Anleitung unix2webd

Unix2webd ist ein einfacher HTTP(S)-Server, der es ermöglichen soll, die Ausgaben von Unix-Programmen als Web-Seiten darzustellen. Um Programme mit unterschiedlichen Parametern aufrufen zu können, können über eine automatisch generierte Formularseite Aufrufparameter bestimmt werden.

Aufruf des Programms:

unix2webd [-a] [-b path] [-c] [-d] [-e] [-f|-ff] [-fc] [-fp path] [-g|-gg] [-h home] [-i path] [-k] [-l|-lp path] [-o|-op path|-og path|-oG path] [-of] [-or ips] [-p port] [-q] [-r text] [-s|-sp path|-sg path|-sh path|-sG path] [-sU user] [-sP pwd] [-sB B64] [-sf] [-t sec] [-tu mysec] [-u user] [-v] [-w] [-D] [-E] [-H] [-N] [-P] [-S] [-Sc path] [-Sk path] [-Sa path] [-Sf] [-U|-UU] [-V [-Ve path]|-Vh] [-?]

-b <path> : Pfad für externe Binaries.
-c : ?&edit verhindern.
-d : Directories werden mit Links angezeigt.
-e : u2w-Dateien werden nicht ausgewertet.
-f : Uploads mit POST erlauben.
-ff : Uploads mit POST auch ohne Anmeldung.
-fc : Bei Uploads Dateiname vom Client übernehmen.
-fp <path> : Pfad für Uploads mit POST.
-g : Uploads sind mit PUT erlaubt.
-gg : Uploads sind mit PUT ohne Angabe von User und PWD erlaubt.
-h <htmlpfad> : Root des Servers ist <htmlpfad> und nicht ./html
-i <path> : Pfad für *.i2w Dateien.
-k : Keepalive erlauben
-l : Logging einschalten
-lp <path> : Logging in Datei <path>
-o : Zugriff wird über .hosts Datei geregelt.
-of : .host Datei muss vorh. sein, sonst kein Zugriff auf Verzeichnis
-op <pfad> : Es gibt nur eine globale .hosts-Datei unter <pfad> (muss nicht .hosts heissen).
-og <path> : <path> ist .hosts-Datei für alle Verzeichnisse ohne .hosts.
-or <ips> : Zugriff nur von den angegebenen IP-Adressen.
-p <port> : unix2webd verwendet Port <port> und nicht 3232
-q : Keine Meldungen ausgeben.
-r <text> : <text> im Browser-Anmeldefenster zeigen.
-s : Zugriff wird über .passwd Datei geregelt, nicht über User.
-sf : .passwd Datei muss vorhanden sein, sonst kein Zugriff auf Verzeichnis.
-sh <path> : .passwd-Datei für alle Verzeichnisse und Root-Verzeichnis des Servers nach Anmeldung aus .passwd-Datei.
-sp <path> : Es gibt nur eine globale .passwd-Datei unter <pfad> (muss nicht .passwd heissen).
-sg <path> : .passwd-Datei für alle Verzeichnisse ohne .passwd-Datei.
-sU <user> : User für Anmeldung festlegen
-sP <pwd> : Kennwort für Anmeldung festlegen
-sB <b64> : User:Kennwort Base64 kodiert
-t <sek> : Timeout der Verbindung auf <sek> Sekunden.
-tu <mysek> : Timeout der Verbindung auf <mysek> Mikrosekunden.
-u <user> : Server läuft als <user>, wenn durch Seite kein User und PWD erfragt. Std User ist sonst nobody.
-v : Version des Programms ausgeben.
-w : Upload von .?2w und .?3w Dateien erlauben.
-D : unix2webd als Daemon laufen lassen.
-E : leere Parameter nicht an Programme übergeben.

-H : Es wird kein HTTP/1.1 Header gesendet.
-N : Methode OPTIONS nicht erlaubt.
-P : Jede Seite erfordert die Angabe von User und PWD.
-S : HTTPS verwenden.
-Sc <path> : Datei mit SSL-Certifikat.
-Sk <path> : Datei mit SSL-Key.
-Sa <path> : Datei mit SSL-CA.
-Sf : Force Client Zertifikat.
-U : Server wechselt den User nicht.
-UU : Server wechselt User erst nach Öffnen von *.u2w.
-V : Root des Servers ist <htmlpfad>/<username>.
-Ve <path> : Root des Servers ist <htmlpfad>/<path>, bei keiner Anmeldung.
-Vh : Root des Servers ist Home-Verzeichnis des angemeldeten Users.
-? : Diese Kurzhilfe anzeigen.

Wird keine Seite angefordert, so wird im htmlpfad nach den Dateien index.u2w, index.html und index.htm gesucht.

Das u2w Format:

Wenn der unix2webd-Server eine Seite übertragen soll, die mit .u2w endet, dann wird die Datei ausgewertet und daraus eine HTML-Seite generiert.

Aufbau einer u2w Datei:

Ab der Version 8 ist die Sonderbedeutung der ersten beiden Zeilen aufgehoben. Die Titelzeile muss nun mit %title <Titel> angegeben werden. Die Kopfzeile kann mit %h2 <Kopfzeile> %/h geschrieben werden. Alle anderen Zeilen werden so dargestellt, wie sie in der Datei vorliegen. Leerzeilen werden als Absatzwechsel betrachtet.

Hyperlinks können mit %hlink(”<Linkurl>”, “Beschreibung”) erzeugt werden. Es auch noch die alte Form &<Linkurl>&Beschreibung& verwendet werden. Dabei müssen &-Zeichen in der <Linkurl> quotiert werden: &. Alternativ kann die gesamte <Linkurl> in ’”’ eingeschlossen werden: &”<Linkurlmit&>”&Beschreibung&.

Der Aufruf von Unix-Programmen wird durch das Zeichen ’‘’ eingeleitet und beendet. Beispielsweise fuegt ‘date‘ das aktuelle Datum in die Seite ein. Vor der Ausgabe werden alle Newlines der Unixausgabe in Leerzeichen umgewandelt. Sollen Ausgaben, die mehr als eine Zeile erzeugen sinnvoll dargestellt werden, so sollte der Befehl in einer eigenen Zeile stehen. Durch das Weglassen des zweiten ’‘’ wird erreicht, dass für die Ausgabe ein Zeichensatz mit fester Zeichenlänge verwendet wird und Zeilenumbrüche beachtet werden. Alternativ zu ‘unixbefehl‘ kann auch #unixbefehl# verwendet werden, es duerfen dann im unixbefehl ’‘’ vorkommen und Newlines werden nicht mehr umgewandelt. Quoting funktioniert noch nicht. Zur Erzeugung von Option- und Select-Feldern sollte ‘...‘ verwendet werden und zur Vorbelegung von Textareaformularen sollte man #...# benutzen.

Ab hier steht die Ausgabe von ’df -k’:

‘df -k

oder

#df -k

Möchte man Parameter an Befehle übergeben, so kann man diese über ein HTML-Formular erfragen lassen. Dazu wird in der zweiten Zeile der u2w-Datei der Befehl %parameter eingegeben. Die Zeile direkt nach %parameter ist dann die Überschrift des Formulars. Die Zeilen danach bis zum %end Befehl erzeugen die Seite mit dem Formular und die Zeilen nach dem %end Befehl die Seite nach Eingabe der Parameter und Bestätigung mit OK, wobei die

erste Zeile nach %end die Überschrift der Seite ist. Sollen am Ende des Eingabeformulars keine OK- und CLEAR-Buttons erscheinen, kann ”%end nb” eingeben werden. Die Buttons, die erzeugt werden sollen, können angebeben werden: ”%end ok|ab|back|clear”. Mit ok -> OK-Button, ab -> Abbruch-Button, back -> Zurück-Button und clear -> Clear-Button. Wenn die automatisch erzeugten Buttons nicht am Ende der Seite stehen sollen, so können die Buttons mit %endpar [ok|ab|back|clear] erzeugt werden. Die Formularseite ist dann bei %end zuende. Mit der Syntax ok:”<beliebiger Text>” können beliebige Buttons angezeigt werden. Die Variable %ok hat nach dem drücken dieses Buttons dann den Wert <beliebiger Text> und nicht OK. Neben %parameter gibt es noch %paramter_get und %parameter_multipart. Bei %parameter_get wird der Browser angewiesen, die Einträge aus dem Formular mit der Methode GET an den Server zu schicken. Man kann die Parameter dann direkt in der URL ablesen, so dass man die Ergebnisseite als Favorit speichern kann. Mit %parameter_multipart erreicht man, dass die Parameter als Multipart-Content-Block gesendet werden. Das ist die einzige Möglichkeit, um Dateiinhalte vom Browser abzurufen. Wird vor oder hinter %parameter %resultpage[:<target>] <url> angegeben, dann wird <url> als Ergebnisseite des Formulares eingetragen. Ist <target> angegeben, wird die Ergebnisseite dort angezeigt. %setoben <obenurl>: Oberer Teil einer Menüseite ändern, %setrechts <rechsturl>: Ergebnisteil einer Menüseite ändern und %setlinks <linksurl>: Linker Teil einer Menüseite ändern.

Es werden folgende Formularfelder unterstützt:

Textfelder, Passwortfelder, Checkboxen, Radiobuttons, Textareas, Optionsfelder mit einfacher und mehrfacher Auswahl und Linklisten. Parameter werden mit einem ’%’ Zeichen eingeleitet:

```
%text[:<br>:<max>[:ro]] <name>[, <Vorbelegung>][; <Beschreibung>]
%password[:<br>:<max>[:ro]] <name>[, <Vorbelegung>][; <Beschreibung>]
```

Erzeugen Felder zur Texteingabe oder Passworteingabe. Der eingegebene Wert kann anschliessend mit $<name> referenziert werden. Ist <Beschreibung> angegeben, steht vor dem Feld die <Beschreibung>, andernfalls <name>. Der optionale Parameter
 gibt die Breite des Feldes an. Über <max> kann man die Einzugebenden Zeichen begrenzen. Mit :ro wird ein Readonlyfeld erzeugt.

```
%textarea[:<b>:<h>[:ro]] <name>[, <Vorbelegung>][; <Beschreibung>]
```

Ergibt ein Feld zur Eingabe eines längeren Textes. Die Größe der Box kann mit <b> für Breite und <h> für Höhe eingestellt werden. Mit :ro wird ein Readonlyfeld erzeugt.

```
%check[:<sp>[::ro]] <name> <c1>[:<t1>] ... <cn>[:tn>] [, <Vorbel.>][; Beschr.]
```

Erzeugung von Checkboxen. Ist der optionale Parameter <sp> angegeben, so werden bis zu <sp> Boxen in einer Zeile angezeigt. Ist <t_n> angegeben, dann wird <t_n> als Beschreibung ausgegeben. Die Variable $<name> enthält alle Variablennamen der Checkboxen (<c_1> <c_2> ... <c_n>), die markiert sind. Um alle markierten Werte zu bestimmen, kann man %par(<name>[, <trenn>]) verwenden. %par(<name>[, <trenn>]) hat als Ergebnis alle <c_n>, die Markiert sind. Ist <trenn> angegeben, dann sind die einzelnen Werte durch <trenn> getrennt. Mit :ro wird ein Readonlyfeld erzeugt.

```
%checkbox <name>[, 1][; <Beschreibung>]
```

Es wird ein Button zum Markieren angezeigt. Ist das Feld markiert, enthält $<name> den Wert ’1’, ist das Feld nicht markiert, ist $<name> nicht definiert. Ist ”, 1” angegeben, dann ist Das Feld schon markiert.

```
%option[:<a>[::ro]] <name> <o1>[:<t1>] ... <on>[:<tn>] [, <Vorbel.>][; <Beschr.>]
%select[:<a>] <name> <o1>[:<t1>] ... <on>[:<tn>] [, <Vorbel.>][; <Beschr.>]
```

Erzeugung von Optionsfelder, mit einfacher Markierung option und mehrfacher Markierung, select. Bei option wäre eine Vorbelegung z. B. <o_2>, bei select z. B. <o_1> <o_3> <o_5>. Ist :<t_n>

angegeben, so wird zur Auswahl des Wertes o_n der Text t_n angezeigt. Ist der optionale Parameter <a> angegeben, so bestimmt er die Anzahl sichtbarer Wahlmöglichkeiten.

`%radio[:<sp>] <name> <r`$_1$`>[:<t`$_1$`>] ... <r`$_n$`>[:t`$_n$`>] [, <Vorbel.>][; Beschr.]`

Erzeugung von Radiobuttons. Ist der optionale Parameter <sp> angegeben, so werden bis zu <sp> Buttons in einer Zeile angezeigt.

```
%radiobutton <name> <r1> [, Vorbelegung][; Beschreibung]
%radiobutton <name> <r2> [, Vorbelegung][; Beschreibung]
...
%radiobutton <name> <rn> [, Vorbelegung][; Beschreibung]
```

Einzelne Angabe der Radiobuttons.

```
%file <name> [; Beschreibung]
```

Es wird ein Feld zum Auswählen einer lokalen Datei angezeigt. <name> erhält dann den Pfad der ausgewählten Datei. Wenn das Formular mit %parameter_multipart beginnt, dann wird vom Browser die ausgewählte Datei mit Post an den Server geschickt. Dazu muss unix2webd mit -f gestartet sein. Dann wird eine Datei mit Namen <name> im Verzeichnis, das mit -fp spezifiziert wurde abgelegt. Ist der Server mit -fc gestartet, dann wird der Dateiname vom Client übernommen. Ist -fp nicht angegeben, dann lautet das Verzeichnis upload. Nach Erfolg werden noch folgende Variablen fürs Ergebnisfenster generiert: <name>_file ist der Basename der gesendeten Datei, <name>_path ist der komplette Pfad, <name>_error sind gegebenenfalls Fehlermeldungen, <name>_mime ist der Mime-Type und <name>_store ist der Pfad, unter dem die Datei gespeichert wurde. Wenn -f/-fp nicht angegeben ist, dann kann der Inhalt der Ausgewählten Datei über <name> angesprochen werden. Allerdings ist die Größe auf etwa 32 kByte beschränkt.

`%linklist[:<sp>] <name> <l`$_1$`>[:<t`$_1$`>] ... <l`$_n$`>[:<t`$_n$`>] [; <Beschr.>]`

Es wird eine Liste mit Hyperlinks generiert. Dargestellt wird jeweils $<t_n>$ wenn angegeben, sonst $<l_n>$ und <name> erhält den Wert $<l_n>$. Ist <sp> angegeben, dann werden die Links in einer Tabelle mit <sp> Spalten angeordnet.

```
%var <name>, Belegung
```

Es wird ein Parameter <name> mit Belegung belegt. Nützlich, um anderen u2w-Dateien Parameter mitzugeben, mit der die Verarbeitung gesteuert werden kann.

```
%okbutton [<value>]
```

OK-Button erzeugen. Ist <value> angegeben, dann mit Beschriftung <value> sonst mit Beschriftung OK. %ok enthält in der Ergebnisseite den Wert des OK-Buttons, also “OK” oder <value>.

```
%pbutton <name>; <text>
```

Submit-Button in Formularen mit Namen <name> und Wert <text>.

```
%hauptmenu
```

Hauptmenübutton in Menüseiten erzeugen.

```
%let <name>=Belegung
%let <name>, Belegung
%leti <name>=Belegung
%letf <name>=Belegung
%lets <name>=Belegung
%leta <name>=Belegung
%letai <name>=Belegung
```

```
%letaf <name>=Belegung
%letas <name>=Belegung
```

Der Parameter <name> wird sofort mit Belegung belegt. Bei %var steht der Wert erst nach dem Ausfüllen des Formulars zur Verfügung. Der Wert von <name> nach %let gilt nur im aktuellen Fenster. Wird eine Berechnung oder Textersetzung öfter benötigt, kann man das Ergebnis speichern. Bei der Verwendung von %let <name>=Belegung werden Leerzeichen, die hinter dem '=' stehen, mit zur Belegung gezählt. Sollen in einer Seite mit %parameter Variablen in beiden Fenstern (Parameter- und Ergebnisfenster) zu sehen sein, dann muss %let vor dem ersten %parameter stehen. Bei %let wird der Type automatisch bestimmt. Soll der Typ festgelegt werden, wird mit %leti ein Integer, mit %letf ein Float und mit %lets ein String zugewiesen. %leta[ifs] hängt einen zusätzlichen Wert an die Variable an. Die einzelnen Werte können dann mit %par oder %foreach durchlaufen werden.

Beispiel:

```
%let anz=‘ls | wc -l‘
%if $anz > 0
Im Verzeichnis sind $anz Dateien.
%else
Im Verzeichnis sind keine Dateien.
%fi
```

```
%allvars, %allpars
```

%allvars ergibt Liste mit allen Variablen, die mit %let definiert sind. %allpars sind alle Parameter, wie vom Browser geliefert.

```
%par(name[, <sep>])
```

%par gibt den Inhalt des Parameters name aus. Enthält name mehrere Werte, dann werden sie durch <sep> getrennt.

```
%skip_empty, %/skip_empty
```

%skip_empty sorgt dafür, dass nicht definierte Variablen nicht an Programme übergeben werden. %/skip_empty leere Variablen werden wieder an Programme übergeben. Standardmäßig werden leere Variablen an Programme übergeben, damit die Positionen der einzelnen Werte immer gleich ist.

```
%setquote(<quote>[, <var>])
```

Bestimmte Zeichen in Variablen sollen gequotet werden. Das erste Zeichen in <quote> gibt das Zeichen an, mit dem gequotet wird und alle anderen Zeichen bestimmen die Zeichen, die gequotet werden sollen. Z. B. <quote>==”’@”, dann werden alle ”’” und ”@” als ”’” bzw. ”5” an Shellprozeduren übergeben. Ist <var> angegeben, dann gilt die Regel nur für diese Variable.

```
%split(<w>, <c>[, <nv>[, <nb>]])
```

<w> am Zeichen <c> Zerlegen. Sind <nv>/<nb> angegeben, dann nur <nv>te bis <nb>te Element ausgeben.

```
%substr(<string>, <start>[, <länge>])
```

Teilstring zurückliefert, ist <länge> negativ, dann wird vom Ende gezählt.

```
%strpos(<string>, <such>[, <pos>])
```

Substring <such> in <string> ab Postion <pos> suchen. Es wird die Startposition zurückgegeben.

```
%replace(<r>, <s>, <t>)
```

Alle <r> in <t> durch <s> ersetzen.

`%rand([<von>[,<bis>]])`

Zufallszahl im Bereich <von> bis <bis> generieren. Sind <von> und <bis> Integerzahlen, dann wird eine Intergerzahl generiert, sonst eine Gleitkommazahl. Sind <von> und <bis> nicht angegeben, dann wird eine Zahl im Bereich 0.0 bis 1.0 generiert.

`%date([<format>[,<date>]])`

Datum formatiert ausgeben. Ist <date> nicht angeben, dann wird das aktuelle Datum/Zeit ausgeben. Ist <format> nicht angeben, dann ist das Format: tt.mm.jj HH:MM:SS. Im Formatstring können folgende Codes verwendet werden:

```
%a: Wochentag (Mo, Di, Mi...)
%b: Monatsname (Jan, Feb, Mär...)
%m: Monat
%d: Tag
%y: Jahr (zwei Ziffern)
%Y: Jahr (vier Ziffern)
%H: Stunde
%M: Minuten
%S: Sekunden
%w: Wochentab (1-7)
%%: %
```

`%isodate(<datum>)`

Datum aus dem Format tt.mm.jj ins Isoformat (jjjj-mm-tt) umwandeln. Wird <datum> im Isoformat übergeben, dann wird <datum> zurückgeliefert.

`%nonl, %/nonl`

Normalerweise werden alle Parameterfelder in einer eigenen Zeile gedruckt. Ist %nonl angegeben, dann gilt nur noch %n als Zeilenumbruch. Insbesondere kann man mit %nonl Tabellen mit mehreren Parameterfeldern in einer Zeile erzeugen. %/nonl schaltet wieder in den Normalzustand.

Beispiel:

```
%table:lll border; Tabellenüberschrift
Art|Feld 1|Feld 2
%nonl
%for i 1 2 3
Art $i
%text feld1_$i;
%text feld2_$i;
%n
%/for
```

`%button <Linkurl>; Text des Buttons`

Es wird ein Button erstellt, der beim Klicken auf die Seite <Linkurl> verzweigt.

`%backbutton[:<steps>] [; <Text des Buttons>]`

Es wird ein Button gezeigt, der in der Historyliste des Browsers einen Schritt oder wenn <steps> angegeben, dann <steps> Schritte zurück geht. Ist <Text des Buttons> nicht angegeben, steht Zurück im Button.

`%closebutton`

Button “Schließen” zum Schließen des Fensters.

`%table[:format] [border] [; Tabellenüberschrift]`

Ab hier wird Text und Parameter in einer Tabelle angeordnet, so dass alle Eingabefelder untereinander stehen. Die Tabellen spalten werden für Parameterfelder automatisch bestimmt.

Im Text können Spalten mit dem Zeichen ’|’ voneinander getrennt werden. Die Tabellenüberschrift wird zentriert über der Tabelle angezeigt. Um die Ausrichtung der einzelnen Spalten zu ändern, kann ein Format angegeben werden. Das Format bestimmt die Ausrichtung der Spalten. Es sind erlaubt: ’l’: linksbündig, ’c’: zentriert, ’r’: rechtsbündig, ’j’: Blocksatz, alle anderen Zeichen sorgen dafür, dass genau dieses Zeichen in allen Zeilen untereinander steht, z. B. ’,’ oder ’.’ für Dezimalbrüche. Anmerkung: Von den meisten Browsern wird nur ’l’, ’r’ und ’c’ unterstützt.

`%tablehead <Spaltenüberschriften>`

<Spaltenüberschriften>: durch ’|’ getrennte Spaltenüberschriften.

`%/table`

Hier endet die Anordnung in einer Tabelle. Bei %end wird die Tabelle automatisch beendet.

Beispiel:

```
%table border ; Plattenplatz
%tablehead Verzeichnis | frei | Prozent
/home | 10 k | 1
/usr | 30 m | 10
%/table
```

`%left`

Es wird auf Linksbündige Ausgabe geschaltet. Formulare sind normalerweise Zentriert.

`%center`

Es wird auf zentrierte Ausgabe geschaltet.

`%fs<n> %/fs`

Zeichengröße auf <n> stellen, Zeichengröße zurück.

`%tt %/tt`

Zeichensatz auf Courier New einstellen, Zeichensatz wieder normal.

`%h<n> %/h`

Headline Stufe <n> einschalten, wieder ausschalten.

`%b %/b`

Ausdruck in Fett, Fett aus.

`%u %/u`

Unterstreichen, Unterstreichen aus.

`%green %red %yellow %blue %magenta %aqua %black`

Ausdruck in den Farben Grün, Rot, Gelb, Blau, Magenta, Hellblau, Schwarz.

`%/green %/red %/yellow %/blue %/magenta %/aqua %/black`

Ende farbige Schrift.

`%n`

Absatzende einfügen.

`%html ... %/html`

Zwischen %html und %/html steht reiner HTML-Text, der identisch an den Browser geschickt

wird. Der HTML-Text kann sich über mehrere Zeilen erstrecken.

`%html(<html>)`

<html> Code an den Browser schicken.

`%head ... %/head`

Zwischen %head und %/head steht HTML-Text für den <HEAD>-Bereich einer HTML-Seite. Z. B. Java-Scripte, CSS...

`%user`

Einfügen des aktuell angemeldeten Users (vom Browser erhalten).

`%pwd`

Einfügen des vom Browser erhalten Kennwortes.

`%myport`

Port, unter dem der u2w-Server läuft.

`%clientip`

IP-Adresse des aufrufenden Clients.

`%this`

Dateiname der aktuellen *.u2w, *.s2w oder *.s3w.

`%thispage`

Komplette aktuell angeforderte Seite.

`%thishost`

Host aus der Anforderung.

`%env(<name>)`

Environment-Variable <name> auslesen.

`%background <bild>`

Hintergrund <bild> verwenden. Muß am Anfang der u2w-Datei vor der Zeile für den Titel stehen. Bei Parameterseiten direkt nach %parameter oder nach %end.

`%authorize`

Die Seite wird nur dargestellt, wenn Username und Passwort richtig angegeben sind. %authorize sollte in der ersten Zeile der u2w-Datei stehen.

`%login`

Aufforderung zur Eingabe des Kennwortes an Browser senden.

`%setuser <name>`

Ab hier wird die Verarbeitung als User <name> fortgesetzt. Dazu muss der Server mit -U oder -UU gestartet werden, denn sonst läuft er als nobody (oder der User, der mit -u gewählt wurde) und kann dann den User nicht mehr wechseln. Ist der Server mit der Option -UU gestartet, dann wechselt der Server auf den Standarduser, wenn in der ersten Zeile der *.u2w Datei kein %setuser steht. Der Username - bedeutet, dass der User nicht gewechselt wird.

`%mime <Mime-Type>`

Es wird im HTTP-Header der angegebene <Mime-Type> als Content-Type gesendet. Macht Sinn, wenn man für *.s3w den Mime-Type explizit angeben möchte, sonst wird “application/plain” gesendet. %mime muss vor der ersten Ausgabezeile stehen.

`%savename <dateiname>`

Vorbelegung des Dateinamens bei “Speichern unter”.

`%refresh <Sekunden> <URL>`

Die Seite soll nach <Sekunden> durch die <URL> ersetzt werden. %refresh muss vor der ersten Ausgabezeile stehen.

`%page_not_found`

Fehlermeldung “Page not found” an Browser senden.

```
%if Bedingung
...
%elif Bedingung
...
%else
...
%fi
```

Wenn die Bedingung nicht wahr ist, dann wird der Block bis %else oder %fi übersprungen. z. B.:

```
%if $mode==“bearbeiten”
Es wird bearbeitet ‘do-bearbeite $par‘
%else
Es wird nichts bearbeitet.
%fi
```

```
%for parname w1 w2 w3 ...
...
[%break[ for]]
[%continue]
...
%/for
```

For-Schleife, $parname nimmt nacheinander die Werte w_n an. %break verlässt die %for-Schleife.

```
%foreach parname1 parname2
...
[%break[ foreach]]
[%continue]
...
%/foreach
```

For-Schleife, $\$parname_1$ nimmt nacheinander die Werte von $parname_2$ an. Wichtig: Bei $parname_2$ muss der Name, also ohne ’$’, angegeben sein. Ein Parameter kann mehrere Werte durch Checkboxen oder %leta bekommen.

```
%while Bedingung
...
[%break[ while]]
[%continue]
...
%/while
```

While-Schleife.

```
%switch <wert>
%case w1
...
%case w2
...
[%default]
...
%/switch
```

Es wird der %case-Zweig ausgeführt, für den <wert> == w_n ist. Der %case-Zweig endet beim nächsten %case oder %default. Gibt es kein w_n == <wert>, dann wird der %default-Zweig ausgeführt, wenn er angegeben ist.

```
%break [while|for]
```

Verlassen einer Schleife. Ist [while|for] angegeben, dann wird diese Schleife beendet, sonst die innerste.

```
%continue [while|for]
```

Am Schleifenanfang beginnen. Ist [while|for] angegeben, dann wird diese Schleife neu begonne, sonst die innerste.

```
%parameter Bedingung
...
%end
```

Der Parameterblock wird ausgeführt, wenn die Bedingung wahr ist. Der Block mit %parameter ohne Bedingung wird nur ausgeführt, wenn noch keine Parameter übgergeben wurden. Von den %parameter-Blöcken mit Bedingung wird nur der erste Block, fuer den die Bedingung wahr ist ausgeführt. Es gibt immer nur die Parameter des letzten Formulars. Möchte man Parameter weiternutzen, dann müssen sie mit %var gesichert werden.

Beispiel:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Beginn Beispiel
%title Folgeformulare
%parameter
%h2 Was soll gemacht werden?%/h
%option form ue:“User einrichten” ua:“User ändern”; Aktion
%end
%parameter $form==“ue”
%h2 Welcher User soll eingerichtet werden?%/h
%text user; Username
%var form, einrichten
%end
%parameter $form==“ua”
%h2 Welcher User soll geändert werden?%/h
%option user ‘cat userliste‘; Username
%var form, aendern
%end
%if $form==“einrichten”
%h2 Der User $user soll eingerichtet werden%/h
%fi
%if $form==“aendern”
%h2 Der User $user soll geändert werden%/h
%fi

‘usermod $form $user
%% Ende Beispiel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%isset(<varname>)
```

ist wahr, wenn <varname> definiert ist.

Beispiel:

```
%if %isset(var)
Var hat den Wert $var
%else
Var ist nich definiert.
%fi
```

`%image(<url>[, [align][, [breite][, höhe]]])`

Einfügen eines Bildes. align bestimmt die Ausrichtung, mögliche Werte sind left, right, top, middle und bottom. Breite und Höhe wird in Pixel oder % angegeben. Sollen die Breite und die Höhen angegbeben werden und align weggelassen werden, dann sieht der Aufruf folgendermaßen aus: %image(“URL des Bildes”, , 30, 20).

%% Beliebiger Text

Zeilen, die mit %% beginnen sind Kommentarzeilen

`%{...}`

Es können einfache Berechnungen mit %{ Formel } ausgeführt werden. Das gleiche gilt für die Bedingungen, die wie bei der Programmiersprache C behandelt werden. Operatoren, die unterstützt werden sind: ’+’, ’-’, ’*’, ’/’, ’%’, ’==’, ’!=’, ’<’, ’<=’, ’>’, ’>=’, ’&&’, ’&’, ’||’, ’|’, ’?’, ’:’, ’.’. Dabei ist ’.’ die Konkatenation von Strings und ’?’, ’:’ wie in C: %{ <Bedingung> ? <true-Wert> : <false-Wert> }

Wird eine u2w-Datei vom Browser mit Parametern in der Form: http://.../dat.u2w?par1=test&par2=hallo aufgerufen, so kann in der Datei mit $par1, $par2 darauf zugegriffen werden.

Menüsystem

Mit einfachen Mitteln ist es möglich, ein System von Menüs und Untermenüs aufzubauen. Dazu ist die Seite in drei Teile aufgeteilt. Im oberen Teil der Seite steht eine Überschrift, im linken Teil die Links und im rechten Teil die Formulare und Ergebnisse. Alle Links des linken Bereiches stellen das Ergebnis im rechten Bereich dar. Es stehen folgende Befehle zur Verfügung:

`%menuhead[:<size>] [<url>]`

Die folgenden Zeilen erzeugen den oberen Bereich der Menüseite. <size> bestimmt die Größe des Kopfbereiches in Prozent. Ist <url> angegeben, dann wird damit der Kopfbereich bestimmt.

`%menu[:<size>] [<url>]`

Die folgenden Zeilen erzeugen den linken Teil. Alle Hyperlinks in diesem Bereich haben als Target das rechte Fenster. <size> bestimmt die Breite des linekn Bereiches. Ist <url> angegeben, wird damit der linke Teil bestimmt. Im linken Teil gibt es noch die Befehle:

`%sub[:<r_url>[:<o_url>]] <umenü`$_1$`>[.<umenü`$_2$`>[.<umenü`$_3$`>...]] [ &l`$_1$`&T`$_1$`&[ &l`$_2$`&T`$_2$`&...]`

Untermenü mit dem Namen <umenü$_n$> und der Überschrift Beschreibung. Wird das Untermenü ausgewählt, wird ein Menü mit den Links link$_1$ und link$_2$ angezeigt. Weiterhin werden die Untermenüs %sub <name>.* angezeigt. Zusätzlich werden Buttons zum Zurückspringen und zum Sprung ins Hauptmenü erzeugt. Ist <r_url> angegeben, dann wird diese Seite im rechten Frame angezeigt. Ist <o_url> mit angegeben, dann wird <o_url> im oberen Frame dargestellt.

`%shell[:user] [<untermenü`$_1$`>[.<untermenü`$_2$`>...],] <shellaufruf>; Beschreibung`

Direkter Aufruf des Schellkommandos ohne den Umweg über eine eigene Datei. Die

Untermenüs müssen durch %sub definiert werden. Bei angegebenem [:user] wird der Prozess als User user ausgeführt. Entspricht dem Aufruf der Datei:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Beginn Beispiel
%setuser user        %% wenn angegeben
%h2 Beschreibung%/h
‘<shellaufruf>
%% Ende Beispiel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

`%subfile[:<r_url>[:<o_url>]] [<untermenü$_1$>[.<untermenü$_2$>...],] <linkurl>; Beschr`

Der Inhalt der Datei <linkurl> wird im linken Fenster dargestellt. Zusätzlich werden Buttons zum Zurücksprung und zum Sprung ins Hauptmenü erzeugt. In <linkurl> ist der Befehl %subfile erlaubt. <r_url> und <o_url> wie bei %sub.

`%link[:<target>] [<untermenü$_1$>[.<untermenü$_2$>...],] <linkurl>; Beschreibung`
`%exitmenu [<untermenü$_1$>[.<untermenü$_2$>...],] <linkurl>; Beschreibung`
`%newmenu [<untermenü$_1$>[.<untermenü$_2$>...],] <linkurl>; Beschreibung`
`%newwind [<untermenü$_1$>[.<untermenü$_2$>...],] <linkurl>; Beschreibung`

%link erzeugt Link, andere Schreibweise für &<linkurl>&Beschreibung& nach %sub [<untermenü$_1$>[.<untermenü$_2$>...],]. Ist <target> angegeben, Zeigt der Link auf das Fenster <target>. Der Inhalt der Datei <linkurl> wird bei %newwind in einem neuen Fenster dargestellt. Bei %exitmenu wird der aktuelle Frameset verlassen und bei %newmenu wird der Inhalt im gesamten Fenster dargestellt. %exitmenu verhält sich anders als %newmenu, wenn ein Bereich eines Framesets ein Frameset enthält. Die Befehle ohne Untermenüeintrag dürfen auf allen Seiten stehen.

`%end [<defseite>]`

Wenn <defseite> angegeben ist, dann wird <defseite> im rechten Teil angezeigt. Sonst bestimmt der Text nach %end den rechten Bereich beim Neuaufbau der Seite.

`%frame[:<rows>[:<cols>[:<url$_{f1}$>[:<url$_{f2}$>]]]]`
`...`
`%/frame`

Ergibt Frameset, es darf nur <rows> oder <cols> angegeben sein. <rows> ergibt zwei Teile übereinander, <cols> zwei Teile nebeneinander. <url$_{f1}$> bestimmt den linken/oberen und <url$_{f2}$> den rechten/unteren Teil des Framesets. Sind die <url$_{fn}$> nicht angegeben, dann bestimmt der Bereich zwischen %frame und %/frame den linken/oberen Bereich und alles nach %/frame den rechten/unteren Bereich. %setframe <name> <url> ändert den mit <name> angegebenen Frame in <url>.

`%include <dateiname>`

Die Datei <dateiname> wird eingefügt, als ob der Inhalt hier stehen würde. Bisher werden nur *.u2w Dateien unterstützt. Jede Datei wird unabhängig von der Endung als *.u2w Datei interpretiert. Es sind bis zu 10 verschachtelte %includes möglich. Mit %return kann man die eingefügte Datei sofort verlassen.

`%return`

Include Datei verlassen.

`%input(<shellaufruf>)`

Wie bei include, nur wird hier die Ausgabe des <shellaufrufs> als u2w-Datei Ausgewertet.

`%eval <Zeile>`

Die <Zeile> wird ausgewertet, es wird aber nichts ausgegeben. Z. B.: %eval ‘Shellaufruf‘, die

Ergebnisse des Shellaufrufes werden nicht ausgegeben.

`%exit[([<exitwert>])]`

u2w-Programm sofort verlassen. Beim Interpreter kann ein <exitwert> angegeben werden.

`%flush`

Daten aus dem Chunk-Puffer sofort an den Browser senden.

`%sslsubject`

Ausgabe des SSL-Subjects, wenn der Server mit HTTPS Unterstützung läuft und der Client einen gültigen SSL-Key verwendet.

`%sslissuer`

Ausgabe des SSL-Issuers, wenn der Server mit HTTPS Unterstützung läuft und der Client einen gültigen SSL-Key verwendet.

Befehle zur Dateibearbeitung

`%store(<parname>, <dateiname>)`

Der Inhalt des Parameters <parname> wird in die Datei <dateiname> gesichert. Das ist eine einfache Möglichkeit, um Rückgaben eines Textareafeldes zu sichern.

`%read(<dateiname>[, <parametername>])`

Datei einlesen. Ist <parametername> angegeben, dann wird der Inhalt der Datei im Parameter gespeichert, ansonsten wird der Inhalt der Datei ausgegeben.

`%delete(<dateiname>)`

Datei <dateiname> löschen.

`%open(<dateiname>)`

Datei <dateiname> zum Schreiben und Lesen öffnen. Der Rückgabewert ist ein Handle auf die Datei.

`%create(<dateiname>)`

Datei <dateiname> neu anlegen oder auf Länge Null kürzen und zum Schreiben öffnen. Der Rückgabewert ist ein Handle auf die Datei.

`%readline(<handle>)`

Eine Zeile aus der Datei, die mit %open geöffnet wurde, lesen.

`%print(<text>[, <handle>])`

Den <text> in die Datei schreiben, die mit %open oder %create geöffnet wurde. Ist <handle> nicht angegeben, dann auf STDOUT schreiben.

`%write(<parname>[, <handle>])`

Den Inhalt des Parameters <parname> in die Datei schreiben, die mit %open oder %create geöffnet wurde. Ist <handle> nicht angegeben, dann auf STDOUT schreiben.

`%eof(<handle>)`

Prüfen, ob noch Zielen aus der Datei gelesen werden können.

`%close(<handle>)`

Datei schliessen.

`%error(<text>)`

Den <text> auf dem Fehlerkanal ausgeben.

`%log(<text>)`

Den <text> ins Logfile schreiben (unix2webd muss mit -lp <Datei> aufgerufen werden).

`%rename(<alterName>, <neuerName>)`

Datei <alterName> in Datei <neuerName> umbenennen.

`%filesize(<Datei>)`

Größe einer Datei bestimmen.

Beispiel für Dateioperationen: `%let hout=%create(“test2.dat”)`
`%if $hout = ””`

```
Die Datei test2.dat konnte nicht geöffnet werden.
%exit
%fi
%let hin=%open(“test.dat”)
%if $hin != ””
  %while !%eof($hin)
    %let t=%readline($hin)
    %if $t ~ “test”
      %write(t, $hout)
    %else
      %print(Der Text $t passt nicht)
    %fi
  %/while
  %close($hin)
%else
Die Datei test.dat konnte nicht geöffnet werden.
%fi
```

Befehle für die Verbindung mit einer MySql-Datenbank

`%mysqlport(<port>)`

Ändern des Standardports für die Verbindung zur MySql-Datenbank.

`%mysqlconnect(<server>, <user>, <pwd>, <db>)`

Verbindung zur MySql-Datenbank aufbauen.

`%mysqlquery(<query>)`

Anfrage an die MySql-Datenbank. Rückgabewert kann bei Erfolg mit %if abgefragt werden.

`%mysqlgetline`

Die nächste Zeile der letzten %mysqlquery-Anfrage bereitstellen. Kann in einer %while Schleife eingesetzt werden, denn Rückgabewert ist FALSE, wenn keine Zeile mehr eingelesen werden kann.

`%mysqlreadline([<ssep>])`

die nächshte Zeile der letzten %mysqlquery-Anfrage ausgeben. Die Spalten werden mit <ssep> getrennt, wenn angegeben, sonst mit ’ ’.

`%mysqlline([<ssep>]`

Die Zeile, die mit %mysqlgetline bereitgestellt wurde ausgeben. die Spalten werden mit <ssep> getrennt, wenn angegeben, sonst mit ’ ’.

`%mysqlnumfields`

Anzahl der Spalten der letzten %mysqlquery bestimmen.

`%mysqlvalue(<s>)`

Den Wert der Spalte <s> der letzten mit %mysqlgetline bereitgestellten Zeile ausgeben. Die Spalten beginnen bei 0.

`%mysqlreadvalue`

Den nächsten Wert einer Zeile ausgeben.

`%mysqlisvalue`

True, wenn noch Werte mit %mysqlreadvalue ausgegeben werden können.

`%mysqlwrite(<query>)`

Anfrage an die MySql-Datenbank wie %mysqlquery, es wird kein Ergebnis erwartet.

`%mysqlid`

Primary Key des letzten mit “insert into” erzeugten Datensatzes.

`%mysqlrows`

Anzahl Datensätze, die beim letzten update geändert wurden.

`%mysqlread(<query>[, <ssep>[, <zsep>]])`

Eine mysql-Query ausführen und Ergebnis ausgeben. Dabei Spalten mit <ssep> trennen und Zeilen mit <zsep> trennen. Default für <ssep> und <zsep> ist ’ ’.

`%mysqlwert(<tabelle>, <spalte>, <id>)`

Ergibt Wert der Spalte <spalte> aus Tabelle <tabelle> mit ID=<id>. Voraussetzung ist, dass der

Primary-Key der Tabelle <tabelle> ID lautet.

`%mysqlins(<tabelle>, “f1=’w1’, f2=’w2’, ...”)`

Datensatz in Tabelle <tabelle> einfügen, wenn er noch nicht vorhanden ist. Der Rückgabewert ist ID des gefundenen Datensatzes oder ID des neu erzeugten Datensatzes.

`%mysqlenums(<tabelle>, <spalte>), %mysqlsets(<tabelle>, <spalte>)`

Ergibt die möglichen Werte der Enum/Set Spalte <spalte> in der Tabelle <tabelle>.

`%mysqlstore(<datei>, <tabelle>, <spalte>, <id>)`

Dateiinhalt in MySql-Datenbank speichern.

`%mysqltest(<query>)`

Query ausführen und nur Errorcode zurückliefern.

Anmerkung: Bisher müssen alle Befehle mit Ausnahme der Kommandos zur Veränderung der Schriftart und Größe und '%{' am Anfang der Zeile stehen. Sie dürfen allerdings mit Leerzeichen und Tabulatoren eingerückt sein.

Als Endungen für die Dateien ist neben u2w noch u3w, s2w und s3w möglich. u3w-Dateien sind HTML-Seiten mit eingebetteten Shell-Aufrufen.

s2w erzeugt im gegensatz zu u2w eine Textseite, also kein HTML-Format.

```
%%Beispiel für eine s2w Seite:
#uname -a
%%Ende Beispiel
```

s3w ergibt eine Binärseite, das macht Sinn, um die Ausgaben von Programmen direkt abzufragen.

```
%%Beispiel für eine s3w Datei:
#tar cf - /home
%%Ende Beispiel
```

Berechtigungsvergabe mit .access (access.uaw bei Windows)

Wenn in einem Verzeichnis die Datei .access (bei Windows access.uaw) gefunden wird, dann wird der Zugriff auf dieses Verzeicnnis nur den in der .access-Datei eingetragenen Usern gewährt. Das setzt eine Anmeldung über den Browser voraus. Dies kann über den Schalter -P beim Starten von unix2webd oder durch %authorize in den *.u2w Dateien erreicht werden. In der .access-Datei steht in jeder Zeile ein Username, dem der Zugriff auf das jeweilige Verzeichnis erlaubt ist.

Berechtigungsvergabe mit .passwd (passwd.uaw bei Windows)

Wenn unix2webd mit dem Parameter -s gestartet wird, dann werden die Anmeldeinformiationen aus der .passwd (bei Windows passwd.uaw) Datei für die Rechtevergabe verwendet. Aufbau der .passwd Datei:

```
[a][r][w] user_1:passwort_1 [setuser_1]
[a][r][w] user_2:passwort_2 [setuser_1]
...
```

Dabei ist $user_n$:$passwort_n$ jeweils mit base64 verschlüsselt eingetragen. Wenn unix2webd mit der Option WITH_SSL kompiliert ist, dann wird $user_n$:$passwort_n$ mit base64 verschlüsselt und zusätzlich wird die MD5-Summe gebildet, damit das Kennwort nicht ausgelesen werden kann. Die Flags a, r und w:

a: Der User darf über ein Webinterface die .passwd und .hosts Datei verändern.
r: Der User darf in diesem Verzeichnis lesen.
w: Der User darf in diesem Verzeichnis schreiben.

[$setuser_a$], [$setuser_b$] sind User des Servers, auf den der Prozess nach erfolgter Anmeldung mit user und pwd wechselt. Ist [$setuser_n$] nicht angegeben, dann wechselt der Prozess auf den Standarduser, der über -u <stduser> angeben wird. Ist [$setuser_n$] gleich "-", dann wechselt der User nicht. Ist der Server mit der Option -sg <path> gestartet, dann wird die globale Passwort-Datei für Verzeichnisse verwendet, in denen keine .passwd-Datei gefunden wird. Steht in einer .passwd-Datei als User ein +, dann wird die Globale Passwort-Datei zusätzlich eingelesen. Ein - als User setzt die Einstellungen für einen nicht angemeldeten Benutzer. Das hat allerdings zur Folge, dass der Browser kein Fenster zur Passwortabfrage generiert. Um sich als User anzumelden muss man von einem anderen Verzeichnis kommen, in dem es keinen Eintrag mit "-" gibt oder in der ersten Zeile der *.u2w Datei muss %authorize stehen, damit nach einem User und Passwort gefragt wird. Der unix2webd-Server kann auch mit -P gestartet werden, damit man sich immer anmelden muss. Um die .passwd Datei zu verändern, muß der User das a Flag besitzen. Um die .passwd Datei zu verändern, gibt man folgende URL an: http://host:port/verzeichnis/.passwd?&access für die .passwd-Datei eines Verzeichnisses oder

http://host:port/.passwd?&global für die globale .passwd-Datei.

Berechtigungsvergabe mit .hosts (hosts.uaw bei Windows)

Wenn in einem Verzeichnis die Datei .hosts steht, dann dürfen nur Rechner, deren IP-Adresse in der .hosts verzeichnet ist, auf dieses Verzeichnis zugreifen. Aufbau der .hosts:

```
[a][r][w] IP-Adresse1
[a][r][w] IP-Adresse2
...
```

Die Flags a, r und w haben die gleiche Bedeutung wie bei .passwd. Wenn .passwd und .hosts vorhanden ist, dann müssen die Rechte in beiden Dateien gewährt werden. Als IP-Adresse kann auch ein Netz in der Form Netzwerk/Bits z. B. 10.1.0.0/21 angegeben werden. Ein ”-” als IP-Adresse steht für alle Hosts. Die .hosts wird von oben nach unten durchsucht und wenn ein passender Eintrag gefunden wird, dann werden die Rechte gesetzt. Wenn ein Netzwerk angegeben ist, dann müssen alle Hosts, für die andere Einstellungen gemacht werden sollen, vor dem Netzwerkeintrag stehen. Ein Eintrag mit ”-” macht nur als letzten Eintrag Sinn. Die URL für die Änderung der .hosts: http://host:port/verzeichnis/.hosts?&access für die .hosts eines Verzeichnisses oder ...?&global für die globale .hosts.

Einsatz von unix2web als Server für Systemsicherungen

für die Sicherung von Systemen mit unix2web gibt es zwei Möglichkeiten:

1. Auf dem Backupserver (z. B. der Networker) läuft unix2webd und die zu sichernden Daten werden mit httpput an den Server geschickt. Dazu wird auf der Server unix2webd so konfiguriert, dass ein bestimmter Rechner über einen bestimmten Port auf ein Verzeichnis zugreifen darf. Beispiel:

   Start von unix2webd mit folgenden Optionen:

   unix2webd -D -h <pfad> -u <user> -P -sf -g -of -p <port> -t <timeout>

   Die Parameter im einzelnen:

   -D : Programm als Daemon starten
   -h <pfad> : Pfad, unter dem die Sicherungsverzeichnisse liegen
   -u <user> : User, unter dem unix2webd läuft. Der User muß Schreib- und Leserechte im Sicherungsverzeichnis haben.
   -P : Authentifizierung erforderlich,
   -sf : die Daten stehen in der .passwd Datei.
   -g : Uploads mit PUT erlaubt.
   -of : Der Zugriff muß über .hosts geregelt sein.
   -p <port> : Port für die Verbindung
   -t <timeout>: Timeout sollte auf mindestens 60 s gesetzt werden. Sicherungsbefehle wie dump durchsuchen als erstes die Verzeichnisse nach zu sichernden Datein, das kann teilweise recht lange dauern. Der Server unterbricht die Verbindung nach <timeout> s.

   unter dem <pfad> wird dann für jeden zu sichernden Server ein Pfad eingerichtet: <pfad>/<rechner-pfad>. In dieses Verzeichnis müssen zwei Dateien eingerichtet werden:

   .hosts mit folgedem Inhalt:

   ```
   rw <ip-adresse-des-hosts>
   ra <ip-adresse-des-adminrechners>*
   ```

   .passwd mit folgendem Inhalt:

   ```
   rw <base64-verschlüsselter-username:kennwort>
   ra <base64-verschlüsselter-username:kennwort-des-admins>*
   ```

   * Die Einträge für den Administrator können auch entfallen.

   Dann kann mit httpput eine Sicherung auf diesen Server erfolgen:

   ```
   httpput -u <username> -p <kennwort> -d /<rechner-pfad> <Sicherungshost>:<port>
           <zu-sichernde-Dateien ...>
   ```

   oder

   ```
   dump <parameter> | httpput -u <username> -p <kennwort> -f
                                /<rechner-pfad>/<dateiname-der-Sicherung>
                                <Sicherungshost>:<port>
   ```

2. Auf dem zu sichernden Rechner läuft unix2webd. Wenn ein Rechner hinter einer Firewall steht, ist das die einzige Möglichkeit, denn es sind im allgemeinen nur Verbindungen erlaubt, die zu diesem Rechner hin aufgebaut werden. Mit unix2web können dann Dateien gesichert weden, die von einem anderen Programm angelegt wurden oder es können Archive von unix2webd erzeugt werden. Dazu müssen Programme installiert sein, die das Archiv auf die Standartausgabe ausgeben. Unter Unix kann das z. B. mit dump oder tar geschehen. Bei Windows Betriebssystemen wird man vor der Sicherung eine Datei anlegen, die dann mit unix2web abgezogen wird. Unter Win 2000 kann ntbackup in einen

Datei sichern. Beispiel:

Auf dem zu sichernden Rechner wird unix2webd mit folgenden Parametern gestartet:

```
unix2webd -D -h <pfad> -u <user> -P -sf -of -p <port>
```

Die Parameter im einzelnen:

-h <pfad> : Pfad, unter dem die zu Sichernden Dateien liegen oder der Pfad, unter dem die *.s3w Befehle liegen.
-u <user> : User, unter dem unix2webd läuft. Der User muß Leserechte für die zu sichernden Daten haben.
-P : Authentifizierung erforderlich, -sf: die Daten stehen in der .passwd Datei.
-of : Der Zugriff muß über .hosts geregelt sein.
-p <port> : Port für die Verbindung

unter dem <pfad> wird dann für jeden zu sichernden Server ein Pfad eingerichtet: <pfad>/<rechner-pfad>. In dieses Verzeichnis müssen zwei Dateien eingerichtet werden: .hosts mit folgedem Inhalt:

```
r <ip-adresse-des-hosts>
```

.passwd mit folgendem Inhalt:

```
r <base64-verschlüsselter-username:kennwort>
```

Wenn die Daten zurückgesichert werden sollen, dann muss in der .hosts und der .passwd das Merkmal rw stehen. Das kann auch dauerhaft so eingestellt sein. Zusätzlcih muss dann der Server unix2webd mit dem Parameter -g gestartet werden.

Die Sicherungen werden dann mit httpget abgezogen:

httpget -f <dateiname> -u <user> -p <kennwort> [-P <proxy:proxport>]
<sicherungsdatei>|<sicher.s3w>

-f <dateiname> : Die Sicherung wird als <dateiname> abgelegt
-u <user> : Username aus der .passwd
-p <kennwort> : Kennwort aus der .passwd
-P <proxy:proxport>: bei Verwendung eines Porxyservers
<sicherungsdatei> : Dateiname der Sicherung
<sicher.s3w> : s3w-Skript auf dem Server, das die Sicherung erstellt

Aussehen einer sicher.s3w:

```
%% s3w Skript zur Sicherung eines Verzeichnisses mit tar
#tar cf - /zu/sicherndes/verzeichnis | gzip -1 -c
%% Ende s3w Skript
```