminal-Steuerung benoetigt...!

Die 3 Modi sind:

1. Command-Mode     jeder Tastendruck wird als Teil eines Befehls interpretiert
2. Input-Mode       jeder Tastendruck wird als Texteingabe interpretiert
3. last line Mode   fuer komplexere Kommandos, die in der untersten Zeile
                    des Bildschirms editiert werden

Zum Wechsel zwischen den Modi dienen die Tasten [ESC] und der Doppelpunkt [:],
wobei ESC auch gleichzeitig eine Eingabe im Input-Mode beendet. Der : muss im
Command-Mode eingegeben werden. Die wichtigsten Kommandos sind:

i         wechselt in den Input-Mode. Zeichen werden an der aktuellen Cursor-
          position eingefuegt (insert)
a         wechselt in den Input-Mode. Zeichen werden nach der aktuellen Cursor-
          position angefuegt (append)
s         wechselt in den Input-Mode. Zeichen, die an der aktuellen Cursor-
          position stehen werden ueberschrieben (substitute)
C         wechselt in den Input-Mode. Der Rest der Zeile ab der aktuellen
          Cursorposition wird durch neuen Text ersetzt
o         wechselt in den Input-Mode. Nach der aktuellen Zeile wird eine neue
          Zeile eingefuegt
O         wechselt in den Input-Mode. Vor der aktuellen Zeile wird eine neue
          Zeile eingefuegt
x         loescht das aktuelle Zeichen
dd        loescht die aktuelle Zeile
u         nimmt das letzte Kommando zurueck
.         wiederholt das letzte Kommando
[ESC]     Durch druecken der ESC-Taste wechselt man vom Input-Mode wieder
          zurueck zum Command-Mode

:         wechselt vom Command-Mode in den last-line Mode
:e datei  laedt das File 'datei' in den Arbeitsspeicher zum Editieren
:q!       Beendet den vi OHNE zu speichern (wenn man Mist gebaut hat, hi)
:x        Ende MIT speichern unter dem (bisherigen) Default-Filenamen
:w datei  speichert das Arbeitsergebnis unter dem (neuen) Filenamen 'datei'
```

5. Wichtige Server (Programme) fuer den User

mail <adressat> E-Mail an einen anderen User schreiben. Bei vernetzten Systemen werden die Mails auch weitergeleitet. Nach der Eingabe des Subject, zu der man aufgefordert wird, gehen alle weiteren Zeile in den 'Body' der Mail. Das End der Message wird durch einen Punkt am Zeilenanfang markiert (Wie bei a-Befehl des Editors ed)

telnet <hostname> Verbindung zu einem anderen Host ueber TCP/IP herstellen Nach dem Einloggen arbeitet man im Zielsystem so als ob man dort direkt eingeloggt waere. Beenden mit 'exit'.

ping <hostname> Aussenden von Testpaketen an einen anderen Host und Messung der Datenlaufzeit. Achtung: ueber Funk sicherstellen, dass die Pakete nicht wie im Internet alle Sekunde ausgesendet werden! Das muellt die qrg zu!

finger <username> Liefert Informationen ueber einen dem System bekannten User

7plus <[path/]file> Packt das File, wenn es nicht die Extention .7pl oder .p01 usw. hat, in 7PLUS-Files von 10k Laenge. Die Files landen im Default-Directory. Wenn der Filename die Extention .7pl oder .p01 hat, werden die 7PLUS-Files entpackt.

zip <outfile[.zip]> <infile>
Packt das File <infile> mit PKZIP in ein File <outfile>.zip ins Default-Directory. Die Angabe .zip darf fehlen.

unzip <file.zip> Entpackt ein *.zip-File

gzip <file> GNU-Zip-Programm. Zippt ein File, l”scht es und ersetzt es durch <file>.gz (die gezippte Version). Der Aufruf mit Parameter -d macht das gleiche umgekehrt: gzip -d <file>.gz Um die Ausgangsversion zu erhalten, muss VORHER ein Kopie gemacht und das Original gezippt werden, damit der Pfad korrekt eingetragen wird.

tar GNU-Archivierungsprogramm. Die Archive tragen die Extention

.tar oder wenn gleichzeitig auch mit gzip komprimiert wird, .tgz und bei nachtraeglicher Komprimierung .tar.gz
Die beiden Moeglichkeiten werden wie folgt aufgerufen:

gzip -dc <file>.tar.gz | tar xvofp -
Vollstaendiger Befehl zum Entpacken eines *.tar.gz Archivs. Parameter:
-d = decompress, -c = use standard output AND keep original file(s). Die Pipe uebergibt an tar mit den Optionen:
x = extract, v = verbose (ausfuehrlichste Infos), o = old file format (not ANSI), f = use filename as archive input file, p = extract all permission rights too (Uid, Gid).

tar [-]xvzf <file>.tgz
entpackt und ent-archiviert ins gegenwaertige Default-Directory (vorher anlegen und hineinwechseln). Neue Subdirectories werden angelegt, falls noetig. Die Parameter

x, z und f sind MUSS-Parameter, f MUSS an letzter Stelle
stehen und das Minuszeichen darf fehlen, obwohl dies in der
mit   tar --help   angezeigten Hilfe nicht steht.
Bedeutung der Parameter:
x = extract with full path. Statt x kann auch stehen:
    t = list contents of archive
    r = append files to archive (Gegenteil von x)
    c = create new archive (vor r anzuwenden)
    d = find differences between file system and archive
v = verbose (umfangreichste Infos liefern)
z = filter archive through gzip (fehlt wenn Endung = .tar)
f = use filename as archive input file

(wird fortgesetzt)

Und nun viel Spass beim Ausprobieren der Befehle und des ed und/oder vi (hi),
beim Zippen oder Archivieren.

Kommentare, Korrekturen und Verbesserungsvorschlaege werden dankend erbeten
an Hartwig, DG2MIC @ DB0FSG.#BAY.DEU.EU (AX.25) oder
            dg2mic@db0hy.ampr.org (TCP/IP-AmprNet) oder
            Hartwig.Harm@muenchen-land.baynet.de (Internet)

Filename bei DG2MIC: c:\bay\r\linux\doslinux.hlp