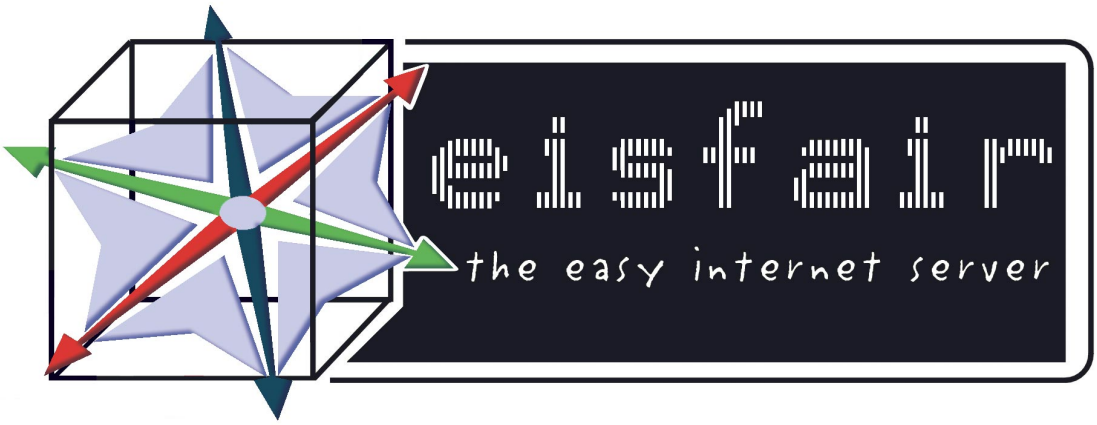

eisfair Entwickler-Dokumentation

Das eisfair Dokumentations-Team

news:spline.eisfair.dev

Letzte Änderung am 3. Oktober 2016

Inhaltsverzeichnis

1 Charta für Paketentwickler

1.1 Vorwort

Als Entwickler eines *eisfair* Paketes sollte man einige Grundregeln beachten, um den Ideen und der Philosophie des eisfair-Projektes gerecht zu werden. Dazu bietet sich als erstes die Vorstellung von *eisfair* aus dem Oktober 2002 an:

> *eisfair* ist ein einfach zu installierender Internet-Server, der als zugrundeliegendes Betriebssystem Linux verwendet. Es wird ausschliesslich freie Software verwendet. Die Installation und der Betrieb von *eisfair* setzen jedoch keine Linux-Kenntnisse voraus und ist mit einfachen und – besonders wichtig – einheitlichen Mitteln möglich.
>
> *eisfair* soll keine weitere Linux-Distribution sein, davon gibt es schon viel zu viele. Intention ist die Installation eines Internet-Servers, bei dem die angebotenen Applikationen und Dienste im Vordergrund stehen sollen – nicht das Betriebssystem.
>
> Ein durchschnittlicher Anwender wird durch die enorme Funktionsvielfalt einer modernen Linux-Installation (wie z.B. SuSE) regelrecht erschlagen und ist schon bei der Auswahl der nötigen Pakete schlichtweg überfordert. Genau hier soll *eisfair* den Anwender unterstützen. Die für Linux/Unix verfügbaren Applikationen zeichnen sich dadurch aus, dass sie einerseits sehr ausgereift sind und stabil laufen, andererseits jedoch alle verschieden zu installieren und zu konfigurieren sind. Hier wurde für jedes Software-Paket ein eigenes Süppchen gekocht. Die für *eisfair* angebotenen Anwendungen verfügen über eine einheitliche Konfigurationsschnittstelle, so dass sich der Aufwand für die Einarbeitung/Konfiguration auf ein Minimum reduzieren lässt.
>
> Um dies mit den verfügbaren Applikationen für Linux zu realisieren, wurde in die Konfigurationsschnittstelle zwischen *eisfair* Systemverwalter und den jeweiligen Applikationen eine Schicht eingefügt, welche eine typische *eisfair* Konfiguration in die applikationsspezifischen Konfigurationen der jeweiligen Dienste (z.B. eines Mailservers) umsetzt. Damit wird der Einsatz von gängigen Internet-Diensten möglich, ohne dass sich der Verwalter eines *eisfair* Systems mit der (übrigens auch sehr heterogenen) Dokumentation eines konkreten Programms auseinandersetzen muss. Dass eventuell dabei Abstriche bei speziellen Anforderungen gemacht werden müssen, weil nicht jedes Feature eines Programms allgemein umgesetzt werden kann oder wird, ist ein Nachteil, der jedoch hinnehmbar ist, wenn die Funktionalität einer typischen Serverumgebung ausreichend ist.
>
> Die *eisfair* Konfiguration wird auf ASCII-Dateien abgebildet, die einem einheitlichen Format unterliegen. Sie entsprechen im wesentlichen einer Konfigurationsdatei, wie sie im *fli4l* Projekt verwendet wird. Eine graphische Benutzeroberfläche ist in Entwicklung. Mit dieser wird man zukünftig die *eisfair* Applikationen über ein Web-Interface konfigurieren können. Nichtsdestotrotz ist jederzeit die Bearbeitung der ASCII-Dateien mit einem normalen Editor möglich.

(Text gekürzt, der vollständige Text kann der eisfair *Benutzerdokumantation entnommen werden)*

1.2 Welche Software ist für *eisfair* geeignet?

Zunächst sollte man sich überlegen, ob das geplante *eisfair* Paket überhaupt zum *eisfair* - Konzept passt, d.h. mit diesem harmoniert. Ist das der Fall, müssen einige Überlegungen angestellt werden, wie das Paket für *eisfair* portiert werden kann.

Da *eisfair* selbst keine weitere Linux-Distribution sein soll, sondern explizit als Serversystem ausgelegt ist, eignen sich grundsätzlich die für Linux verfügbaren Server-Applikationen. Klassische Client-Software wie ein Office-Paket (z.B. OpenOffice) oder ein Webbrowser (z.B. FireFox) scheiden daher aus. Auch X11 sowie Pakete wie KDE oder Gnome sind primär als Client-Userinterface ausgelegt und eignen sich somit nicht als *eisfair* Pakete.

Ebenso sind Pakete wie ein komplettes LAMP („Linux, Apache, MySQL, PHP“) nicht als Ganzes für eisfair geeignet. *eisfair* ist modular aufgebaut, daher sollten solche Applikationen auch als eigene Programmpakete angeboten werden. Tatsächlich existieren für *eisfair* die oben aufgeführten Pakete bereits seit langer Zeit. Eine Kombination der Einzelpakete Apache, MySQL und PHP ist hier viel flexibler und leichter zu beherrschen.

Da *eisfair* selbst komplett auf freier Software basiert, passen sicher auch freie Applikationen am besten zu *eisfair*. Bei der Verwendung von nicht-freien Applikationen in einem Paket muss unbedingt darauf geachtet werden, dass der Inhaber der Rechte an dieser Software eine Weiterverbreitung als *eisfair* Paket auch gestattet. Zusätzlich muss die Frage geklärt werden, wie mit einer eventuellen Einzellizensierung umzugehen ist.

1.3 Portierung einer Server-Applikation auf *eisfair*

Die wichtigste Regel, die bei der Umsetzung eines Paketes zu beachten ist: die Installation, Konfiguration und die Bedienung der Applikation muss möglichst einfach sein. Dies drückt insbesondere der erste Buchstabe von *eisfair* aus: **„EASY - für den Anwender!“**. Dabei sollte man auf exotische Konfigurationen, welche die Server-Anwendung leisten könnte, aber die Konfiguration des Paketes verkomplizieren würde, besser verzichten. Wenn der Grossteil der potentiellen Anwender mit dem Standardumfang arbeiten kann, sollte man solche „Schmankerl“ weglassen. Es sollte also nicht unbedingt jede erdenkliche Funktion der Basissoftware unterstützt werden. Im Zweifelsfall geht eine einfache Konfiguration vor. Anhand des Feedbacks der User lässt sich im Laufe der Zeit viel besser feststellen, welche Features tatsächlich benötigt werden. Diese Features konnen dann Schritt für Schritt in das Paket eingearbeitet werden und lassen zudem das Leben des Projektes erkennen.

Einfache und übersichtliche Konfigurationen sind die Grundvoraussetzungen für ein *eisfair* Paket. Dabei muss unbedingt die *eisfair* Konfigurationsschicht beachtet werden. Diese darf unter keinen Umständen umgangen werden. Das Paket muss sich möglichst nahtlos ins Gesamtsystem integrieren. Untergräbt man als Entwickler die Philosophie von eisfair, wird die ganze Idee, die hinter *eisfair* steckt, hinfällig. In diesem Fall ist für den Paketentwickler eine Standard-Linux-Distribution besser geeignet.

Neben der Konfiguration sollte man auch der Installation des *eisfair* Paketes ein grösseres Augenmerk widmen. Die Installation sollte möglichst einfach, geradlinig und performant

über die Bühne gehen. Es sollte auch auf jeden Fall vermieden werden, einen Reboot des *eisfair* Servers zu erzwingen. Ein Server zeichnet sich durch seine Verfügbarkeit aus. Ist ein Reboot des Servers notwendig, um die Software zu initialisieren, wird diese Grundregel verletzt. Ausnahmen kann es natürlich geben, sie sollten jedoch vermieden werden.

Wichtig ist auch die Möglichkeit, ein Update eines Paketes installieren zu können. Dabei muss auf Abwärtskompatibilität geachtet werden. Bisherige Konfigurationen sollten bei einem Package-Update soweit wie möglich übernommen werden. Das beinhaltet auch die korrekte Handhabung mglw. entfallener oder umbenannter Konfigurationsparameter. Das Paket muss ebenso vom Anwender beliebig oft installiert werden können, ohne dass sich dabei plötzlich Dubletten von Daten/Konfigurationen einstellen. Der Paketentwickler muss also jederzeit damit rechnen, dass der Anwender das Paket im Falle einer Fehlfunktion „drüberinstalliert“. Installiert werden die Pakete für *eisfair* grundsätzlich über das *eisfair* Setup-Menü. Eine davon abweichende Installation ist nicht erlaubt!

Genauso wichtig wie die Entwicklungsarbeit, die man in die Portierung eines *eisfair* Paketes steckt, ist auch die Erstellung einer entsprechenden Dokumentation. Diese sollte man nicht vernachlässigen. Eine der wichtigsten Säulen von *eisfair* ist die Dokumentation. Man sollte mindestens die Zeit, die für die Portierung der Applikation auf *eisfair* benötigt wurde, auch in die Erstellung der Dokumentation stecken. Je besser das Handbuch ist, desto grösser wird die Akzeptanz der Anwender.

1.4 QUALITÄT - Pflichten eines Paketautors

Die Einhaltung der in diesem Dokument aufgeführten Richtlinien dient der Sicherstellung eines gewissen Qualitätsniveaus. Die Stabilität und die Qualität von *eisfair* steht und fällt mit der Disziplin des Paketautors.

Mit der Veröffentlichung eines *eisfair* Paketes endet jedoch nicht die Tätigkeit des Entwicklers. Dieser sollte auch gegenüber den Anwendern eine gewisse Verantwortung zeigen. Dazu gehört neben der kontinuierlichen Pflege des Paketes auch die Bereitschaft, Support zu bieten. Sollte das die Möglichkeiten des Entwicklers überschreiten, sollte sich dieser nach Personen umsehen, die bereit sind, entsprechenden Support zu leisten. Nichts ist für einen Anwender schlimmer, als wenn er auf sich alleine gestellt ist.

Zur Pflege des Paketes gehören selbstverständlich auch Bugfixes. *eisfair* soll nicht nur „Easy“ sein, sondern auch sicher. Werden Sicherheitslücken in der Software bekannt, sollte der Entwickler diese schliessen und ein Update zur Verfügung stellen. Es ist niemandem gedient, einen Server zu betreiben, der Löcher wie ein Schweizer Käse hat.

1.5 Urheberrechte

Der Entwickler hat neben seinen Pflichten natürlich auch Rechte - nämlich zumindest die Urheberrechte an seiner Portierung. Da *eisfair* als Plattform grundsätzlich auf der Basis von freier Software steht, sollte man - wenn möglich - auch sein Paket als freie Software deklarieren, d.h. unter die GNU PUBLIC LICENSE (siehe http://www.gnu.org/), kurz „GPL“, stellen.

Am besten deklariert man dies in sämtlichen selbst erstellten Klartext-Dateien (Konfigurationen, Shell-Scripts usw.) durch einen entsprechenden Absatz als Kommentar. Beispiele dafür findet man unter eisfair zuhauf, z.B. in den Konfigurationsdateien unter `/etc/config.d`.

Ist es aus urheberrechtlichen Gründen nicht möglich, das Paket unter die GPL zu stellen, sollte man dies auch klarstellen. Das sollte am besten bereits bei der Installation geschehen. Viele Anwender legen Wert darauf, ausschliesslich freie Software zu verwenden. Dieses Anliegen sollte man als Paketentwickler auf jeden Fall respektieren.

So, genug der Vorrede, jetzt geht es ans Eingemachte: Viel Spass bei der Lektüre dieser Dokumentation und dann ... beim Entwickeln :-)

Frank Meyer, im Juni 2005

Überarbeitet, aktualisiert und erweitert vom eisfair-Team 2011

2 Grundlagen der Entwicklung unter *eisfair*

Am *eisfair* Projekt arbeiten viele unterschiedliche Entwickler zusammen. Um dennoch ein möglichst reibungsloses Zusammenspiel der einzelnen Pakete und Komponenten zu gewähren, wurden einige Werkzeuge erstellt, aber auch Standards und Regeln abgesprochen und definiert.

Dabei sind die Richtlinien der GPL Lizenz unbedingt einzuhalten. Das bedeutet, eigenentwickelte Programme oder Modifikationen an bekannten Paketen müssen auch im Sourcecode an einer allgemeinzugängigen Stelle einsehbar sein.

2.1 Die Plattform *eisfair*

Sie unterliegt zwar einer ständigen Weiterentwicklung, dennoch werden einige Grundsätze relevant bleiben. Der Schwerpunkt des Einsatzes liegt in der Verwendung als Serverplattform. Dabei ist es egal, ob es sich um einen kleinen energiesparenden Homeserver oder um eine grosse Datenbankplattform für ein mittelständisches Unternehmen handelt. Die schnelle Installation von kompletten Lösungen und die einfache menügestützte Administration sind wichtige Hervorhebungsmerkmale. Dabei ist immer die Stabilität der Plattform der entscheidende Punkt, dem sich alle funktionalen Erweiterungen unterordnen. Eventuell nötige Eingriffe in systemrelevante Komponenten und Konfigurationsdateien müssen daher immer in Absprache mit dem Entwickler-Team erfolgen. Für die Umsetzung werden dann möglichst transparente Lösungen oder allgemeingültige Schnittstellen zum Einsatz kommen. Ein gutes Beispiel hierfür ist die Datei `/etc/services`, welche durch Aufrufen des Scripts `update-services` *$package* erweitert werden kann.

Bei der Entwicklung von *eisfair* wird darauf geachtet, dass das System schlank und schnell bleibt. Übermässige Verzeichnisbaum-Erweiterungen (`/opt`, `/srv`, `/usr/local/sbin`) werden vermieden. Auch sollten Binaries, wenn sie nicht paketübergreifend verwendet werden, nach Möglichkeit unterhalb des Verzeichnisses `/usr/local/`*$package*`/*` abgelegt werden. Von einzelnen wichtigen Dateien kann man dann immer noch Symlinks in das Verzeichnis `/usr/bin` legen. Das verbessert die Übersichtlichkeit, vereinfacht die Deinstallation, erhöht die Sicherheit und bremst das System beim Durchsuchen der ausführbaren Verzeichnisse nicht aus.

Auf der *eisfair* Plattform wird man keine man-Page-Verzeichnisse finden. Das ist Absicht. Der *eisfair* Administrator/Benutzer kommuniziert mit einer speziellen Konfigurationsschicht, deren Parameter und Wirkungsweise in der jeweiligen Paket-Dokumentation verzeichnet sind.

eisfair verwendet zum gegenwärtigen Zeitpunkt die glibc-2.15 Bibliothek. Deshalb werden

aktuelle Pakete anderer Linux-Distributionen nicht auf *eisfair* funktionieren. Wer Software für *eisfair* entwickeln möchte, nutzt dazu am besten die Entwicklungsumgebung, siehe [developer].

2.2 Die Entwicklungsumgebung

Sie ist in dem Paket „Development Environment for *eisfair* “ [developer] zusammengefasst. Durch die Installation besitzt man mit dem GCC-Compiler, den Binutils, den Devtools und dem glibc-Paket eine gute Entwicklungsplattform. Daneben steht bspw. noch Perl zur Verfügung und bei Bedarf kann mit dem kernel-dev Paket der aktuelle Sourcecode für den Kernel hinzugezogen werden.

Beim Kompilieren wird man mitunter feststellen, dass die Umgebung doch nicht ausreicht. Der eine oder andere Header fehlt einem immer mal wieder. Jetzt könnte man sich auf die Suche nach dem Sourcecode begeben und die fehlende Komponente mal eben selber anpassen und kompilieren. Vorher ist allerdings ein Blick in die „devel“-Kategorie von Pack-Eis [pack-eis] angebracht. Sehr viele Komponenten sind dort, genau wie das glibc-Paket, in einer Developer-Version vorhanden. Diese sollte man aus Kompatibilitätsgründen unbedingt verwenden. Warum das so empfohlen wird, wird schnell klar: Ein Blick in die optionalen Parameter bei der Erstellung so mancher Komponenten zeigt eine wahre Flut von Möglichkeiten der Erstellung auf. Diese können wichtig sein oder aber zu einem nicht funktionsfähigen System führen.

Deshalb gibt es von jedem Library Paket eine Developer Version. Sie besteht aus dem Paketnamen, welcher mit dem Library-Paket identisch ist, erweitert durch den Zusatz „`-dev`“. Also z.B.:

Library Pakete:

```
libdb
libssl
```

Developer Pakete:

```
libdb-dev
libssl-dev
```

Wie man sieht, sollten Library Pakete auch durchaus Versionsinformationen im Namen enthalten. Die meisten Programme benötigen von einer Bibliothek eine ganz bestimmte Version und haben mit neueren oder älteren Fassungen Probleme.

Hinweis

Library Developer Pakete enthalten:
- header files
- object files
- static libraries
- install script
- deinstall script
- package file with `<require-lib>`-tag

Vor der Kompilierung von Paketen ist besonders auf die Verzeichnis-Parametrierung zu achten. Diese ist oft mit Werten vorbelegt, die auch bei einer unachtsamen Paketerstellung niemandem schaden. Das heisst, alles wird unter dem Verzeichnis `/usr/local` abgelegt. So landen dann aber auch Konfigurationsparameter und Bibliotheken an Stellen, wo sie *eisfair* konform nicht verwendet werden können. Vielfach wird man also um das übergeben folgender Konfigurationsparameter nicht herumkommen.

Beispiel:

```
./configure --prefix=/usr                \
            --sysconfdir=/etc            \
            --localstatedir=/var/lib     \
            --libdir=/usr/lib            \
            --libexecdir=/usr/lib/$package
```

Der Make Befehl ermöglicht oft die Angabe eines Parameters, welcher die *eisfair* Paketerstellung erleichtert.

```
make DESTDIR=/tmp/$package install
```

Alle Dateien werden so in die richtigen Unterverzeichnisse einsortiert, allerdings nicht in das laufende Betriebsystem sondern unter das angegebene `/tmp/`*$package* Verzeichnis. Mit dem Strip Befehl können dann abschliessend alle eingebetteten Kommentare und Anmerkungen aus den fertigen Dateien entfernt werden.

Beispiel:

```
strip -R .note -R .comment /tmp/$package/usr/bin/*
strip -R .note -R .comment /tmp/$package/usr/lib/*
strip -R .note -R .comment /tmp/$package/usr/lib/$package/*
```

3 Überblick über ein *eisfair* Paket

3.1 Grundsätzliche Konventionen

3.1.1 Auswahl des Paketnamens

Der erste Schritt bei der Paketerstellung ist die Wahl des richtigen Paketnamens. Der Paketname wird an sehr vielen Stellen verwendet. Er ist zum Beispiel Bestandteil jeder Konfigurationsvariable eines Pakets und auch Teil des Namens vieler im Paket befindlicher Dateien.

Es empfiehlt sich daher, einen **kurzen** Namen für das Paket zu wählen. Es versteht sich von selbst, dass „dhcpd“ dem Namen „dynamic_host_configuration_protocol_daemon“ vorzuziehen ist. Dem Anwender wird an den meisten Stellen sowieso nicht der Paketname, sondern ein anderer Text angezeigt.

Generell **verboten** ist die Aufnahme der **Versionsnummer** in den Paketnamen. Ein Paket darf also nicht „foo_1.1.2“ heissen, sondern muss den Namen „foo“ bekommen.

Als Zeichen des Paketnamens sind ausschliesslich Kleinbuchstaben, Zahlen, der Bindestrich „–“ („Dash“) und der Unterstrich „_“ („Underscore“) zulässig.

Zudem sollte das Paket nicht den Namen eines anderen Pakets als Präfix getrennt mittels eines Bindestrichs „–“ enthalten. Dies kann zu Probleme mit manchen Deinstallationsroutinen führen. Aus diesem Grund sollte generell auf die Verwendung des Bindestrichs „–“ im Paketnamen verzichtet und als Alternative besser der Unterstrich „_“ verwendet werden.

Eine der wenigen Ausnahmen davon sind Treiberpakete („scsi-buslogic“), Entwicklerbibliotheken („glibc-dev“) und Bibliotheken („libz1-2“).

3.1.2 Dateinamen, Variablennamen

Viele Dateinamen, Verzeichnisnamen und Variablennamen leiten sich vom Paketnamen ab. Im Paket foo gibt es also eine Reihe Dateien, die den Paketnamen foo als Namensbestandteil haben. Beispiele dafür sind die Dateien `/etc/check.d/foo.ext` und `/var/install/config.d/foo-update.sh` sowie die Variablen `START_FOO` und `FOO_BAR`.

Dateinamen werden dabei **immer** in Kleinbuchstaben geschrieben, Variablennamen in der Konfigurationsdatei **immer** in Grossbuchstaben.

Im folgenden ist der Paketname immer als *$package* angegeben. Ist zum Beispiel eine Datei `/usr/local/`*$package*`/`*$package*`-action.sh` angegeben, entspricht das im Paket

„foo“ der Datei `/usr/local/foo/foo-action.sh`, eine Variable `START_FOO` wird als `START_`*$PACKAGE* geschrieben.

3.2 Bestandteile eines Pakets

Ein Installationspaket für *eisfair* ist im wesentlichen ein mit gzip oder bzip2 komprimiertes Tar-Archiv, in dem neben den eigentlichen Dateien des Paketes noch einige weitere Informationen und Skripte enthalten sein müssen.

Das Paket selbst spiegelt den Dateibaum ab dem root-Verzeichnis, also „/“ wieder. Eine zu installierende Datei `/usr/bin/foo` muss also im Tar-Archiv als `usr/bin/foo` (ohne absoluten Pfad, also relativ!) abgespeichert sein.

Datei	Beschreibung
`/etc/check.d/`*$package*	Die Konfiguration (Seite 44) muss mittels Eischk (Seite 51) auf Gültigkeit geprüft werden.
`/etc/check.d/`*$package*`.exp`	Paketspezifische Wertdefinitionen für Eischk (Seite 51).
`/etc/check.d/`*$package*`.ext`	Individuelle Prüfungen für Eischk (Seite 51).
`/etc/config.d/`*$package*	Die aktuelle Konfiguration (Seite 44) ist die wichtigste Datei der *eisfair*–Konfigurationsschicht. Diese Datei wird nicht im Paket mitgeliefert, sondern während der Installation (Seite 25) angelegt.
`/etc/backup.d/`*$package*`*`	Advanced Configuration File Handling (Seite 49) legt Backups der Konfiguration an. Diese Dateien dürfen **nicht** mit dem Paket ausgeliefert werden.
`/etc/default.d/`*$package*	Die Default-Konfiguration (Seite 44) ist für das Advanced Configuration File Handling (Seite 49) und Webconf zwingend notwendig und muss jedem Paket beiliegen.
`/etc/filelist.d/`*$package*`-files.txt`	Liste aller Dateien (Seite 36) des Paketes mit den entsprechenden Zugrifssrechten.
`/etc/init.d/`*$package*	Mittels des Init-Scripts (Seite 40) wird das Paket gesteuert (z.B. Start, Stop).
`/etc/rc2.d/Sxx`*$package*	Start des Pakets beim Systemstart (Seite 40).
`/etc/rc2.d/Kxx`*$package*	Beenden des Pakets beim Shutdown (Seite 40).
`/tmp/install.sh`	Mittels `install.sh` wird die Installation (Seite 25) eines Pakets durchgeführt.

Datei	Beschreibung
`/tmp/preinstall.sh`	Das Skript `preinstall.sh` dient zur Vorbereitung der Installation (Seite 25) eines Pakets.
`/usr/share/doc/`*$package*`/`*$package*`.txt`	Eine Dokumentation (Seite 76) ist jedem Paket mitzuliefern.
`/usr/share/doc/`*$package*`/changes.txt`	In den Release Notes (Seite 76) wird die Historie der Änderungen an einem Paket gepflegt.
`/var/install/bin/`*$package*`-action`	Diverse paketspezifische Verwaltungsscripts.
`/var/install/config.d/`*$package*`.sh`	Das Apply–Script dient zur Verarbeitung einer neuen Konfiguration (Seite 48)
`/var/install/config.d/`*$package*`-update.sh`	Script zum übernehmen einer Paketkonfiguration beim Update (Seite 28) eines Pakets.
`/var/install/deinstall/`*$package*	Über die Deinstallationsroutine (Seite 37) kann ein Paket wieder deinstalliert werden.
`/var/install/help/`*$package*	In der Hilfedatei (Seite 47) werden die einzelnen Konfigurationsvariablen näher beschrieben.
`/var/install/menu/setup.services.`*$package*`.menu`	Hier wird die Menüstruktur (Seite 68) für das *eisfair* Setup-Menü definiert.
`/var/install/packages/`*$package*	In der Paket–Info–Datei (Seite 15) werden die grundlegenden Daten zu einem Paket angegeben.

Die einzelnen Bestandteile werden in den entsprechenden Kapiteln dieser Entwicklerdokumentation näher erklärt.

Explizit nicht in ein Eisfair-Paket dürfen:

- info und man Dateien unter `/usr/info` bzw. `/usr/man`
- locale Dateien unter `/usr/share/locale` ausgenommen sind de | en | fr
- eine eigene glibc
- Bibliotheken
- Links
- leere Verzeichnisse

Links und leere Verzeichnisse müssen im Installationsskript angelegt werden.

Nur in **Ausnahmefällen** und nach vorheriger Abstimmung mit dem *eisfair* Entwicklerteam dürfen in ein Paket:

- ein eigener Kernel (nur wenn unumgänglich)
- Kernel-Module (bitte nachfragen, ob diese nicht besser in das offizielle Kernelpaket mit aufgenommen werden können)

Um Probleme mit den Benutzerrechten (zum Beispiel Dateien, die einem User mit der ID 501 gehören, der auf dem *eisfair* -Server gar nicht existiert) zu verhindern, empfiehlt es sich, Packages mit

```
tar -czv --owner=root --group=root -f package.tar.gz *
```

zu verpacken. Damit gehören die entpackten Dateien zunächst dem Superuser **root**. Dieser Zustand kann, sofern gewünscht, im Installationsskript `/tmp/install.sh` geändert werden.

4 Die Paket-Info-Datei

4.1 Hintergrund der Paket-Info-Datei

Diese Datei soll eine bessere Übersicht über die verfügbaren Pakete und notwendige Verwaltungsdaten bereitstellen.

Sie wird auch dazu verwendet, um das Paket in einer eis-list.txt aufzulisten.

4.2 Namen und Pfad

Die Info-Datei wird mit jedem Paket zweimal ausgeliefert: Sie ist erstens im Paket enthalten und liegt im Verzeichnis `/var/install/packages/` mit dem Paketnamen als Name und wird zusätzlich parallel zum eigentlichen Paket ausgeliefert. Dabei erhält sie die Dateiendung `.info`.

Beispiel:
Wenn der Pfad zum Paket `mail.tar.gz` bspw. `http://www.meine-domain.de/eisfair/` lautet (also die URL das Paketes `http://www.domain.de/eisfair/mail.tar.gz` ist), dann heisst die zugehörige Info-Datei `mail.tar.gz.info`; deren URL lautet somit `http://www.domain.de/eisfair/mail.tar.gz.info`.

Die Datei ist zudem als `/var/install/packages/mail` im tar-Archiv enthalten.

4.3 Format

Die Info-Datei hat eine XML-Struktur, die folgendermassen aussieht:

```
<package>
...
</package>
```

Innerhalb des Paket-Tags sind die unten beschriebenen Tags erlaubt. Die Werte sind, soweit nicht anders gekennzeichnet, Freitext. Werte, die nicht Freitext sind, sind case-sensitiv (auf Gross- und Kleinschreibung achten!). Freitexte dürfen folgende Zeichen nicht enthalten: „<“, „>“,„$“.

Jedes Tag (bis auf `<description>`) muss mit Start-Tag, Wert und End-Tag in genau einer Zeile stehen.

Im folgenden werden alle Tags einer Infodatei kurz erläutert. Bis auf die mit (optional) gekennzeichneten Ausnahmen, darf keines dieser Tags leer bleiben oder fehlen!

Tag	Bedeutung
`<name>`	Der Name des Paketes.
`<short>`	Eine Kurzbeschreibung (max 60 Zeichen). Dieses Tag sollte nicht nochmals den Namen des Paketes enthalten.
`<version>`	Die Version des Paketes (siehe version-Tag (Seite 17)).
`<date>`	Das Releasedatum dieser Version.
`<system>`	Für welches System dieses Paket zugelassen ist. Dieses Tag kann mehrmals vorhanden sein, wenn das Paket auf mehreren Zielsystemen installiert werden kann.
`<author>`	Der Name des Autors inklusive Email-Adresse.
`<status>`	Der Status des Paketes (siehe status-Tag (Seite 18)).
`<section>`	Der Bereich, dem dieses Paket angehört (siehe section-Tag (Seite 19)).
`<sha1sum>`	(optional) Checksumme zur automatischen Prüfung der Integrität eines Pakets (ab base 1.4.0). Hinweis: Dieses Tag kann im PackageInfoFile innerhalb des Archives nicht enthalten sein, da zum Zeitpunkt der Erstellung des Archives die Checksumme noch nicht bekannt ist. Das Tag ist somit nur in der externen *.info-Datei vorhanden.
`<space>`	(optional) Gesamter zur Installation benötigter Plattenplatz (ab base 1.4.0). Hinweis: Dieses Tag kann im PackageInfoFile innerhalb des Archives nicht enthalten sein, da zum Zeitpunkt der Erstellung des Archives die Grösse noch nicht bekannt ist. Das Tag ist somit nur in der externen *.info-Datei vorhanden.
`<require-package>`	(optional) Weiteres Paket, welches zum Betrieb unbedingt benötigt wird. (siehe require-package-tag (Seite 22))
`<require-lib>`	(optional) Verweist auf ein Library-Paket (siehe require-lib-tag (Seite 23))
`<sub-package>`	(optional) Ist ein Paket inkrementell aufgebaut, werden hier alle weiteren Teilpakete angegeben. (siehe sub-package-tag (Seite 23))
`<description>`	Eine ausführlichere Beschreibung des Programmes, Hinweise, Tipps, Patches ...

Die Reihenfolge der Tags ist beliebig. Die Tag-Namen müssen in Kleinbuchstaben gesetzt sein. Es folgen weitere Beschreibungen der Tags.

4.3.1 <name>-Tag

Das `<name>`-Tag darf nur die Zeichen a-z, 0-9, „-“ und „_“ enthalten, wobei eine Angabe der Versionsnummer des in dem Paket enthaltenen Programms vermieden werden sollte. Es gibt allerdings Ausnahmen, wie z.B. `<name>`apache2`</name>` und `<name>`apache`</name>`.

4.3.2 <version>-Tag

Wie in der OpenSource-Welt üblich, werden auch die eisfair-Pakete mit dreiteiligen Versionsnummern nach folgendem Schema versehen: `X.Y.Z`,

Diese drei Teile haben jeweils eine konkrete Bedeutung.

X - Major Der erste Block gibt ein sog. *Major-Release* an. Das heisst, das ist eine Art Hauptversion des Paketes. Diese Nummer ändert sich i. d. R. nur dann, wenn das Paket komplett neu aufgebaut wurde oder die internen Komponenten ebenfalls einen Sprung der Major-Version gemacht haben.

Y - Minor Der zweite Block gibt ein *Minor-Release* an. Das heisst, jede Weiterentwicklung eines Paketes in Verbindung mit neuen Features, wird an dieser Stelle einen Versionsschritt machen. Wenn also bei einem Paket ein neues Feature eingebaut wird, dann wird dieses bspw. einen Versionsschritt von *1.2.5* zu *1.3.0* machen.

Z - Bugfix bzw. Patchlevel Wenn in einem Paket Fehler beseitigt werden, so schlägt sich das auf die letzte Ziffer der Versionsbezeichnung nieder, also auf *Z*. Das ist die Nummer des *Bugfix-Release*. Um beim vorigen Beispiel zu bleiben: Es wird für ein Paket in der Version 1.2.5 ein Fehler gemeldet und behoben. Somit hat das neue Paket die Version 1.2.6.

So entwickelt sich die Versionsnummer weiter. Nach einem weiteren Bugfix wird es die Version 1.2.7 und nach dem Einbau eines neuen Features wie bspw. einer neuen Konfigurationsvariablen wird es die 1.3.0. Die Verwendung dieses Versionierungsschemas ermöglicht es zudem, die korrekte Versionsnummernvergabe vollständig zu automatisieren.

Die Variablen haben den Wertebereich von 0-99 und die Version 0.0.0 ist verboten.

Hinweis

Üblicherweise ist es so, dass bei Developerversionen (Testing-Versionen) der Minor (Y) ungerade und bei den stabilen Versionen gerade ist. Dieses Vorgehen wird vom Paket packagedevelopment konsequent umgesetzt.

4.3.3 <date>-Tag

Das Release-Datum des Pakets im Format: `YYYY-MM-DD`, (nach ISO 8601)

YYYY	4-stellige Jahreszahl
MM	2-stelliger Monat
DD	2-stelliger Tag

4.3.4 <status>-Tag

Ein eisfair-Paket kennt drei verschiedene Status, welche sich aus folgender Überlegung ergeben: Ein völlig neues Paket beginnt sein Leben mit einer Versionsnummer 0.x.x, also kleiner als Major-Version 1.0.0. Alle Pakete mit einer solchen Version werden per Konvention als *unstable* gekennzeichnet. Ein solches Paket wurde noch nicht getestet und die Installation ist nur Experten zu empfehlen, da dieses Paket noch erhebliche Fehler und Ungereimtheiten enthalten kann.

Hat das Paket einen gewissen Reifegrad erlangt, dann wird es die Version 1.0.0 erreichen. Ab hier gilt das bereits beim Version-Tag angesprochene Vorgehen, dass ein Paket mit geradem Minor als *stable* und ein Paket mit ungeradem Minor als *testing* gekennzeichnet wird.

So ist die Version 1.0.0 eine Stable-Version und die Versionen 1.0.x sind die Bugfixes dieses Stable-Release. Die Version 1.1.0 wäre demgegenüber jedoch eine Testing-Version. Die gesamte Frage nach dem Status eines Pakets wird damit wesentlich vereinfacht, da sich dieser unmittelbar aus der Versionsnummer herleiten läesst. Es ergibt sich bspw. folgender Ablauf:

- Release 1.0.0 <– Stable
- Bugfix 1.0.1 <– ebenfalls Stable
- Bugfix 1.0.2 <– nochmal Stable
- Vorabversion 1.1.0 <– Testing
- Bugfix 1.0.3 <– wieder Stable
- Vorabversion 1.1.1 <– Testing
- Release 1.2.0 <– Stable
- Bugfix 1.2.1 <– Bugfix der neuesten Stable-Version
- Bugfix 1.0.4 <– Bugfix einer älteren Version

Wie dieses Beispiel zeigt, werden hier problemlos drei verschiedene Versionen gepflegt: Das aktuelle Stable (1.2.0) durch Bugfixes (1.2.1), ein älteres Stable (1.0.0) durch Bugfixes (1.0.1 bis 1.0.4) sowie die Entwicklerversion 1.1.0 bis 1.1.2. Da das Release 1.2.0 aus der Testing 1.1.1 hervorgegangen ist, muss jedoch sichergestellt werden, dass die nun kommenden Testing-Versionen bei der nächsten ungeraden Nummer nach dem Release weiterlaufen. Das wären hier die Versionen 1.3.0, 1.3.1 usw.

Bzgl. der Version eines Paketes und dem damit einher gehenden Status sollen folgende Grundsätze zur Anwendung kommen:

testing Dieses Paket wurde schon getestet und alle bekannten Fehler wurden beseitigt. Wenn ein Paket diesen Status erreicht hat, sollte es einen Feature-Freeze erhalten, d. h. es werden keine neuen Features mehr eingebaut, sondern nur noch Bugfixes vorgenommen. Ein Paket soll diesen Status erst dann erhalten, wenn mindestens fünf Benutzer das Paket installiert haben und alle gemeldeten Fehler vom Autor beseitigt wurden.

stable Dieses Paket ist ausreichend getestet und arbeitet auch in vielen Kombinationen mit anderen Programmen und Treibern wie erwartet. Eine Dokumentation ist vorhanden. Ein Paket soll bzw. kann diesen Status erst dann erhalten, wenn es vorher den Status „unstable“ bzw. „testing“ besass.

Hinweis

Das Paket „packagedevelopment“ erzeugt bis einschliesslich Paketversion 0.5.5 Pakete mit Versionen 0.x.x mit dem Status *testing*. Es gibt bereits einen Feature-Request im eisfair Bugtracker um hier das korrekte Verhalten zu implementieren.

4.3.5 <section>-Tag

Dieser Wert ordnet das Paket in eine bestimmte Kategorie ein. Darüber lässt sich eine Menüauswahl o.ä. realisieren. Es existieren folgende Möglichkeiten für diesen Wert:

- backup - In dieser Sektion sollen Pakete platziert werden, welche sich mit dem Sichern und Wiederherstellen von Daten beschäftigen.
- base - Diese Sektion enthält Basisbestandteile des eisfair-Systems und wird hauptsächlich vom Team direkt betreut.
- chat - Hier können Pakete platziert werden, welche sich mit den verschiedensten Formen des Chats beschäftigen. Beispiele hierfür sind IRC, Jabber oder auch TeamSpeak.
- communication - Die Sektion 'communication' enthält Pakete, welche sich unter dem Stichwort `Bürokommunikation` oder `Office communication` zusammenfassen lassen. Das sind bspw. die Pakete `asterisk`, `eisfax` oder `vbox`.
- contrib
- database - Hier werden alle Pakete zu Datenbanken abgelegt. Das betrifft sowohl die jeweilige Datenbank direkt, als auch die Tools dafür. Zwei Beispiele sind hier die Pakete `mysql` sowie `phpmyadmin`.
- devel - Unter dieser Sektion werden all die Pakete geführt, welche sich mit der Entwicklung beschäftigen. Das sind also all die Pakete, welche für die Entwicklung von weiteren eisfair-Paketen benötigt werden, also gcc & Co.
- drivers - In dieser Sektion werden Treiber-Pakete abgelegt, wie sie für spezielle Hardware benötigt werden. Dies sind bspw. Pakete mit Treibern für die AVM-ISDN-Karten oder RAID-Controller.
- game - In der Sektion 'game' sollen Pakete platziert werden, welche Serverdienste bzw. Spiele-Proxies für (Online-)Spiele bieten.
- kernel - Diese Sektion enthält Basisbestandteile des eisfair-Systems und wird hauptsächlich vom Team direkt betreut.

- lib - Diese Sektion beinhaltet die Bibliotheks-Pakete. Diese Pakete müssen immer mit der Zeichenfolge `lib` beginnen, was jedoch von packagedevelopment beim Anlegen eines solchen Paketes ggf. automatisch ergänzt wird.
- libdev - In dieser Sektion werden die Dev-Pakete zur jeweiligen Bibliothek aufgeführt.
- mail - Hier werden alle Pakete in Zusammenhang mit Emails platziert. Beispiele dafür sind natürlich das `mail` Paket, sowie `antispam` oder auch `archimap`.
- misc - In der Sektion `misc` können Pakete platziert werden, welche eine Funktionalität bieten, welche sich nicht einer der anderen Sektion zuordnen lässt. Beispiele sind hier die Pakete `lcd` oder `motion`.
- multimedia - Die Sektion `multimedia` dient der Aufnahme von Paketen welche sich mit jeglicher Form von Video- und/oder Audiodaten beschäftigen. Dazu zählen bspw. die Pakete `vdr`, `mpg123` oder `cdrecord`.
- netservices - Die Pakete dieser Sektion bieten Dienste in Zusammenhang mit dem Netzwerk an. Dabei handelt es sich um die verschiedensten Arten von Services und Daemons wie sie bspw. die Pakete `bonding`, `dns` oder `inet` bieten.
- netutils - Im Gegensatz zur vorherigen Sektion werden hier die Pakete geführt, welche den Admin bei seiner Arbeit unterstützen sollen. Also finden sich hier verschiedene Utilities wie bspw. `addhost` oder auch `phpldapadmin`.
- news - In dieser Sektion werden Pakete abgelegt, welche sich mit dem News-System beschäftigen, also Newsreader.
- plang - Die Sektion `plang` beinhaltet Pakete zur Entwicklung bzw. Programmierung in den verschiedensten Programmiersprachen. Dies sind bspw. die Pakete für Java, Python oder Perl.
- printer-file - In dieser Sektion werden die Pakete für Druck- und Dateidienste abgelegt. Dies sind bspw. die Pakete für NFS-Freigaben sowie Samba oder auch verschiedene Konverter wie `html2ps` oder `a2ps`.
- security - Pakete dieser Sektion beschäftigen sich mit der Sicherheit des Servers bzw. nicht nur des Servers. Dazu zählen bspw. die Pakete `certs` oder `clamav`.
- system - Hier werden Pakete für den allgemeinen Betrieb des Servers vorgehalten. Dazu zählen bspw. Pakete wie `quota` oder `smartmon`.
- utils - In der Sektion `utils` werden Tools und Werkzeuge untergebracht, welche für die ganz normale Benutzung oder auch Administration des Servers benötigt werden. Beispiele sind die Pakete `zip`, `sasl` oder `chroot`.
- web - Hier werden die Pakete für den Webzugriff auf den Server vorgehalten. Dies sind die Basispakete wie bspw. `apache2` sowie darauf aufbauend bspw. `apache2_webalizer` oder `trac`.

Entfallene Sektionen

Im Zuge der Bereinigung der Sektion-Struktur sind die folgenden Sektionen entfallen bzw. aufgesplittet oder umbenannt worden. Dem einen oder anderen Entwickler sind diese mglw.

noch bekannt:

- admin - admin-tools zum Installieren und Konfigurieren
- dns-dhcp
- etc
- inet-mail-news
- interpreter - perl, ruby, python und co.
- net - alles rund ums Netzwerk

Die Pakete dieser Sektionen sollen zum jeweils nächsten Release nach Möglichkeit in der nun gültigen Sektion erscheinen.

4.3.6 <sha1sum>-Tag

Mit der Bildung einer Prüfsumme über das Archiv und mit dem Eintrag dieser sha1sum kann die fehlerfreie Übertragung vor der Installation geprüft werden.

Ermittlung der Checksumme:

```
sha1sum -b test.tar.gz | cut -d' ' -f1
```

Dieses Tag kann nur im separaten Info-File gesetzt werden, da zum Zeitpunkt der Berechnung der Prüfsumme das Archiv des Paketes bereits fertig ist.

4.3.7 <space>-Tag

Mit dem Eintrag des gesamten zur Installation benötigten Plattenplatzes kann geprüft werden, ob dieser zur Verfügung steht.

Der benötigte Plattenplatz berechnet sich aus der Archivgrösse zuzüglich der Grösse der Einzeldateien.

Die Angabe erfolgt in ganzen Megabyte.

Beispiel (benötigt werden 13 Megabyte):

```
<space>13</space>
```

Dieses Tag kann nur im separaten Info-File gesetzt werden, da zum Zeitpunkt der Berechnung der Gesamtgrösse das Archiv des Paketes bereits fertig ist.

4.3.8 <system>-Tag

Dieses Tag legt fest, für welches System das Paket entwickelt wurde, zum Beispiel für das System 'eisfair-1'.

Im Moment sind die Tags

```
eisfair-1
eisfair-2
eisxen-1
```

zugelassen.

```
<system>eisfair-1</system>
```

Dieses Tag darf auch mehrfach in einer Paket-Info-Datei vorkommen.

```
<system>eisfair-1</system>
<system>eisfair-2</system>
```

4.3.9 <require-package>-Tag

Dieser Wert definiert eine Abhängigkeit zwischen dem beschriebenen Paket und einem anderen Paket.

Das notwendige Paket kann dabei auf mehrere Arten referenziert werden. Die einfachste Variante ist der direkte Verweis auf die jeweilige Paket-Info-Datei. Das kann entweder durch die Angabe des relativen Pfades (Beispiele 1 und 2) oder durch die Angabe der absoluten URL (Beispiel 3) geschehen.

Alternativ kann ein Paket auch über den Paketnamen referenziert werden (Beispiele 4 und 5). Damit diese Variante verwendet werden kann, muss über die eis-list.txt (Seite 121) eine index-Datei (Seite 122) bereitgestellt werden, in der das benötigte Paket enthalten ist.

Sollte eine bestimmte Mindestversion gefordert werden, so kann diese bei allen Varianten optional durch ein Leerzeichen getrennt hinter dem Dateinamen bzw. Paketnamen angegeben werden.

Ist das beschriebene Paket von mehreren Paketen abhängig, so kann das `<require-package>`-Tag mehrfach vorkommen. Gibt es keine Abhängigkeiten, so kann es entfallen.

Die (immer vorhandene) Abhängigkeit zum Paket base muss nicht explizit genannt werden, es sein denn, eine bestimmte Version wird vorausgesetzt.

Beispiele:

```
<require-package>bar.tar.gz.info 1.3.1</require-package>
<require-package>../packages/foo.tar.gz.info 1.0.5</require-package>
<require-package>http://www.domain.de/foo.tar.gz.info 1.0.5</require-package>
<require-package>foo 1.0.5</require-package>
<require-package>bar</require-package>
```

4.3.10 <require-lib>-Tag

Da Bibliotheken aus Paketen nicht nur von einer bestimmten Anwendung benötigt werden, erfolgt ihre Auslieferung in separaten Paketen. Diese werden so vor der Installation des eigentlichen Paketes auf den Rechner geladen.

Beispiele:

```
<require-lib>libdb4 1.0.0</require-lib>
<require-lib>http://www.domain.de/libldap2.tar.gz.info 1.0.0</require-lib>
```

4.3.11 <sub-package>-Tag

Das Tag `<sub-package>` gibt an, welche Unterpakete vom Hauptpaket selbständig geladen werden. Diese Angaben dienen lediglich der Information, z.B. für *eisfair* -Download-Mirror-Systeme.

Syntax:

`<sub-package>`LOCATION EXTRA_DATA`</sub-package>`

Die Form von LOCATION entspricht derselben wie für `<require-package>`.

Als EXTRA_DATA können zusätzliche Informationen übermittelt werden. Beispielsweise wird beim base-Update der benötigte Platz auf der Festplatte in MB angegeben. Dieser Wert ist paketspezifisch und optional.

4.4 Update eines Pakets

Wenn ein Paket aktualisiert wird, sind folgende Punkte zu beachten:

- Updaten des `<version>`-tags
- Eventuell Rückstufung des `<status>`-Tags

Wird für die Paketentwicklungs "packagedevelopment" verwendet, werden diese Punkte automatisch korrekt behandelt.

4.5 Beispiel

Anbei ein kleines Beispiel, welches obige Definitionen illustrieren soll.

```
<package>
<name>mail</name>
<short>Mail services</short>
<version>1.2.7</version>
<date>2004-06-25</date>
<system>eisfair-1</system>
<author>Juergen Edner, fli4l-eisfair(at)telejeck(dot)de</author>
<status>testing</status>
<section>mail</section>
<sha1sum>559df80365121fb576761f145011e2ebbcfc71cb</sha1sum>
<space>13</space>
<require-lib>libdb4 1.0.0</require-lib>
<require-lib>libgdbm 1.0.0</require-lib>
<require-lib>libssl 1.0.0</require-lib>
<require-package>base 1.0.5</require-package>
<require-package>inet 1.0.0</require-package>
<require-package>perl 1.0.0</require-package>
<description>
Mail Services

EXIM      Version: 4.34            EXISCAN   Version: 4.34-22
FETCHMAIL Version: 6.2.5           IPOP3D    Version: 2003.83
IMAPD     Version: 2003.338        VACATION  Version: 1.2.6.1

Attention: This package requires as minimum an installed update-1.0.5
           package!
Attention: Be careful, only mail usernames in lowercase are allowed in
           this package!

</description>
</package>
```

5 Installation, Bootprozess, Shutdown

5.1 Paketinstallation

Da es zur Installation einer Software meist nicht ausreicht, die Dateien an die richtige Stelle zu kopieren, sondern normalerweise zusätzlich gewisse aktionen ausgeführt werden müssen, gibt es auf *eisfair* die Installationsskripte `preinstall.sh` und `install.sh`.

Beide Skripte sind optional, müssen also nicht zwingend in jedem Paket enthalten sein.

5.1.1 preinstall.sh

Bei der Paketinstallation wird nach dem Download des Pakets als erstes – falls vorhanden – das Skript `tmp/preinstall.sh` aus dem archiv extrahiert und ausgeführt.

In diesem Skript wird üblicherweise als erster Schritt eine überprüfung der installierten Base-Version durchgeführt. auf `<require-package>` darf man sich hier nicht verlassen, da dieses Tag erst mit Base Version 1.0.3, und die Variante über Paketnamen erst seit Base Version 1.0.8 eingeführt wurde.

Beispiel für die Prüfung der *eisfair* -Version in `preinstall.sh`:

```
required_base='1.6.3'

### check base version
if [ ! -f /var/install/bin/check-version ] ||
   [ `/var/install/bin/check-version base $required_base` = 'new' ]
then
    echo
    echo
    colecho "Wrong eisfair version!" rd w brinv
    colecho "Version must be $required_base or higher." rd w brinv
    colecho "Please update your eisfair system first." rd w brinv
    echo
    echo
    exit 1
fi
```

Dies ist auch die einzige Stelle, an der `echo/colecho` noch erlaubt ist. Die Verwendung von `mecho` ist hier nicht möglich, da ja gerade auf eine installierte base-Version geprüft wird und es beispielsweise bei *eisfair* Version 1.0.3 noch kein mecho gab.

Wie in diesem Beispiel ersichtlich kann die Installation im Fehlerfall über einen Rückgabewert ungleich Null abgebrochen werden.

Neben dieser Prüfung von Installationsvoraussetzungen können in diesem Skript Vorbereitungen zur Installation getroffen werden, beispielsweise die Deinstallation der vorhergehenden Version eines Pakets.

Beispiel für die Deinstallation einer alten Paketversion in `preinstall.sh`:

```
package_name='foo'

### deinstall old package version
if [ -f "/var/install/deinstall/${package_name}" ]
then
    /var/install/deinstall/${package_name} update
fi
```

Ab der *eisfair* Version 1.7.1 können Pakete bei einem update auch über die Standardfunktionen, messages, Meldungen schreiben, dafür ist eine Anpassung für die Deinstallation notwendig. Auch sollte der Parameter update mit '—' 2 Zeichen beginnen, —update. Vorsicht bitte bei Paketen, die im Deinstall Skript noch 'update' stehen haben

Beispiel für die Deinstallation einer alten Paketversion in `preinstall.sh`:

```
package_name='foo'

### deinstall old package version
if [ -f "/var/install/deinstall/${package_name}" ]
then
    /var/install/deinstall/del-package -p ${package_name} --update
fi
```

achtung: Bei der ausführung der `preinstall.sh` sind die Dateien des Pakets noch nicht installiert. auf Paketbestandteile kann daher nicht zugegriffen werden!

5.1.2 install.sh

Sofern das Preinstall–Skript nicht mit einem Fehler abgebrochen hat, wird als nächster Schritt das komplette Tar–archiv entpackt, und zwar immer vom Root-Verzeichnis „/“ ausgehend, so dass alle Dateien im richtigen Verzeichnis landen.

Danach wird – falls vorhanden – das Skript `/tmp/install.sh` ausgeführt. Dieses Skript dient zur Installation des Pakets, also z.B. zur Erstellung von notwendigen symbolischen Links, zum Einfügen von Menüeinträgen oder zur übernahme der Konfiguration einer alten Paketversion.

Beispiel für die Installation eines *eisfair*-Pakets:

```
#!/bin/sh
# ----------------------------------------------------------------------
# /tmp/install.sh - foo installation
#
# Creation   :  2009-12-31 starwarsfan
# Last update:  $Id: foo 22527 2010-01-02 16:51:38Z starwarsfan $
#
# Copyright (c) 2001-2013 the eisfair team, team(at)eisfair(dot)org
```

```
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
# ---------------------------------------------------------------------

# include eislib
. /var/install/include/eislib

# set package name
package_name=foo

# ---------------------------------------------------------------------
# Setup step 1
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# add menu
# ---------------------------------------------------------------------
/var/install/bin/add-menu     \
        setup.services.menu   \
        setup.services.${package_name}.menu \
        "${package_name}"

# ---------------------------------------------------------------------
# Create default config
# ---------------------------------------------------------------------
if [ -f "/etc/config.d/${package_name}" ]
then
    update=true
fi

/var/install/config.d/${package_name}-update.sh


# ---------------------------------------------------------------------
# Aktivate configuration if necessary
# ---------------------------------------------------------------------
if "${update:-false}"
then
    /var/install/config.d/${package_name}.sh --quiet
    /etc/init.d/${package_name} start
fi


exit 0
# ---------------------------------------------------------------------
# End
# ---------------------------------------------------------------------
```

5.2 Update von Paketen

Beim Update eines Pakets gibt es mehrere Dinge zu beachten. Besonders ist zu Berücksichtigen, dass das Upgrade oftmals nicht von der letzten Version erfolgt, sondern dass ein Update auch von einer sehr viel älteren Version erfolgen kann.

In jedem Fall müssen beim Update die obsoleten Dateien der Vorversion entfernt werden. Zudem ist es für *eisfair* -Pakete verpfichtend, dass die Konfiguration der Vorversion übernommen wird. Dabei müssen auch eventülle änderungen der Konfiguration (z.B. Hinzufügen neür Variablen) durchgeführt werden.

Für das Entfernen der Vorversion hat sich die Verwendung des Deinstall-Skripts der alten Version bewährt. Da bei einem Update nicht alle Dateien gelöscht werden dürfen (z.B. die Konfiguration), muss das Deinstall–Skript einen Update-Modus kennen, d.h. es werden nicht alle Dateien gelöscht, wenn der Parameter „`update`“ übergeben wird. Ein Beispiel dazu ist im Deinstall-Script unter „Deinstallieren von Paketen (Seite 37)“ zu finden.

Bei der übernahme der Konfiguration gibt es mehrere Fälle. Im einfachsten Fall handelt es sich gar nicht um ein Update des Pakets sondern um eine Neuinstallation und es ist die Default–Konfiguration aus `/etc/default.d/` zu übernehmen.

Sofern bereits eine Konfiguration einer Vorversion unter `/etc/config.d/` vorhanden ist, dient diese als Basis zur Erstellung der Konfiguration. Dabei müssen:

- Neü Konfigurationsvariablen, die in der alten Konfiguration noch nicht vorhanden sind, in die Konfiguration eingefügt werden.
- alte Konfigurationsvariablen, die es in der aktüllen Paketversion nicht mehr gibt, entfernt werden.
- Konfigurationsvariablen, die sich geändert haben, angepasst werden. (z.B. Umbenennung, anpassung der Syntax...)
- Unveränderte Konfigurationsvariablen übernommen werden.

Für diese übernahme der Konfiguration muss daher ein eigenes Skript erstellt werden, das üblicherweise `/var/install/config.d/`*$package*`-update.sh` genannt wird.

Beispiel für ein Skript zur übernahme der Konfiguration eines *eisfair* -Pakets:

```
#!/bin/sh
# ----------------------------------------------------------------------
# /var/install/config.d/foo-update.sh - paramater update script
#
# Creation    :  2009-12-31 starwarsfan
# Last update:  $Id: foo-update.sh 2342 2010-01-02 16:51:38Z starwarsfan $
#
# Copyright (c) 2001-2013 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
```

```
# ----------------------------------------------------------------------

#exec 2> `pwd`/foo-update-trace$$.log
#set -x

# include configlib for using printvar
. /var/install/include/configlib

# set package name
package_name=foo

# ----------------------------------------------------------------------
# Set the default val"us for configuration
# ----------------------------------------------------------------------
START_FOO='no'


# ----------------------------------------------------------------------
# Read old configuration and rename old variables
# ----------------------------------------------------------------------
rename_old_variables()
{
    # read old values
    if [ -f /etc/config.d/${package_name} ]
    then
        /var/install/bin/backup-file --quiet ${package_name}
        . /etc/config.d/${package_name}
    fi
}

# ----------------------------------------------------------------------
# Write config and default files
# ----------------------------------------------------------------------
make_config_file()
{
    internal_conf_file=${1}
    {
    # ------------------------------------------------------------------
    printgpl --conf ${packageName} "2009-12-31" "starwarsfan"
    # ------------------------------------------------------------------


    # ------------------------------------------------------------------
    printgroup "Basic configuration"
    # ------------------------------------------------------------------
    printvar "START_FOO" "Use: yes or no"

    # ------------------------------------------------------------------
    printend
    # ------------------------------------------------------------------

    } > ${internal_conf_file}
    # Set rights
```

```
    chmod 0600 ${internal_conf_file}
    chown root ${internal_conf_file}
}

# ----------------------------------------------------------------------
# Create the check.d file
# ----------------------------------------------------------------------
make_check_file()
{
    printgpl --check ${package_name} '2009-12-31' 'starwarsfan' \
        >/etc/check.d/${package_name}
    cat >> /etc/check.d/${package_name} <<EOFG
# Variable       OPT_VARIABLE         VARIABLE_N           VALUE
START_FOO      -                -                   YESNO

EOFG

    # Set rights for check.d file
    chmod 0600 /etc/check.d/${package_name}
    chown root /etc/check.d/${package_name}

#    printgpl --check_exp ${package_name} '2009-12-31' 'starwarsfan' \
#        >/etc/check.d/${package_name}.exp
#    cat >> /etc/check.d/${package_name}.exp <<EOFG
#
#
#EOFG

    # Set rights for check.exp file
#    chmod 0600 /etc/check.d/${package_name}.exp
#    chown root /etc/check.d/${package_name}.exp

#    printgpl -check_ext ${package_name} '2009-12-31' 'starwarsfan' \
#        >/etc/check.d/${package_name}.ext
#    cat >> /etc/check.d/${package_name}.ext <<EOFG
#
#
#EOFG

    # Set rights for check.ext file
#    chmod 0600 /etc/check.d/${package_name}.ext
#    chown root /etc/check.d/${package_name}.ext
}



# ----------------------------------------------------------------------
# Main
# ----------------------------------------------------------------------
# Write default config file
make_config_file /etc/default.d/${package_name}

# Update from old version
```

```
rename_old_variables

# Write new config file
make_config_file /etc/config.d/${package_name}

# Write check.d file
make_check_file

exit 0
# ----------------------------------------------------------------------
# End
# ----------------------------------------------------------------------
```

Je nach Anwendungsfall muss dieses Script natürlich noch um die Behandlung weiterer Fälle wie „Löschen von Variablen“, „Umbenennen von Variablen“ oder ähnlichem erweitert werden.

Es ist zu beachten, das bei array Variablen der erste Eintrag vorhanden sein muss, auch wenn das gesamte array nicht verwendet wird. Dies ist besonders bei Nutzung von joe oder vi an Stelle des ece als Editor relevant.

Nach dem Neuschreiben der Konfiguration muss diese noch auf Gültigkeit geprüft und der Dienst neu gestartet werden. Das kann mit einem direkten Aufruf des Edit–Skripts geschehen und gibt dem Anwender gleich die Möglichkeit, die veränderte Konfiguration zu überarbeiten.

5.2.1 Erweitertes update

Bei Verwendung dieses Beispiels werden die Dateien unter `/etc/default.d` und `/etc/check.d` nicht mit ausgeliefert,
da diese beim Verarbeiten des `$package-update.sh` erzeugt werden.
Gleichzeitig wird bei den check Dateien mit 'Here-Documents' gearbeitet.

```
#!/bin/sh
# ----------------------------------------------------------------------
# /var/install/config.d/eisfax-update.sh - parameter update script
#
# Creation:     2005-05-29 jv
# Last Update:  2006-06-11 hb
#
# Copyright (c) 2001-2013 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
# ----------------------------------------------------------------------


#exec 2>/public/eisfax-update-trace$$.log
#set -x
```

```
package_name=eisfax

# include configlib for using printvar
. /var/install/include/configlib
. /var/lib/eisfax/eisfax.info

### ---------------------------------------------------------------------
### Set the default values for configuration
### ---------------------------------------------------------------------
START_EISFAX='no'

EISFAX_COUNTRY_CODE='49'
EISFAX_AREAa_CODE='40'
EISFAX_LONG_DISTANCE_PREFIX='0'
EISFAX_INTERNATIONAL_PREFIX='00'
EISFAX_MAX_DIALS='6'
EISFAX_ADMIN_USER_FAX_PASSWD=''
EISFAX_SAMBA_PRINTER_AUTO='yes'

EISFAX_ANALOG_USE='no'

EISFAX_ANALOG_N='1'

EISFAX_ANALOG_1_NaME=''
EISFAX_ANALOG_1_ACTIVE='no'
EISFAX_ANALOG_1_DEVICE='ttyS0'
EISFAX_ANALOG_1_FAaX_PRINTER_NAME='EISFAX'
EISFAX_ANALOG_1_CONFIGURE_AUTO='yes'
EISFAX_ANALOG_1_ID='+49.40.12345678'
EISFAX_ANALOG_1_LOCaL_IDENTIFIER='Fax-Station'
EISFAX_ANALOG_1_SERVER_NaME='EISFAX Server'
EISFAX_ANALOG_1_FAX_HEADER='von %%l %%n an %%d Seite %%P von %%T /'
EISFAX_ANALOG_1_HEAaDER_DaTE='%F %R'
EISFAX_ANALOG_1_RINGS_BEFORE_ANSWER='1'
EISFAX_ANALOG_1_SPEAKER_VOLUME='off'
EISFAX_ANALOG_1_SND='yes'
[...]

### ---------------------------------------------------------------------
### Read old configuration and rename old variables
### ---------------------------------------------------------------------
rename_old_variables()
{
    # read old val"us
    if [ -f /etc/config.d/${package_name} ]
    then
        /var/install/bin/backup-file --quiet ${package_name}
        . /etc/config.d/${package_name}
    fi

}
```

```
### ---------------------------------------------------------------------
### Write config and default files
### ---------------------------------------------------------------------
make_config_file()
{
    internal_conf_file=${1}
    {
    # ---------------------------------------------------------------------
    printgpl --conf ${package_name} "2005-06-26" "hb"
    # ---------------------------------------------------------------------

    # ---------------------------------------------------------------------
    printgroup "Basic configuration"
    # ---------------------------------------------------------------------

    printvar "START_EISFAX" "Start EISFAX server"
    printvar "EISFAX_COUNTRY_CODE" ""
    printvar "EISFAX_AREA_CODE"  ""
    printvar "EISFAX_LONG_DISTANCE_PREFIX" ""
    printvar "EISFAaX_INTERNATIONAL_PREFIX" ""
    printvar "EISFAX_MAX_DIALS" "Wieviele Wahlwiederholungen"
    printvar "" "ausgefuehrt werden sollen"

    printvar "EISFAX_ADMIN_USER_FAX_PASSWD" "Passwd for admin-user fax"

    printvar "EISFAX_SAMBA_PRINTER_AUTO" "Soll EISFAX die Samba config"
    printvar "" "aufrufen: yes or no"

    # ---------------------------------------------------------------------
    printgroup "ANALOG settings"
    # ---------------------------------------------------------------------

    printvar "EISFAX_ANALOG_USE" "Use: yes or no"

    printvar "EISFAX_ANALOG_N" "number of modems"

    # begin EISFAX_ANALOG_N
    idx=1

    while [ "${idx}" -le "${EISFAX_ANALOG_N}" ]
    do

        # section marker EISFAX_ANALOG_N
        if [ "${idx}" -le  "${EISFAX_ANALOG_N}"  ]
        then
            # -------------------------------------------------------------
            printgroup "EISFAX_ANALOG_${idx}"
            # -------------------------------------------------------------
        fi

        printvar "EISFAX_ANALOG_${idx}_NAME" \
                       "Use a name what ever you want, it's only for you"
        printvar "EISFAX_ANALOG_${idx}_ACTIVE" "Use only: yes or no"
```

```
        printvar "EISFAX_ANALOG_${idx}_DEVICE" "ttyS0 or ttyS1 or ttyS2"
        printvar "EISFAX_ANALOG_${idx}_FaX_PRINTER_NAME" \
                                          "Please, look at the Doku"
        printvar "EISFAX_ANALOG_${idx}_CONFIGURE_AUTO" \
                                        "Use: yes or no or scratch"
        printvar "EISFAX_ANALOG_${idx}_ID"
                                     "Your Fax-ID (+49.40.12345678)"
        printvar "EISFAX_ANALOG_${idx}_LOCAL_IDENTIFIER" \
                                     "Fax_Station name (fax header)"
        printvar "EISFAX_ANALOG_${idx}_SERVER_NAME" "EISFAX Server"
        printvar "EISFAX_ANALOG_${idx}_FAX_HEADER" \
                              "von %%l %%n an %%d Seite %%P von %%T /"
        printvar "EISFAX_ANALOG_${idx}_HEADER_DATE" "%F %R"
        printvar "EISFAX_ANALOG_${idx}_RINGS_BEFORE_ANSWER" "1"
        printvar "EISFAX_ANALOG_${idx}_SPEAKER_VOLUME" \
                                     "off, quiet, low, medium, high"
        printvar "EISFAX_ANALOG_${idx}_SND" "Use: yes or no"
        [...]

    # end EISFAX_ANALG_N
    idx=`${EXPR_EXEC} ${idx} + 1`
    done

    # ------------------------------------------------------------------
    printend
    # ------------------------------------------------------------------

    } > ${internal_conf_file}
    # Set rights
    chmod 0600 ${internal_conf_file}
    chown root ${internal_conf_file}
}

### ---------------------------------------------------------------------
### Create the check.d file
### ---------------------------------------------------------------------
make_check_file()
{
    printgpl --check ${package_name} "2005-06-26" "jv" \
        >/etc/check.d/${package_name}
    cat >>/etc/check.d/${package_name} <<EOF
# Variable                          OPT_VARIABLE            VARIABLE_N        VALUE
START_EISFAX                        -                       -                 YESNO

EISFAX_COUNTRY_CODE                 START_EISFAX            -                 EISFAX_
EISFAX_AREA_CODE                    START_EISFAX            -                 NONE
EISFAX_LONG_DISTANCE_PREFIX         START_EISFAX            -                 ENUMERI
EISFAX_INTERNATIONaL_PREFIX         START_EISFAX            -                 ENUMERI
EISFAX_MAX_DIALS                    START_EISFAX            -                 NUMERIC
EISFAX_aDMIN_USER_FaX_PaSSWD        START_EISFAX            -                 NOTEMPT
EISFAX_SAMBA_PRINTER_AUTO           START_EISFAX            -                 YESNO

EISFAX_ANALOG_USE                   START_EISFAX            -                 YESNO
```

```
EISFAX_ANALOG_N                      EISFAX_ANALOG_USE        -                 NUMERIC
EISFAX_ANALOG_%_NAME                 EISFAX_ANALOG_USE        EISFAX_ANALOG_N   NONE
EISFAX_ANALOG_%_ACTIVE               EISFAX_ANALOG_USE        EISFAX_ANALOG_N   YESNO
EISFAX_ANALOG_%_DEVICE               EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   NOTEMPT
EISFAX_ANALOG_%_FaX_PRINTER_NAME     EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   EISFAX_
EISFAX_ANALOG_%_FaX_PRINTER_NAME     EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   WARN_EI
EISFAX_ANALOG_%_CONFIGURE_AUTO       EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   EISFAX_
EISFAX_ANALOG_%_ID                   EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   NOTEMPT
EISFAX_ANALOG_%_LOCAL_IDENTIFIER     EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   NOTEMPT
EISFAX_ANALOG_%_SERVER_NaME          EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   NOTEMPT
EISFAX_ANALOG_%_FaX_HEaDER           EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   NOTEMPT
EISFAX_ANALOG_%_HEaDER_DaTE          EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   NOTEMPT
EISFAX_ANALOG_%_RINGS_BEFORE_aNSWER  EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   NUMERIC
EISFAX_ANALOG_%_SPEAKER_VOLUME       EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   EISFAX_
EISFAX_ANALOG_%_SND                  EISFAX_ANALOG_%_ACTIVE   EISFAX_ANALOG_N   YESNO
[...]
EOF

    # Set rights for check.d file
    chmod 0600 /etc/check.d/${package_name}
    chown root /etc/check.d/${package_name}


    printgpl --check_exp ${package_name} "2005-06-26" "jv" \
        >/etc/check.d/${package_name}.exp
    cat >>/etc/check.d/${package_name}.exp <<EOF
EISFAX_USER          = '([-,_[:lower:]0-9])+'
                     : 'no valid user name, because only the following charctershare a
EISFAX_COUNTRYC      = '(RE:NUMERIC)(RE:NUMERIC)'
                     : 'Use numeric country code: 00 ... 99'
EISFAX_FTYPE         = '()|pdf|tif|ps|png'
                     : 'Use only: pdf, tif, ps or png'
EISFAX_RFTYPE        = 'pdf|tif|ps'
                     : 'Use only: pdf, tif, or ps'
EISFAX_RSEND         = '()|no|faxmaster|(RE:MAILADDR)|(RE:EISFAX_USER)'
                     : 'Use only: no or faxmaster or e-mail address'
EISFAX_FMaSTER       = 'yes|no|errors'
                     : 'Use only: yes, no, errors'
EISFAX_SVOLUME       = 'off|quiet|low|medium|high'
                     : 'Use only: off, quiet, low, medium or high'
EISFAX_PQUEUE        = 'pr[1-3]|repr[1-9]?[0-9]|usbpr[1-9]?[0-9]'
                     : 'Use only: pr1, pr2, pr3, repr1 ... repr99, usbpr1 ... usbpr99'
[...]
EOF

    # Set rights for check.exp file
    chmod 0600 /etc/check.d/${package_name}.exp
    chown root /etc/check.d/${package_name}.exp


    printgpl --check_ext ${package_name} "2005-08-31" "hb" \
        >/etc/check.d/${package_name}.ext
```

```
    cat >>/etc/check.d/${package_name}.ext <<EOF
if ( EISFAX_ANALOG_use == "yes" && EISFAX_isdn_use == "yes" )
   then
   error "Either EISFAX_ANALOG_USE _or_ EISFAX_ISDN_USE can be 'yes' "
fi

EOF

    # Set rights for check.ext file
    chmod 0600 /etc/check.d/${package_name}.ext
    chown root /etc/check.d/${package_name}.ext
}

### ---------------------------------------------------------------------
### Main
### ---------------------------------------------------------------------
# write default config file
make_config_file /etc/default.d/${package_name}

# update from old version
rename_old_variables

# write new config file
make_config_file /etc/config.d/${package_name}

# write check.d file
make_check_file

exit 0
### ---------------------------------------------------------------------
### End
###---------------------------------------------------------------------
```

5.3 Dateiliste *$package*-files.txt

Zu jedem Paket gehört auch noch eine Liste der enthaltenen Dateien und ihrer Rechte. Diese soll später u. a. zur Überprüfung der Dateirechte dienen. Das kann besonders für Library Pakete wichtig werden, da das Überprüfen der Anwesenheit der Paketbeschreibungsdatei wenig über die Vollständigkeit der installierten Bibliotheken aussagt. Auch an eine Anpassung der Deinstallationsscripte ist in diesem Zusammenhang gedacht. Die Datei ist wie folgt aufgebaut:

```
    Filetype      b = Binaer, u = Unix-Text
    Rights        Zugriffsrechte
    User          Benutzer
    Group         Gruppe
    Package       Paketname
    File          Datei mir relativem Verzeichnis
```

Beispiel:

```
# ----------------------------------------------------------------------
# foo-files.txt - list of all files of package 'foo'
#
# Creation:     2004-11-17 jv
# Last update:  2006-05-30 max
# ----------------------------------------------------------------------
u 0755 root root foo tmp/install.sh
u 0755 root root foo tmp/preinstall.sh
b 0755 root root foo usr/bin/foorun
b 0644 root root foo usr/local/foo/bin/foo.so
u 0644 root root foo usr/local/foo/UDF/foosql.sql
u 0644 root root foo usr/share/doc/foo/changes.txt
u 0644 root root foo usr/share/doc/foo/foo.txt
```

5.4 Deinstallieren von Paketen

Die Deinstallationsroutine muss in der Datei `/var/install/deinstall/`*$package* abgelegt werden und beinhaltet sämtliche notwendigen Aktionen um ein Paket wieder sauber vom System zu entfernen.

Die gängigsten Punkte sind:

- Anhalten des Dienstes
- Löschen der Konfiguration incl. der Backups, der Check-Dateien und des apply-Skripts
- Entfernen des Start-/Stop-Skripts incl. aller Symlinks
- Entfernen der Dokumentation, Dateiliste, Paket-Info-Datei
- Entfernen der Menüeinträge und der Menüdateien
- Entfernen von Cronjobs
- Entfernen von paketspezifischen Logdateien
- Entfernen des Deinstall-Skripts
- Entfernen der eigentlichen Software incl. Konfigurationsdateien und ggf. Datenbestände

Nicht entfernt werden dürfen im allgemeinen:

- Benutzer und Benutzergruppen, sofern es noch Dateien gibt, die diesen gehören. Die Accounts sind in diesem Fall zu sperren.
- Dateien von Anwendern.

Eine abweichende Entscheidung muss im Einzelfall geklärt werden.

Sofern die Deinstallationsroutine auch zum Update eines Paketes genutzt wird, muss dort eine Sonderbehandlung erfolgen.

Beispiel für ein Skript zur Übernahme der Konfiguration eines *eisfair*-Pakets:

```
#!/bin/sh
# ----------------------------------------------------------------------
# /var/install/deinstall/foo - deinstall script
#
# Creation   :  2009-12-31 starwarsfan
# Last update:  $Id: foo 22527 2010-01-02 16:51:38Z starwarsfan $
#
# Copyright (c) 2001-2013 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
# ----------------------------------------------------------------------

# Set package name
package_name=foo

# Set filelist
filelist=/etc/filelist.d/foo-files.txt

# Check if this is an update
if [ "${1}" = "--update" ]
then
    update=true

    # ------------------------------------------------------------------
    # add everything to this list what should _not_ be removed during an
    # update. This can be folders too, but let the bash expand their
    # content using an '*' like 'foo/bar/*'. The entries listet here must
    # be given on the file list of the package. all other stuff must be
    # remove by hand using an additional line like 'rm /foo/bar/bla.txt'.
    filesToLeave="etc/config.d/foo
                  etc/backup.d/foo*
                  etc/filelist.d/foo-files.txt
                  var/install/deinstall/foo
                  var/install/menu/setup.services.foo*"
else

    filesToLeave=''
fi

# ----------------------------------------------------------------------
# Remove package content except some special files if it's an update
# instead of a deinstallation
# ----------------------------------------------------------------------
while read id perm user group package file
do
    # ------------------------------------------------
    # Remove file only if it is not on the ignore list
    removeCurrentFile=true
```

```
    for currentFileToLeave in ${filesToLeave}
    do
        if [ "${currentFileToLeave}" == "${file}" ]
        then
            removeCurrentFile=false
            break
        fi
    done

    if ${removeCurrentFile}
    then
        case ${id} in
            b|u)
                if [ "${file}" != "tmp/install.sh" -a \
                     "${file}" != "tmp/preinstall.sh" ]
                then
                    rm -f /${file}
                fi
            ;;

            f)
                # Remove directories
                rmdir --ignore-fail-on-non-empty /${file} 2>/dev/null
            ;;
        esac
    fi
done < ${filelist}



# ----------------------------------------------------------------------
# Check files and default config must be removed separately because they
# are not on the file list if the package is used as it was created by
# create-package-new.sh, they are created dynamically during installation
# ----------------------------------------------------------------------
rm -f /etc/check.d/foo*
rm -f /etc/default.d/foo*



# ----------------------------------------------------------------------
# Stop remove for update only
# ----------------------------------------------------------------------
if "${update:-false}"
then
    exit 0
fi

# ----------------------------------------------------------------------
# Remove package from menu system / remove all menu and config files
# ----------------------------------------------------------------------
/var/install/bin/del-menu setup.services.menu setup.services.foo.menu
```

```
# ----------------------------------------------------------------------
# Remove config file
# ----------------------------------------------------------------------
rm -f /etc/config.d/foo
rm -f /etc/backup.d/foo*
rm -f /etc/filelist.d/foo-files.txt

# ----------------------------------------------------------------------
# Remove deinstall script
# ----------------------------------------------------------------------
rm -f /var/install/deinstall/foo

# ----------------------------------------------------------------------
exit 0
```

5.5 Bootprozess und Shutdown

Wie bei Linux üblich werden im Rahmen des Systemstarts die einzelnen Dienste über Kontrollskripte („Init–Skript“) im Verzeichnis `/etc/init.d/` gestartet und beim Shutdown wieder beendet.

Um diesen Mechanismus nutzen zu können, benötigt man als erstes das Init–Skript namens `/etc/init.d/`*$package*. Dieses muss mehrere Startparameter verstehen —zumindest `start` zum Starten des Dienstes sowie `stop` zum Stoppen des Dienstes. Auch sollte es den Parameter '—quiet' verstehen, damit beim Aufruf mit diesem Parameter die 'echo' Ausgaben auf die Console unterdrückt werden.

Um nun festzulegen, dass ein Dienst gestartet werden soll, und an welcher Stelle im Startprozess dies geschehen soll, muss dieses Skript unter `/etc/rc2.d/S`*nn$package* verlinkt werden.

Anstatt `rc2.d` können dabei je nach Runlevel die Verzeichnisse `rc0.d`, `rc2.d`, `rc6.d` und/oder `rcS.d` in Frage kommen, wobei für die meisten Serverdienste wohl nur Runlevel 2 relevant ist.

Die Startposition wird dabei numerisch direkt hinter dem S angegeben und ist oben mit *nn* symbolisiert. Die niedrigen Nummern werden dabei für die für den Serverbetrieb nötigen Funktionen (z.B. SCSI-Treiber, Netzwerktreiber + Einstellungen) verwendet. Bei der Auswahl der Startposition für das eigene Paket ist darauf zu achten, dass es sich harmonisch in die Startreihenfolge einfügt.

ANALOG dazu wird das selbe Skript unter `/etc/rc2.d/K`*mm$package* verlinkt, um das Beenden des Dienstes beim Shutdown zu steuern. Hier ist darauf zu achten, dass unkritische Dienste als erstes beendet werden.

ACHTUNG

Grundsatz: Die Summe der Start– und Stopposition muss immer 100 ergeben.

Das Skript /etc/init.d/functions stellt ab der *eisfair* -base 1.7.4 die Funktionen wie sie im Nachfolgenden Beispiel benutzt werden bereit.

Beispiel für ein Init-Script:

```
#!/bin/sh
# ----------------------------------------------------------------------
# /etc/init.d/foo - init script for foo
#
# Creation:     2005-05-12  max
# Last Update:  2005-07-20  fabian
#
# Copyright (c) 2001-2013 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
# ----------------------------------------------------------------------

# include functions
. /etc/init.d/functions

# include configuration
. /etc/config.d/foo

_do_start ()
{
    if ${forcestart:-false}
    then
        START_FOO=yes
    fi

    ### check if starting foo is allowed
    if [ "$START_FOO" = 'yes' ]
    then
        boot_mesg " * Starting foo ..."
        loadproc foo
    fi
}

_do_stop ()
{
    boot_mesg " * Stopping foo ..."
    killproc foo
}

_do_status ()
{
    statusproc foo
}

_do_usage ()
```

```
{
    echo "Usage: $0 [-q|--quiet] {start|forcestart|stop|restart}"
}

#---------------------------------------------------------------------------
# main
#---------------------------------------------------------------------------

while [ ${#} -gt 0 ]
do
    case "${1}" in
        -q|--quiet)
            _quiet=true
            shift
        ;;
        *)
            _action=${1}
            shift
        ;;
    esac
done

case "${_action}" in
    start)
        _do_start
    ;;
    forcestart)
        forcestart=true
        _do_start
    ;;
    stop)
        _do_stop
    ;;
    restart)
        _do_stop
        sleep 3
        _do_start
    ;;
    status)
        _do_status
    ;;
    *)
        _do_usage
        exit 1
    ;;
esac

exit 0
```

Im Beispiel ist neben den oben bereits erklärten Parametern `start` und `stop` der Parameter `restart` aufgeführt, der für die Verarbeitung einer Konfiguartionsänderung notwendig ist und im weiteren Verlauf dieser Dokumentation noch näher erklärt wird.

Hinweis

Es muss garantiert sein, dass der Dienst im init.d–Script startbar ist, auch wenn START_FOO auf „no“ gesetzt ist. Dies wird über den Parameter „forcestart“ gewährleistet.

Als weiteres Startskript gibt es `/etc/init.d/boot.`*$package*. Dieses Skript dient zum Aufräumen vor dem eigentlichen Starten des Dienstes und wird **vor** den Init–Skripten beim Systemstart ausgeführt.

Für dieses Skript ist ANALOG ein Link `/etc/init.d/boot.d/S`*nn$package* zur Festlegung der Startposition zu erstellen. auch dieses Skript wird mit dem Parameter `start` aufgerufen.

Hier können z.B. übriggebliebene temporäre Dateien oder auch eventuell veränderte Zugriffsrechte einer Datei oder eines Verzeichnisses korrigiert werden.

6 Eisfair Konfigurationsschicht

6.1 Grundlagen der *eisfair* Konfiguration

Das wohl wichtigste Merkmal von *eisfair* ist die für alle Pakete einheitliche Konfiguration. Deshalb ist die Nutzung der *eisfair*–Konfigurationsschicht eine absolute Voraussetzung für ein *eisfair*–Paket.

Im Folgenden wird der Aufbau einer solchen *eisfair*–Konfigurationsdatei sowie die damit zusammenhängenden Mechanismen beschrieben.

6.2 Speicherung der Konfiguration

Die aktuelle Konfiguration eines Pakets wird immer in der Datei `/etc/config.d/`*$package* gespeichert. Der Aufbau der Konfigurationsdateien ist weiter unten detailiert beschrieben.

Diese Datei wird bei einem *eisfair*–Paket nicht mitgeliefert, da diese bei der Installation eine eventuell bereits vorhandene Konfiguration einer vorher installierten Version des Pakets überschreiben würde. Bei einer Erstinstallation muss diese aber erstellt werden. Das kann z.B. durch Kopieren der Standardkonfigurationsdatei erfolgen. (Siehe auch Kapitel Paketinstallation (Seite 25))

Die Standardkonfigurationsdatei liegt unter `/etc/default.d/`*$package*. Diese Datei muss in einem *eisfair*–Paket mitgeliefert werden. In ihr muss das Paket deaktiviert sein, d.h. die Startvariable `START_`*$PACKAGE* **muss** auf „no“ gesetzt sein.

Bei jedem Editieren einer Konfiguration wird mittels ACFH (Advanced Configuration File Handling (Seite 49)) eine Sicherheitskopie der letzten Konfiguration unter `/etc/backup.d/`*$package* erstellt. Die Verwaltung dieser Sicherheitskopien erfolgt mittels der ACFH–Funktionalität und muss nicht in jedem Paket einzeln implementiert werden.

6.2.1 Aufbau einer *eisfair*–Kofigurationsdatei

Beispiel für eine *eisfair*–Konfigurationsdatei (Paket foo):

```
# ----------------------------------------------------------------------
# /etc/config.d/foo - foo configuration parameters
#
# Creation:     2005-02-01  max
# Last Update:  2005-03-06  max
#
```

```
# Copyright (c) 2001-2014 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# General settings
# ----------------------------------------------------------------------

START_FOO='no'                          # start foo: yes or no

FOO_GENERAL_SETTING='60'                # time for foo
FOO_ANOTHER_SETTING='Test'              # text for foo

FOO_LONG_COMMNET='yes'                  # you can write long comments by
                                        # using more than one line...


# ----------------------------------------------------------------------
# Second Group
#
# You can give a detailed description of this group inside
# the comment
# ----------------------------------------------------------------------

FOO_ARRAY_N='3'                         # number of elements in array

FOO_ARRAY_1_ACTIVE='yes'                # is element active: yes or no
FOO_ARRAY_1_TEXT='Sample'               # some text for an array element
FOO_ARRAY_1_VALUE='12345'               # a number for this element

FOO_ARRAY_2_ACTIVE='no'                 # is element active: yes or no
FOO_ARRAY_2_TEXT='Inactive'             # some text for an array element
FOO_ARRAY_2_VALUE=''                    # a number for this element

FOO_ARRAY_3_ACTIVE='yes'                # is element active: yes or no
FOO_ARRAY_3_TEXT='Something'            # some text for an array element
FOO_ARRAY_3_VALUE='999'                 # a number for this element


# ----------------------------------------------------------------------
# End
# ----------------------------------------------------------------------
```

Damit die Konfiguration mittels eines *eisfair*–Konfigurationseditors bearbeitet werden kann, ist die Struktur der Konfigurationsdatei zwingend einzuhalten. Im Folgenden werden die einzelnen Blöcke näher beschrieben.

Generell werden spezielle Blöcke mittels der Trennzeilen „`#-------`“ gekennzeichnet.

6.2.2 Kopfbereich und Fussbereich

Vor der eigentlichen Konfiguration steht der Kopfbereich, der im Allgemeinen der GPL–Header ist und einen Verweis auf die GPL enthält, da wohl die meisten Pakete unter der GPL stehen werden.

Am einfachsten sollte der hier angegebene Kopfbereich einfach übernommen und nur die relevanten Inhalte angepasst werden.

Als Fussbereich wird das Ende der Datei mit „`# End`“ gekennzeichnet.

6.2.3 Gruppenstruktur

Die gesamte Konfiguration wird in einzelne Gruppen unterteilt. Jede Gruppe wird über den Gruppenkopf näher beschrieben. Dieser wird mittels einer Trennzeile „`#-------`“ eingeleitet und abgeschlossen.

Dazwischen steht in der ersten Zeile der *Gruppenname* und in den Folgezeilen ein *optionaler Kommentar* zur gesamten Gruppe. Dort können nähere Angaben zu den in der Gruppe zu konfigurierenden Funktionen gemacht werden (aber hier noch ohne Bezug zur einzelnen Konfigurationsvariable).

6.2.4 Konfigurationsvariablen

Innerhalb der einzelnen Gruppen stehen die Konfigurationsvariablen. Unter diesen nimmt die Startvariable `START_`*$PACKAGE* eine Sonderstellung ein. Durch diese Variable wird das gesamte Paket aktiviert bzw. deaktiviert.

Alle anderen Variablen haben die Form *$PACKAGE*`_OPTION`, d.h. jede der Variablen bekommt als Präfix den Paketnamen. Die anderen Namensbestandteile werden mittels des Unterstrichs „_“ abgetrennt.

Konfigurationsvariablen werden prinzipiell durchgehend gross geschrieben.

Jede Konfigurationsvariable kann mit einem eigenen Kommentar näher beschrieben werden. Dieser Kommentar kann dabei entweder hinter der Variable stehen (mittels mindestens eines Leerzeichens plus einem Kommentarzeichen „#“ abgetrennt) oder in der Folgezeile. In diesem Fall muss der Kommentar mittels eines oder mehrerer Leerzeichen eingerückt sein, das Kommentarzeichen darf also **nicht** in der ersten Spalte stehen.

Mehrzeilige Kommentare sind möglich, dabei dürfen die Kommentarzeichen in keiner Zeile in der ersten Spalte stehen.

Um Optionen anzugeben, die in einer variablen Anzahl auftreten können, müssen mehrere Felder verwendet werden. Diese bestehen aus einer Variable, die die Anzahl der Werte angibt (`FOO_ARRAY_N`) und den eigentlichen Optionen (`FOO_ARRAY_1_TEXT`, `FOO_ARRAY_1_VALUE`). Diese treten dann je nach angegebener Anzahl mehrfach auf.

Um einzelne Ausprägungen auch innerhalb der Liste ausblenden zu können, ohne jedesmal die gesamte Liste umnummerieren zu müssen, empfiehlt sich die Verwendung einer Aktivierungsvariable (`FOO_ARRAY_1_ACTIVE`). Diese muss dann im paketspezifischen Script geprüft werden.

Mehrdimensionale Felder können in der Form `FOO_ARRAY_1_SUBARRAY_2_VALUE`) angegeben werden. Dazu müssen dann je Ausprägung der ersten Dimension die Anzahl der Ausprägungen der zweiten Dimension angegeben werden (`FOO_ARRAY_1_SUBARRAY_N`).

6.2.5 Prüfen der Konfiguration

Sobald ein Anwender eine Konfiguration über das *eisfair*-Menü (über das edit-Script) geändert hat, wird diese auf Gültigkeit überprüft. Dazu wird das Programm eischk (Seite 51) benutzt, das im weiteren Verlauf dieses Kapitels näher beschrieben wird.

`/etc/check.d/`*$package*
`/etc/check.d/`*$package*`.exp`
`/etc/check.d/`*$package*`.ext`

In der Datei `/etc/check.d/`*$package* wird dabei die eigentliche Prüfung mit Abhängigkeiten und Feldern definiert. Dazu stehen die in `/etc/check.d/base.exp` definierten Regulären Ausdrücke zur Verfügung.

Sofern für die Überprüfung einer eigenen Variable die vordefinierten Ausdrücke nicht ausreichend sind, so können in der Datei `/etc/check.d/`*$package*`.exp` eigene Reguläre Ausdrücke definiert werden. Die Namen dieser eigenen Ausdrücke müssen dabei den Paketnamen als Präfix enthalten (Beispiel: *$PACKAGE*_VALUES).

In der Datei `/etc/check.d/`*$package*`.ext` können mit einer Scriptsprache zudem weitergehende Prüfungen implementiert werden.

6.2.6 Hilfedatei zur Konfiguration

In den neueren Konfigurationseditoren (ECE) können dem Anwender weitergehende Hilfetexte angezeigt werden. Diese müssen dazu in der Datei `/var/install/help/`*$package* hinterlegt werden.

Zeilenumbrüche werden in dieser Datei mit dem Tag `<br/>` gekennzeichnet, Kommentare können XML-Konform mittels `<!--` *Kommentar* `-->` eingefügt werden.

Der Hilfetext jeder einzelnen Variable wird mittels des Tags `<help>` in dieser Datei hinterlegt.

Die Datei hat damit folgendes Format:

Beispiel für eine *eisfair*–Hilfedatei (Paket foo):

```
<!--
# ----------------------------------------------------------------------
# /var/install/help/foo - helptexts for package foo
```

```
#
# Creation:     2005-02-01  max
# Last Update:  2005-03-06  max
#
# Copyright (c) 2001-2014 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
# ----------------------------------------------------------------------
-->
<help name="FOO_GENERAL_SETTING">
    Hier koennen Sie eine Zeitspanne fuer das Paket foo
    definieren.
<br/><br/>
    Default: 60
</help>

<!-- Dies ist ein Kommentar -->

<help name="FOO_ARRAY_%_ACTIVE">
    Die Konfigurationsvariable FOO_ARRAY_%_ACTIVE dient
    zur Aktivierung bzw. Deaktivierung der einzelnen
    Eintraege.
<br/><br/>
    Default: yes
</help>
```

6.3 Anwenden der Konfiguration

Nach erfolgreicher Prüfung der Konfiguration muss diese aktiviert („apply“) werden. Dieser Vorgang wird vollständig im allgemeinen Edit–Script gekapselt.

Für diesen Mechanismus müssen zwei paketspezifische Dateien bereitstehen:

`/var/install/config.d/`*$package*`.sh`
`/etc/init.d/`*$package*

Die Datei `/var/install/config.d/`*$package*`.sh` dient dabei zum Generieren der Konfiguration der im Paket verwendeten Software aus der *eisfair* Konfigurationsdatei. Es handelt sich hier also um das Apply im eigentlichen Sinne.

ACHTUNG

Da eischk die Konfiguration nicht überprüft, wenn `START_`*$PACKAGE*`='no'` gesetzt ist, darf in diesem Fall **kein** Neuschreiben der Konfiguration erfolgen!

Dieses Skript muss im Erfolgsfall den Wert 0 zurückgeben (`exit 0`). Im Fehlerfall muss ein Wert ungleich 0 zurückgegeben werden. Dies bewirkt, dass der Neustart des Basisdienstes über `/etc/init.d/`*$package* **nicht** ausgeführt wird.

Im Script `/etc/init.d/`*$package* muss der Befehl `restart` unterstützt werden. Dieser Befehl wird beim Aufruf als einziger Parameter übergeben. Bei Aufruf mit `restart` muss der Dienst neu gestartet bzw. es muss die Konfiguration so eingelesen werden, dass diese wirksam wird.

Weitere Details dazu können in den Kapiteln Menü (Seite 68) sowie Bootprozess (Seite 40) nachgeschlagen werden.

6.4 ACFH - Advanced Configuration File Handling

6.4.1 ACFH Grundlagen

Das Advanced Configuration File Handling (ACFH) bietet die Möglichkeit, Generationen von Konfigurationsdateien in einem Verzeichnis (`/etc/backup.d/`) sichern zu lassen. Dies geschieht bei Nutzung des Skripts `/var/install/bin/edit` automatisch.

Die gesicherten Konfigurationsdateien tragen den Namen

$package`.YYYY-MM-DD-HH-MI-SS`,

also z.B. `base.2005-03-08-17-57-24`.

Die Anzahl der vorzuhaltenden Generationen von Konfigurationsdateien wird bestimmt durch den Wert der Environmentvariablen `MAX_BACKUP_CONFIG` in der Datei `/etc/config.d/environment`. Übersteigt die Anzahl der Konfigurationsdateien für ein Paket die in `MAX_BACKUP_CONFIG` angegebene Anzahl, so wird jeweils die älteste Konfigurationsdatei gelöscht.

Über ein vorbereitetes Menü des ACFH und entsprechende Skripts lassen sich die Funktionen des ACFH aufrufen. An der Stelle base steht jeweils der entsprechende Paketname.

Beispiel:

```
Advanced base-configuration file handling.

        1: Restore configuration file from default directory

        2: Restore configuration file from backup directory

        3: Backup configuration file to backup directory

        4: Show difference between current and default configuration

        5: Show difference between current and a backup configuration

        6: Show difference between default and a backup configuration
```

```
        7: Show difference between two backup configurations

        0: Exit

Select (1-7, ENTER=Return 0=Exit)
```

Die einzelnen Menüpunkte können allerdings nur dann erfolgreich aufgerufen werden, wenn die entsprechenden Konfigurationsdateien vorhanden sind.

6.4.2 Einbinden von ACFH

Die Einbindung von ACFH in ein Paket ist mit dem neuen Menüformat (XML-Menü) sehr einfach geworden.

Um für ein Paket das ACFH-Untermenü aufrufen zu können sind nur zwei Zeilen im entsprechenden Paket–Menü notwendig.

```
<package>$package</package>
<menu file=setup.services.advancedconfig.menu">
           Advanced configuration file handling</menu>
```

Das `<menu>`–Tag muss in einer Zeile stehen.

Über das `<package>`–Tag wird einerseits die Anzeige im Untermenü (siehe Beispiel oben paketname=base) gesteuert, andererseits auch für die aufzurufenden Skripte eingestellt, um welches Paket und damit um welche Konfigurationsdateien es sich handelt.

Der über das `<package>`–Tag eingestellt Paketname wird in der Environment-variablen $PACKAGE gespeichert. Diese Variable darf nicht anderweitig benutzt oder gar gelöscht werden. Daher steht im ACFH-Untermenü selbst (Datei `setup.services.advancedconfig.menu`) kein `<package>`–Tag.

6.4.3 Menüs mit mehr als einem ACFH Untermenü

Eine spezielle Situation entsteht, wenn von einem Menü aus mehr als ein ACFH Untermenü für verschiedene Konfigurationsdateien in `/etc/config.d/` aufgerufen werden soll.

Dies ist derzeit z.B. im Menü „Base configuration“ der Fall. Hier ein Ausschnitt aus dem Menü:

```
        3: Advanced base configuration file handling

        4: Advanced environment configuration file handling
```

Hier wird das ACFH Untermenü sowohl für die Datei `/etc/config.d/base`, als auch für `/etc/config.d/environment` aufgerufen.

Über das <package>–Tag ist base als Paketname eingestellt. D.h. der Aufruf für base kann wie oben beschrieben erfolgen.

Für die Behandlung der Datei /etc/config.d/environment muss die Option package="" des <menu>–Tag genutzt werden.

```
<menu package="environment" file="setup.services.advancedconfig.menu">
              Advanced environment configuration file handling</menu>
```

Das <menu>–Tag muss in einer Zeile stehen.

Um die Quelldatei des Menüs einheitlich zu gestalten, wurde auch der Aufruf des Menüs für die Datei /etc/config.d/base mit der optionalen Option package="" versehen.

```
<menu package="base" file="setup.services.advancedconfig.menu">
               Advanced base configuration file handling</menu>
<menu package="environment" file="setup.services.advancedconfig.menu">
               Advanced environment configuration file handling</menu>
```

Das <menu>–Tag muss in einer Zeile stehen.

6.5 Eischk

Um die Konfigurationsfehler in Paketen möglichst auszuschliessen wird für jedes Paket, welches eine Config–Datei besitzt, auch eine EISCHK–Datei angelegt: /etc/check.d/*$package*. In ihr sind alle Variablen der Konfigurationsdatei einzutragen. Der Aufbau gliedert sich dabei in 4 Spalten, die wie folgt belegt sind:

6.5.1 VARIABLE

Diese Spalte gibt den Namen der zu überprüfenden Variable aus der Config–Datei an. Wenn es sich dabei um eine Variable handelt, die mehrmals mit verschiedenen Nummern auftauchen kann, wird an Stelle der Nummer ein Prozentzeichen (%) als Platzhalter in den Variablennamen eingefügt. Mehrere Prozentzeichen zur Überprüfung mehrdimensionaler Arrays sind zulässig. Sollten Variablen nur unter bestimmten Konfigurationsbedingungen benötigt werden, so sind sie als Optional zu kennzeichnen. Dazu wird vor die Variable ein „+“ Zeichen vorangestellt. Soll ein gesamtes Array optional sein, dann ist der Zählvariable (z.B. FOO_N) ebenfalls ein „+“ voranzustellen. In diesem Fall müssen aber auch die Array–Elemente als optional gekennzeichnet werden.

Beispiele:

```
optionale Variable:
# Variable:           OPT_VARIABLE:  VARIABLE_N:    VALUE:
+FOO_NAME             -              -              NONE

optionales Element in einem Array:
FOO_N                 -              -              NUMERIC
++FOO_%_VALUE1        -              FOO_N          NONE
```

```
FOO_%_VALUE2          -              FOO_N          NONE

optionales Array:
+FOO_N                -              -              NUMERIC
++FOO_%_VALUE1        -              FOO_N          NONE
++FOO_%_VALUE2        -              FOO_N          NONE

optionales mehrdimensionales Array:
+FOO_N                -              -              NUMERIC
++FOO_%_VALUE_N       -              FOO_N          NUMERIC
++FOO_%_VALUE_%_VAL1  -              FOO_%_VALUE_N  NONE
```

6.5.2 OPT_VARIABLE

Bei Bedarf kann in dieser Spalte eine Abhängigkeit zu einer anderen Variable (`OPT_VARIABLE`) definiert werden. Dadurch findet eine Überprüfung nur dann statt, wenn die `OPT_VARIABLE` auf „`yes`“ steht. Der Name der `OPT_VARIABLE` darf keine Platzhalter für Arrays (%) enthalten. Gibt es keine `OPT_VARIABLE`, ist hier ein „–“ anzugeben.
Ab dem base-Paket v1.3.0 dürfen Platzhalter für Arrays (%) enthalten sein.

Beispiel:

```
# Variable:                  OPT_VARIABLE:                VARIABLE_N:      VALUE:
-------------------------------------------------------------------------------------
EISFAX_ANALOG_USE            START_EISFAX                 -                YESNO
EISFAX_ANALOG_N              EISFAX_ANALOG_USE            -                NUMERIC
EISFAX_ANALOG_%_NAME         EISFAX_ANALOG_USE            EISFAX_ANALOG_N  NONE
EISFAX_ANALOG_%_ACTIVE       EISFAX_ANALOG_USE            EISFAX_ANALOG_N  YESNO
EISFAX_ANALOG_%_DEVICE       EISFAX_ANALOG_%_ACTIVE       EISFAX_ANALOG_N  NOTEMPTY
EISFAX_ANALOG_%_SND          EISFAX_ANALOG_%_ACTIVE       EISFAX_ANALOG_N  YESNO
EISFAX_ANALOG_%_SND_TO       EISFAX_ANALOG_%_SND          EISFAX_ANALOG_N  NONE
EISFAX_ANALOG_%_SND_TYPE     EISFAX_ANALOG_%_SND          EISFAX_ANALOG_N  EISFAX_FT
EISFAX_ANALOG_%_PRN          EISFAX_ANALOG_%_ACTIVE       EISFAX_ANALOG_N  YESNO
EISFAX_ANALOG_%_PRN_QUE      EISFAX_ANALOG_%_PRN          EISFAX_ANALOG_N  EISFAX_PQ
EISFAX_ANALOG_%_PRN_TYPE_GS  EISFAX_ANALOG_%_PRN          EISFAX_ANALOG_N  YESNO
EISFAX_ANALOG_%_PRN_DRV      EISFAX_ANALOG_%_PRN_TYPE_GS  EISFAX_ANALOG_N  NONE
```

6.5.3 VARIABLE_N

Steht in der ersten Spalte eine Variable mit einem Platzhalter % im Namen, so wird hier die Variable angegeben, die die Häufigkeit des Auftretens der Variable definiert. Für mehrdimensionale Array sind auch hier % Zeichen im Namen zulässig. Gibt es keine `VARIABLE_N`, ist hier ein „–“ anzugeben.

Beispiel 1:

```
# Variable:           OPT_VARIABLE:  VARIABLE_N:    VALUE:
BIND9_N               -              -              NUMERIC
BIND9_%_NAME          -              BIND9_N        DOMAIN
BIND9_%_MASTER        -              BIND9_N        YESNO
BIND9_%_NETWORK       -              BIND9_N        IPADDR
```

Beispiel 2 - mehrdimensionale Arrays mit optionalen Parametern:

```
# Variable:           OPT_VARIABLE:  VARIABLE_N:    VALUE:
++BIND9_%_NS_N        -              BIND9_N        NUMERIC
++BIND9_%_NS_%_NAME   -              BIND9_%_NS_N   FQDN
```

6.5.4 VALUE

Diese Spalte enthält die Überprüfungrichtlinie des Inhaltes der Variable. Sie sollte im Interesse einer geringen Fehlerquote möglichst exakt die Eingabemöglichkeiten eingrenzen. Dabei kann auf eine grosse Anzahl von fertigen Definitionen zurückgegriffen, aber auch mittels sogenannter „Regular Expressions“ eigene erstellt werden.

Folgende Definitionen sind bereits vorhanden:

Name	Bedeutung	Beispiele
NONE	Keine Überprüfung durchführen	
NOTEMPTY	Inhalt darf nicht leer sein	’foo- 12’
NOBLANK	Es dürfen keine Leerzeichen enthalten sein	’foo’
ENOBLANK	Kein Eintrag - oder Eintrag wie unter NOBLANK	” / ’foo’
NUMERIC	Nur Zahlen sind zulässig	’1234’
ENUMERIC	Kein Eintrag - oder Eintrag wie unter NUMERIC	” / ’1234’
DOT_NUMERIC	Zwei Zahlen durch einen Punkt getrennt	’12.5’
EDOT_NUMERIC	Kein Eintrag - oder Eintrag wie unter DOT_NUMERIC	” / ’12.5’
NUM_HEX	Hexadezimale Zahl - startend mit 0x	’0xff12’
NUM_ANY	Zahlen in numerischer oder hexadezimaler Schreibweise	’123’ / ”0xff12’
YESNO	Nur yes oder no sind erlaubt	’yes’ / ’no’
MACADDR	Mac Adresse einer Netzwerkkarte	’00:00:E8:83:72:92’
HOSTNAME	Hostname bestehend aus Buchstaben, Zahlen und Bindestrich	’foo-1’
DOMAIN	Domainname inclusive Endung	’bar.local’
EDOMAIN	Kein Eintrag - oder Eintrag wie unter DOMAIN	” / ’bar.local’
FQDN	Vollständiger Internetname - host.domain.location	’foo-1.bar.local’
EFQDN	Kein Eintrag - oder Eintrag wie unter FQDN	” / ’foo-1.bar.local’
OCTET	Zahl im Bereich von 0 bis 255	’128’
IPADDR	Gültige IP-Adresse, bestehend aus 4x OCTET	’192.168.0.128’

Name	Bedeutung	Beispiele
EIPADDR	Kein Eintrag - oder Eintrag wie unter IPADDR	” / ’192.168.0.128’
IPADDRESSES	Eine oder mehrere IP-Adressen durch Leerzeichen getrennt	’192.168.0.1 192.168.0.2 192.168.0.3’
EIPADDRESSES	Kein Eintrag - oder Eintrag wie unter IPADDRESSES	” / ’192.168.0.1 192.168.0.2 192.168.0.3’
IP_ROUTE	Netzwerk, Netzmaske und Gateway durch Leerzeichen getrennt	’192.168.0.0 255.255.255.0 192.168.0.254’
DNS_SPEC	DOMAIN und IP-Adresse durch Leerzeichen getrennt	’bar.local 192.168.0.1’
MASK	Zahl zwischen 0 und 32	’32’
NETWORK	Kombination aus IP-Adresse, Trennstrich und Maske	’192.168.0.0/24’
NETWORKS	Ein oder mehrer Netzwerke durch Leerzeichen getrennt	’192.168.0.0/24 192.168.1.0/24’
ENETWORKS	Kein Eintrag - oder Eintrag wie unter NETWORKS	” / ’192.168.0.0/24 192.168.1.0/24’
MULTIPLE_NETWORKS	Ein oder mehrer Netzwerke durch Leerzeichen getrennt	’192.168.0.0/24 192.168.1.0/24’
EMULTIPLE_NETWORKS	Kein Eintrag - oder Eintrag wie unter MULTIPLE_NETWORKS	” / ’192.168.0.0/24 192.168.1.0/24’
IPADDR_NETWORK	IP-Adresse und Netzwerk durch Leerzeichen getrennt	’192.168.0.1 192.168.1.0/24’
EIPADDR_NETWORK	Kein Eintrag - oder Eintrag wie unter IPADDR_NETWORK	” / ’192.168.0.1 192.168.1.0/24’
MAILADDR	Vollständige E-Mail Adresse	’foo@bar.local’
EMAILADDR	Kein Eintrag - oder Eintrag wie unter MAILADDR	” / ’foo@bar.local’
CRONTAB	Crontab Eintrag: Minute Stunde Tag Monat Wochentag	’15 3 * * fri’
REL_PATH	Relatives Verzeichnis	’install/menu’
E_REL_PATH	Kein Eintrag - oder Eintrag wie unter REL_PATH	” / ’install/menu’
ABS_PATH	Absolutes Verzeichnis	’/var/install/menu’
E_ABS_PATH	Kein Eintrag - oder Eintrag wie unter ABS_PATH	” / ’/var/install/menu’
LOG_INTERVAL	Als Werte sind daily, weekly oder monthly möglich	’daily’ / ’weekly’ / ’monthly’
PORT	Kommunikationsport im Bereich von 1 - 65535	’3050’

Name	Bedeutung	Beispiele
ETH_BASE_DEV_NAME	Name der Ethernet Schnittstelle	'eth0'
ETH_DEV_NAME	Erweiterter Name der Ethernet Schnittstelle	'eth0:3
TR_BASE_DEV_NAME	Name der Tokenring Schnittstelle	'tr2'
TR_DEV_NAME	Erweiterter Name der Tokenring Schnittstelle	'tr2:1'
BR_DEV_NAME	Netzwerk Bridge Schnitstelle	'br1:2'
BOND_BASE_DEV_NAME	Netzwerk Bonding Schnitstelle	'bond0'
BOND_DEV_NAME	Erweiterter Name der Netzwerk Bonding Schnitstelle	'bond0:1'
XEN_BASE_DEV_NAME	XEN Bridge Netzwerk Schnitstelle	'xen-br0'
XEN_BASE_NAME	Erweiterter Name der XEN Bridge Netzwerk Schnitstelle	'xen-br0:1'
DUMMY_DEV_NAME	Platzhalter Schnittstelle (Dummy)	'dummy0'
IP_NET_NAME	Ethernet, Tokenring, Bridge oder Dummy Schnittstelle	'eth0:2' / 'br1' / 'tr0' / 'dummy0'
IP_ROUTE	Netzwerkroute (network netmask gateway)	'192.168.1.0 255.255.255.0 192.168.1.254'
IP_ROUTE_CIDR	Netzwerkroute (network/cidr gateway)	'192.168.1.0/24 192.168.1.254'
PASSWD	Spezielle Regel, die die Sichtbarkeit im ECE beeinflusst. In der Ansicht wird der Wert der Option mit dem Zeichen '*' maskiert. Die Werteeingabe erfolgt über einen speziellen Dialog zur Passworteingabe.	'**********'
READONLY	Spezielle Regel, die die Sichtbarkeit im ECE beeinflusst. In der Ansicht wird der Wert der Option inaktiv dargestellt. Eine Bearbeitung des Wertes wird vom Programm unterbunden.	
HIDDEN	Spezielle Regel, die die Sichtbarkeit im ECE beeinflusst. In der Ansicht wird der Wert nicht dargestellt.	

Da auch mit diesen Möglichkeiten nicht jeder Anwendungsfall abzudecken ist, gibt es die Möglichkeit, eigene Überprüfungen zu definieren. Diese werden durch die Buchstaben „`RE:`“ für Regular Expressions eingeleitet.

Beispiel:

```
# Variable:          OPT_VARIABLE:  VARIABLE_N:    VALUE:
COLOR                -              -              RE:red|green|blue
```

So kann eine einfache Parameterauswahl realisiert werden. Alle anderen Eingaben als „red“, „green“ oder „blue“ werden abgelehnt. Ein komfortablerer Weg stellt der Einsatz einer `/etc/check.d/`*$package*`.exp` Datei dar. In ihr können neben den Regular Expressions ei-

gene Fehlermeldungen definiert werden.

ACHTUNG

Eigene Regular Expressions in der Datei `/etc/check.d/`*$package*`.exp` müssen den Paketnamen als Präfix enthalten.

Beispiel:

```
FOO_COLOR   = 'red|green|blue' : 'only red, green or blue are allowed'
```

In den regulären Ausdrücken können auch Referenzen auf bereits existierende Definitionen enthalten sein. Dadurch ist es einfacher, reguläre Ausdrücke zu konstruieren. Eingefügt werden die Referenzen einfach durch `'(RE:Referenz)'`.

Beispiel:

```
FOO_IP_ROUTE    = '(RE:IPADDR)[[:space:]]+(RE:IPADDR)[[:space:]]+(RE:IPADDR)'
                  : 'no valid route specification (network netmask gateway)'

FOO_DOT_NUMERIC = '(RE:NUMERIC).(RE:NUMERIC)'
                  : 'should be numeric (decimal) with dot e.g. 5.0'
```

Die speziellen Regeln PASSWD, READONLY und HIDDEN sind lediglich für den ECE von Bedeutung. Für eischk stellen sie keine Einschränkung des Wertebereichs dar. Deshalb werden sie in der Datei `/etc/check.d/`*$package* häufig in Kombination mit anderen Regeln verwendet. Beispiel:

Beispiel:

```
# Variable:           OPT_VARIABLE:  VARIABLE_N:    VALUE:
FOO_DEBUG             -              -              YESNO
FOO_DEBUG             -              -              HIDDEN
```

6.5.5 Fehlermeldungen

Findet die Prüfung einen Fehler, erfolgt die Ausgabe der Fehlermeldung auf folgende Art:

```
Error: wrong value of variable HOSTNAME: '' (may not be empty)
Error: wrong value of variable MOUNT_OPT: 'rx' (user supplied regular expression)
```

Beim ersten Fehler wurde der Ausdruck in einem exp file definiert und ein Hinweis auf den Fehler wird mit ausgegeben. Im zweiten Falle wurde der Ausdruck direkt im *.txt File spezifiziert, deshalb gibt es keinen zusätzlichen Hinweis auf die Fehlerursache.

6.5.6 Definition regulärer Ausdrücke

Reguläre Ausdrücke sind wie folgt definiert:

Regulärer Ausdruck: Eine oder mehrere Alternativen, getrennt durch '|', z.B. 'ro|rw|no'. Trifft eine der Alternativen zu, trifft der ganze Ausdruck zu (hier wären 'ro', 'rw' und 'no' gültige Ausdrücke).

Eine Alternative ist eine Verkettung mehrerer Teilstücke, die einfach aneinandergereiht werden.

Ein Teilstück ist ein „Atom", gefolgt von einem einzelnen '*', '+', '?' oder '{min, max}'. Die Bedeutung ist wie folgt:

- 'a*' - beliebig viele a's einschliesslich kein a
- 'a+' - mindestens ein a, sonst beliebig viele a's
- 'a?' - kein oder ein a
- 'a{2,5} - zwei bis 5 a's
- 'a{5} - 5 a's
- 'a{2,} - mindestens 2 a's

Ein „Atom" ist ein

- regulärer Ausdruck eingeschlossen in Klammern, z.B. (a|b)+ trifft auf eine beliebige Zeichenkette zu, die mindestens ein a oder b enthält, sonst aber beliebig viele und in beliebiger Reihenfolge
- ein leeres Paar Klammern steht für einen „leeren" Ausdruck
- ein Ausdruck mit eckigen Klammern '[]' (siehe weiter unten)
- ein Punkt '.', der auf irgend ein einzelnes Zeichen zutrifft, z.B. '.+' trifft auf eine beliebige Zeichenkette zu, die mindestens ein Zeichen enthält
- ein '^' steht für den Zeilenanfang, z.B. '^a.*' trifft auf eine Zeichenkette zu, die mit einem a anfängt und in der beliebige Zeichen folgen, 'a' oder 'adkadhashdkash'
- ein '$' steht für das Zeilenende
- ein '
  ' gefolgt von einem der Sonderzeichen '^.[$()|*+?{'' steht für genau das zweite Zeichen ohne seine spezielle Bedeutung
- ein normales Zeichen trifft auf genau das Zeichen zu, z.B. 'a' trifft auf genau 'a' zu.

Ein Ausdruck mit rechteckigen Klammern bedeutet folgendes

- '[x-y]' - trifft auf irgend ein Zeichen zu, das zwischen x und y liegt, z.B. '[0-9]' steht für alle Zeichen zwischen 0 und 9; '[a-zA-Z]' für alle Buchstaben, egal ob gross oder klein
- '[^x-y]' - trifft auf irgendein Zeichen zu, das nicht im angegebenen Intervall liegt
- '[:character_class:] - trifft auf ein Zeichen der Zeichen-Klasse zu. Relevante Standardzeichenklassen sind: alnum, alpha, blank, digit, lower, print, punct, space, upper, xdigit.

6.5.7 Beispiele für Reguläre Ausdrücke

Sehen wir uns das mal an einigen Beispielen an:

Numerisch: Ein numerischer Wert besteht aus mindestens einer, aber beliebig vielen Zahlen. Mindestens ein, aber beliebig viele drückt man mit '+' aus, eine Zahl hatten wir schon als Beispiel. Zusammengesetzt ergibt das:

```
NUMERIC = '[0-9]+' oder alternativ
NUMERIC = '[[:digit:]]+'
```

NOBLANK: Ein Wert, der keine Leerzeichen enthält ist ein beliebiges Zeichen (ausser dem Leerzeichen) und davon beliebig viele:

```
NOBLANK = '[^ ]*'
```

bzw. wenn der Wert zusätzlich auch nicht leer sein soll:

```
NOBLANK = '[^ ]+'
```

Disk und Partition: Gültige Bezeichner für eine Disk beginnen mit hd bei IDE-Disks bzw sd bei SCSI-Disks. Dann folgen Buchstaben von a-z (a für die erste Disk, b für die 2., ...) und bei Partitionen die Zahlen 1-8 (1-4 für die ersten 4 Partitionen, die Primär bzw. Extended (nur eine) sein können und 5-8 für die logischen Partitionen innerhalb einer extended Partition). Die Ausdrücke sehen dann wie folgt aus:

```
DISK      = '(hd|sd)[a-z]'
PARTITION = '(hd|sd)[a-z][1-8]'
```

Sehen wir uns das ganze nochmal am Beispiel der IP-Addresse an. Eine IP-Adresse besteht aus 4 Octets, die mit einem '.' getrennt sind. Ein Octet kann eine Zahl zwischen 0 und 255 sein. Definieren wir als erstes ein Octet. Es kann

eine Zahl zwischen 0 und 9 sein:	[0-9]
eine Zahl zwischen 00 und 99:	[0-9][0-9]
eine Zahl zwischen 100 und 199:	1[0-9][0-9]
eine Zahl zwischen 200 und 249:	2[0-4][0-9]
eine Zahl zwischen 250 und 255 sein:	25[0-5]

Da sich die ersten drei Teile stark ähneln, kann man sie zusammenfassen:

- ein Zahl zwischen 0 und 199: 1?[0-9]?[0-9] (die ersten beiden Stellen können, aber müssen nicht da sein)

Das ganze sind Alternativen, also fassen wir sie einfach mittels '|' zu einem Ausdruck zusammen: '1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]' und haben damit ein Octet. Daraus können wir nun eine IP-Adresse machen, 4 Octets mit Punkten voneinander getrennt (der Punkt muss mittels *backslash* gequotet werden, da er sonst für ein beliebiges Zeichen steht). Basierend auf der Syntax der Exp-Files sieht das ganze dann wie folgt aus:

```
OCTET  = '1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]'
IPADDR = '((RE:OCTET)\.){3}(RE:OCTET)'
```

6.5.8 Erweiterte Prüfungen der Konfiguration

Manchmal ist es notwendig, komplexere Überprüfungen durchzuführen. Beispiele für solche komplexeren Dinge wären z.B. Abhängigkeiten zwischen Paketen oder Bedingungen, die nur erfüllt sein müssen, wenn Variablen bestimmte Werte annehmen.

Um diese Überprüfungen durchführen zu können, kann man in `/etc/check.d/`*$package*`.ext` kleinere Tests schreiben. Die Sprache besteht aus folgenden Elementen:

1. Schlüsselwoerter:
   - Kontrollfluss:
     - `if (expr) then statement else statement fi`
     - `foreach var in set_var do statement done`
     - `foreach var in var_n do statement done`
   - Aktionen:
     - `warning` „*warning*"
     - `error „error"`
     - `fatal_error „fatal error"`
     - `set var = value`
     - `stat (filename, res)`
     - `split (string, set_variable, character)`
2. Datentypen: strings, numerische Werte, Versionsnummern, Arrays
3. Logische Operationen: `<, ==, >, !=, !, &&, ||, =`

Kommunikation mit dem Nutzer: warning, error, fatal_error

Mit Hilfe dieser drei Funktionen kann man Nutzer warnen, einen Fehler signalisieren oder die Prüfung sofort abbrechen. Das Format sieht wie folgt aus:

- `warning „text"`
- `error „text"`
- `fatal_error „text"`

Im Text kann man auf Variablen Bezug nehmen, indem man einfach den Namen mit einem vorangestellten '$' oder '%' Zeichen in den Text schreibt (evtl. in eingeschlossen, um den Variablenamen vom umgebenden Text abzugrenzen). eischk versucht den darauffolgenden Text (Ziffern, Zahlen, '_') als Variablennamen zu interpretieren und setzt bei einem vorangestellten '$' den Inhalt der Variablen an dieser Stelle ein, wenn Sie definiert ist, bzw bei einem vorangestellten '%' den vollständigen Namen der aktuellen %-Variable. Sonst steht

der originale Text da. Will man wirklich ein '$' oder ein '%' im Text haben, schreibt man '$$' bzw. '%%'.

Zuweisungen

Benötigt man aus irgend einem Grund eine temporäre Variable, kann man diese einfach mit „`set var [= value]`" anlegen. Die Variable darf keine Config–Variable sein. Lässt man den „`= value`" Teil weg, wird die Variable einfach auf „yes" gesetzt, so dass man sie hinterher einfach in einer if-Anweisung testen kann. Wird ein Zuweisungsteil angegeben, kann hinter dem Gleichheitszeichen alles stehen, normale Variablen, indizierte Variablen, Zahlen, Zeichenketten, Versionen.

Arrays

Will man auf einzelne Elemente einer %–Variablen (eines Arrays) zugreifen, kann man das wie gewohnt mit %_var[index] tun, wobei für jedes % Zeichen ein [Index] auftauchen muss.

Abfragen von Eigenschaften einer Datei – stat

`stat()` ermöglicht es, Eigenschaften einer Datei abzufragen. Zur Verfügung gestellt wird im Augenblick lediglich die Grösse einer Datei, andere Attribute sind aber leicht hinzuzufügen. Abfragen sehen wie folgt aus (wobei die verwendeten Parameter nur Beispiele sind):

```
stat ("unix/Makefile", test)
```

Danach sind zwei Variablen definiert:

- `test_res`: Resultat des Systemrufs `stat()` („OK", wenn Systemruf erfolgreich, sonst Fehlermeldung des Systemrufs)
- `test_size`: Grösse der Datei

Das könnte dann z.B. so aussehen:

```
stat ("unix/Makefile", test)
if ("$test_res" == "OK")
then
        warning "test_size = $test_size"
else
        error "Error '$test_res' while trying to get size of ..."
fi
```

Auseinandernehmen von Parametern - split

Oftmals werden Variablen mit mehreren Parametern belegt, die dann in Startup–Scripten erst wieder auseinandergenommen werden. Will man diese bereits vorher auseinandernehmen und Tests mit ihnen durchführen, nimmt man `split()`.

```
split (string, %_var, character)
```

Der String kann durch eine Variable, %–Variable oder direkt als String angegeben werden. eischk zerlegt ihn an den Stellen, an denen das Trennzeichen auftaucht und erzeugt je eine Instanz der %–Variablen. Über diese kann man dann hinterher Tests laufen lassen. Steht zwischen zwei Trennzeichen nichts, wird eine Instanz mit einer leeren Zeichenkette als Wert erzeugt. Ausnahme ist ' ', hier werden alle whitespaces konsumiert und keine leeren Variablen erzeugt.

Sollen die bei der Zerlegung entstandenen Elemente in einem numerischen Kontext verwendet werden (z.B. als Index), muss das beim Aufruf von Split spezifiziert werden. Das geschieht durch das zusätzliche Attribut 'numeric'. Der Aufruf sieht dann wie folgt aus:

```
split (string, %_var, character, numeric)
```

Kontrollfluss

```
if (expr)
then
        statement
else
        statement
fi
```

Eine klassische `if`–Konstruktion wie man sie kennt. Ist die Bedingung wahr, wird der `then`–Teil ausgeführt, ist die Bedingung falsch, wird der `else`–Teil ausgeführt.

Will man Tests über %–Variablen durchführen, muss man jede einzelne Variable testen. Dazu gibt es das `foreach`–Statement in zwei Varianten:

```
foreach loop_var in set_var
do
        statement
done
```

Diese Schleife iteriert über alle %–Variablen, angefangen bei eins bis zum in der dazugehoerigen (in `/etc/check.d/`*$package* stehenden) Variable_N stehenden Index n. Die Laufvariable loop_var nimmt dabei die jeweiligen Werte der %-Variablen an.

Das Statement kann dabei eine der oben beschriebenen Funktionen `if`, `foreach`, `warning`, `error` oder `fatal_error` sein.

Will man genau eine %–Variable testen, kann man diese mittels `var_%`*$index* auswählen. Der index kann dabei eine normale Variable, ein String oder wiederum ein indiziertes Array sein.

```
foreach loop_var in var_n
do
        statement
done
```

Diese Schleife laeuft von 1 bis var_n. Man kann die Schleifenvariable loop_var dazu benutzen, um _% Variablen zu indizieren. Will man also nicht nur über eine _% Variable iterieren,

sondern über mehrere gleichzeitig, nimmt man diese Variante der Schleife und verwendet die loop_var zum Indizieren mehrerer _% Variablen.

Expressions

Die Expressions erlauben so gut wie alles, was man von einer Programmiersprache gewöhnt ist. Ein in einem Ausdruck auftauchender Wert 'val' kann eine Variable, ein String oder ein indiziertes Array sein. Variablen in Strings werden dabei wie oben beschrieben ersetzt. Ein Test auf die Gleichheit zweier Variablen könnte also so aussehen:

```
var1 == var2
"$var1" == "$var"
```

Zu beachten ist dabei, dass der Vergleich in Abhängigkeit vom Typ der Variable erfolgt, der in `/etc/check.d/`*$package* festgelegt wurde. Variablen, die als Typnamen einen mit NUM beginnenden Namen haben, erhalten einen numerischen Typ. Ist einer der beiden Variablen numerisch, erfolgt der Vergleich auf numerischer Basis, d.h. die Zeichenketten werden in Zahlen umgewandelt und dann vergleichen. Sonst erfolgt der Vergleich auf String–Basis; ein Vergleich von '05' und '5' geht ergibt ungleich, ein Vergleich von '18' und '9' ergibt '18' < '9'.

Für den Vergleich von Versionen wird das Hilfskonstrukt numeric(version) eingeführt, welches den numerischen Wert für einen Versionsstring für Vergleichszwecke bestimmt . numeric(version) == sub_version + 1000 * minor_version + 10000 * major_version:

Eine vollständige Auflistung aller Ausdrücke ist in der folgenden Tabelle zu finden.

`expr`	true if
`id`	id == 'yes'
`val == val`	strings/numerische Werte sind identisch
`val != val`	strings/numerische Werte sind verschieden
`val == number`	numerischer Wert von val == number
`val != number`	numerischer Wert von val != number
`val < number`	numerischer Wert von val < number
`val > number`	numerischer Wert von val > number
`val == version`	numeric(val) == numeric(version)
`val < version`	numeric(val) < numeric(version)
`val > version`	numeric(val) > numeric(version)
`( expr )`	Ausdruck in Klammern ist wahr
`expr && expr`	beide Ausdrücke sind wahr
`expr || expr`	mind. einer der beiden Audrücke ist wahr

6.6 Selbstdefinierte Dialoge im ECE

Um die Eingabe von Werten in der eisfair-Konfiguration noch einfacher und sicherer zu gestalten, sieht der Eisfair Konfigurationseditor (ECE) eine Möglichkeit vor, die einfache Eingabezeile zur Parameterbearbeitung gegen selbstdefinierte Dialoge zu tauschen. In solchen

Dialogen können Werte z.B. als Auswahllisten oder in jeder anderen denkbaren Form dem Anwender verfügbar gemacht werden.

Damit der ECE einen Dialog verwendet müssen die folgenden Bedingungen erfüllt sein:

Zum einen muss in der Datei `/etc/check.d/`*$package* eine Prüfregel eingetragen werden, die der Variablen einen eindeutigen symbolischen Typnamen zuordnet. Kann das mit einer speziellen Typprüfung verbunden werden, dann steht ein solcher Name oftmals ohnehin zur Verfügung, anderenfalls muss eine "leere" Prüfregel definiert werden. Beispiel:

Beispiel für eine Datei `/etc/check.d/`*$package* für einen ECE-Dialog :

```
FOO_NETWORK        -      -           IPADDR
FOO_NETWORK        -      -           FOO_IP
FOO_DIR_N          -      -           NUMERIC
FOO_DIR_%_ACCESS   -      FOO_DIR_N   FOO_ACCESS
```

In dem oben aufgeführten Beispiel wird angenommen, dass ein Paket FOO für die Konfigurationsvariable FOO_NETWORK einen speziellen Dialog zur Eingabe oder Auswahl einer IP-Adresse definieren möchte. Als Prüfregel kommt dabei die in der Basis definierte Regel IPADDR zum Einsatz, weshalb eine weitere Regel für die selbe Variable eingefügt wurde, die lediglich einen Bezeichner (FOO_IP) definiert (IPADDR wäre nicht eindeutig!).

Zudem kennt die Konfiguration des FOO-Pakets die Variable FOO_DIR_%_ACCESS, welche ebenfalls einen Dialog erhalten soll. Hier gibt es aber bereits die paketspezifische Prüfregel FOO_ACCESS, die eindeutig genug ist, um direkt als Bezeichner verwendet werden zu können.

Selbstverständlich müssen für FOO_IP und FOO_ACCESS ebenfalls Einträge in der Datei `/etc/check.d/`*$package*`.exp` angelegt werden.

Beispiel für eine Datei `/etc/check.d/`*$package*`.exp` für einen ECE-Dialog :

```
FOO_IP                  = '(RE:NONE)'
                        : ''
FOO_ACCESS              = 'none|read|write'
                        : 'only none, read or write are allowed'
```

Zu guter letzt können nun im Verzeichnis `/var/install/dialog.d` shell-Skripte erstellt werden, die vom ECE verwendet werden sollen. Damit die Zuordnung zu den Variablen funktioniert sind die Namen der Skripte so zu wählen, dass sie den zuvor definierten Typbezeichnern entsprechen. Im Beispiel wären dies: `FOO_IP.sh` und `FOO_ACCESS.sh` (wobei die Gross- Kleinschreibung zu beachten ist!).

Die Skripte selbst können komplexe Programme enthalten und müssen dabei der CUI Skript-Spezifikation entsprechen. Die Syntax und das erforderliche Hintergrundwissen vermittelt das Dokument "libcuidoc", das ebenfalls auf der Webseite des eisfair-Projekts zu finden ist. Neben der C-API der libcui wird dort auch die Skript-Schnittstelle beschrieben und eine Referenz der verfügbaren Skript-Befehle geliefert.

An dieser Stelle sollen lediglich Beispiele für Skripte vorgestellt werden, mit denen man Auswahllisten im ECE realisieren kann. Diese Skripte können kopiert und auf vergleichbare Fälle angepasst werden:

6.6.1 ece_select_list_dlg

Der in der ecelib definierte Dialog "ece_select_list_dlg" implementiert eine einfache Auswahlliste, die zentriert über dem Editorfenster angezeigt wird. Der Anwender kann dabei aus einer Reihe möglicher Werte wählen und den gewünschten Wert (z.B. mit Enter) bestätigen. Die Auswahl wird dann in die bearbeitete Konfigurationsvariable des ECE übertragen.

Beispiel für die Verwendung des Dialogs ece_select_list_dialog :

```
#!/bin/sh

. /var/install/include/cuilib
. /var/install/include/ecelib

#-----------------------------------------------------------------
# exec_dailog
# ece --> request to create and execute dialog
#         $p2 --> main window handle
#         $p3 --> name of config variable
#-----------------------------------------------------------------
exec_dialog()
{
    local win="$p2"

    sellist="BLACK,RED,GREEN,BROWN,BLUE,MAGENTA,CYAN,LIGHTGRAY,"
             DARKGRAY,LIGHTRED,LIGHTGREEN,YELLOW,LIGHTBLUE,
             LIGHTMAGENTA,LIGHTCYAN,WHITE"

    ece_select_list_dlg "$win" "Colors" "$sellist"
}

#-----------------------------------------------------------------
# main routine
#-----------------------------------------------------------------

cui_init
cui_run

#-----------------------------------------------------------------
# end
#-----------------------------------------------------------------

exit 0
```

Bei diesem Beispiel wird folgende Auswahlliste ausgegeben:

```
+----------[ Colors ]-----------+
|                               |
| +-------------------------+^| |
| | BLACK                   | | |
| | RED                     | | |
| |<GREEN =================>| | |
| | BROWN                   | | |
```

```
| | BLUE                    | | |
| | MAGENTA                 | | |
| | CYAN                    | | |
| +-------------------------+v+ |
|     [< OK >] [ Cancel ]       |
|                               |
+-------------------------------+
```

6.6.2 ece_comment_list_dlg

Der Dialog "ece_comment_list_dlg" ist eine Erweiterung des "ece_select_list_dlg" Dialogs. Hier ist es möglich neben den Auswahloptionen Kommentare anzugeben, die gemeinsam mit den Auswahlwerten in einer zweispaltigen Auswahlliste dargestellt werden. Auf diese Weise sieht der Anwender nicht nur die Optionen (die für sich gesehen u.U. vergleichsweise nichtssagend sind), sondern auch einen erklärenden Text, der die Auswahl erleichtert.

Beispiel für die Verwendung des Dialogs ece_comment_list_dialog :

```
#!/bin/sh

. /var/install/include/cuilib
. /var/install/include/ecelib

#-------------------------------------------------------------------
# exec_dailog
# ece --> request to create and execute dialog
#         $p2 --> main window handle
#         $p3 --> name of config variable
#-------------------------------------------------------------------
exec_dialog()
{
    local win="$p2"

    sellist="option1|Dies ist Option1,option2|Dies ist Option2,
             option3|Dies ist Option3,option4|Dies ist Option4"

    ece_comment_list_dlg "$win" "Optionen" "$sellist"
}

#-------------------------------------------------------------------
# main routine
#-------------------------------------------------------------------

cui_init
cui_run

#-------------------------------------------------------------------
# end
#-------------------------------------------------------------------

exit 0
```

Bei diesem Beispiel wird folgende Auswahlliste ausgegeben:

```
+-------------[ Optionen ]----------------+
|                                         |
| +-----------------------------------+^| |
| | option1    | Dies ist Option1     | | |
| | option2    | Dies ist Option2     | | |
| |<option3 ===| Dies ist Option3 ===>| | |
| | option4    | Dies ist Option4     | | |
| +-----------------------------------+v+ |
|          [< OK >] [ Cancel ]            |
|                                         |
+-----------------------------------------+
```

6.6.3 ece_select_cblist_dlg

Der in der ecelib definierte Dialog "ece_select_cblist_dlg" implementiert eine Auswahlliste unter Nutzung von Check Boxen, die zentriert über dem Editorfenster angezeigt wird. Der Anwender kann dabei einer Reihe möglicher Werte unter Nutzung der Cursortasten ansteuern und über die Space Taste auswählen. Ausgewählte Werte werden mit [x] markiert. Nicht ausgewählte Werte haben die Markierung [ ]. Die gewünschten Werte können (z.B. mit Enter) bestätigen werden. Die Auswahl wird dann in die bearbeitete Konfigurationsvariable des ECE übertragen.

Beispiel für die Verwendung des Dialogs ece_select_cblist_dialog :

```
#!/bin/sh

. /var/install/include/cuilib
. /var/install/include/ecelib

#-----------------------------------------------------------------
# exec_dailog
# ece --> request to create and execute dialog
#         $p2 --> main window handle
#         $p3 --> name of config variable
#-----------------------------------------------------------------
exec_dialog()
{
  local win="$p2"

  sellist='apache2,ldapserver,mail,mini_httpd,partimg,pure-ftpd,ssmtp'

  ece_select_cblist_dlg "$win" "Select one or more elements from a list" "$sellist"
}

#-----------------------------------------------------------------
# main routine
#-----------------------------------------------------------------

cui_init
cui_run
```

```
#-----------------------------------------------------------------
# end
#-----------------------------------------------------------------

exit 0
```

Bei diesem Beispiel wird folgende Auswahlliste ausgegeben:

```
+-[ Select one or more elements from a list ]-+
|                                             |
|  [x] apache2                                |
|  [ ] ldapserver                             |
|  [ ] mail                                   |
|  [ ] mini_httpd                             |
|  [ ] partimg                                |
|  [x] pure-ftpd                              |
|  [ ] ssmtp                                  |
|                                             |
|           [<  OK  >]  [ Cancel ]            |
|                                             |
+---------------------------------------------+
```

Für maximal 15 Elemente ist Platz auf einem 80/25 Display, daher wird bei mehr als 15 Elemente eine Fehlermeldung erzeugt:

```
+----[ Checkbox selection ]----+
|                              |
|  Too many elements (16/15).  |
|                              |
|          [<  OK  >]          |
+------------------------------+
```

Es können selbstverständlich auch längere Elemente angezeigt und ausgewählt werden, allerdings ist die Ausgabe auf 58 Zeichen begrenzt. Die restlichen Zeichen werden durch ein .. angedeutet.

Beispiel:

```
+-----------[ Select one or more elements from a list ]------------+
|                                                                   |
|  [ ] dies ist ein ziemlich langer Text der auch ziemlich viel U..|
|  [ ] dies nicht                                                   |
|                                                                   |
|                      [<  OK  >]  [ Cancel ]                       |
|                                                                   |
+-------------------------------------------------------------------+
```

ACHTUNG: Es wird nicht geprüft, ob ein Element mehrfach vorkommt.

Mittels der Variablen ECE_SELECT_CBLIST_VAL_SEPARATOR kann der Separator der Select-Liste eingestellt werden.

ECE_SELECT_CBLIST_VAL_SEPARATOR=':'
setzt z.B. den Separator auf den Doppelpunkt.

7 User-Interface

7.1 Menü

7.1.1 Einleitung

Für viele Pakete ist eine Administration unerlässlich. Sofern eine Administration vorgesehen ist, muss diese über das *eisfair* Setup-Menü erfolgen.

Dieses Kapitel der Entwicklerdokumentation beschreibt die Erstellung und Einbindung eigener Untermenüs in das *eisfair* Menüsystem.

Die einzelnen Menüs liegen unter `/var/install/menu/` und sind in einer XML–ähnlichen Syntax geschrieben. Wichtig dabei ist, dass jedes Tag vollständig in einer separaten Zeile steht. Der Dateiname muss nach folgendem Schema aufgebaut sein:

```
setup.services.$package[.$submenu].menu
```

Beispiele:

```
/var/install/menu/setup.services.foo.menu
/var/install/menu/setup.services.foo.bar.menu
```

Beispiel:

```
setup.services.foo.menu

<!-- /var/install/menu/setup.services.foo.menu -->
<!-- Creation:     2005-04-09  max             -->
<!-- Last Update:  2005-04-09  max             -->
<title>Administration of Package Foo</title>
<package>foo</package>
<version/>
<doc>Read Foo Documentation</doc>
<edit>Edit Foo Configuration</edit>
<menu file="setup.services.foo.bar.menu">All bar functions of foo</menu>
<init task="start">Start Foo</init>
<init task="stop">Stop Foo</init>
<script file="foo-dosomething">Do Something</script>
```

ACHTUNG

Da derzeit keine mehrsprachigen Menüs unterstützt werden, **müssen** die Menütexte in Englisch verfasst sein.

7.1.2 Verfügbare Tags

Im folgenden werden die einzelnen Tags mit den jeweiligen Optionen erklärt.

<!—- – >-Tag

Mit dem `<!--` -Tag wird eine Kommentarzeile eingeleitet und mit dem `-->`-Tag beendet.

Beispiel:

```
<!-- Creation: 2005-04-09 jed -->
```

ACHTUNG

Es dürfen nicht mehrere Kommentarzeilen zusammengefasst werden, dies führt im classic Menu zu einem Fehler in der Darstellung

Beispiel: So nicht

```
<!--
/var/install/menu/setup.services.foo.menu
Creation:     2005-04-09  max
Last Update:  2005-04-09  max
-->
```

<title>-Tag

Mit dem `<title>`-Tag wird der Titel des Menüs angegeben.

In jedem Menü und Untermenü muss das `<title>`-Tag gesetzt werden.

Beispiel:

```
<title>Administration of Package Foo</title>
```

<package>-Tag

Mit dem `<package>`-Tag wird die Umgebungsvariable PACKAGE gesetzt. Diese kann dann von abhängigen Skripten ausgewertet werden.

In jedem Menü und Untermenü eines Pakets muss ein `<package>`-Tag gesetzt werden.

Beispiel:

```
<package>foo</package>
```

<version>-Tag

Mit diesem Tag wird gesteuert, dass die aktuelle *eisfair* -Version und eiskernel–Version auf dieser Menüebene angezeigt werden sollen.

Beispiel:

```
<version/>
```

<url>-Tag

Dient zur Anzeige der aktuellen Paket-Installationsquelle.

Beispiel:

```
<url/>
```

<menu>-Tag

Zum Einhängen eines Untermenüs dient der `<menu>`-Tag. Hier wird mit der Option `file="untermenü"` die Menüdatei des Untermenüs angegeben. Diese muss ebenfalls den Spezifikationen des vorliegenden Kapitels genügen.

Der anzuzeigende Name des Untermenüs wird innerhalb des Tags angegeben.

Beispiel:

```
<menu file="setup.services.foo.bar.menu">All bar functions of foo</menu>
```

Die Menüdateien müssen alle unter `/var/install/menu/` liegen und den Namen `setup.services.`*$package*`.menu` bzw. `setup.services.`*$package*`.`*$submenu*`.menu` tragen.

Tipp für fortgeschrittene Entwickler

Beim Tag `<menu>` kann optional die Option `package=""` verwendet werden um die Variable *$PACKAGE* explizit für ein Untermenü zu setzen. Dies entspricht der Verwendung des Tags `<package>` in diesem Untermenü.

Dieses Vorgehen ist nur für generische Untermenüs (z.B. ACFH) gedacht, bei denen eine explizite Verwendung des Tags `<package>` nicht möglich ist.

<doc>-Tag

Die Dokumentation des Paketes oder eine beliebige andere Datei im Unix-Textformat kann mit Hilfe des `<doc>`-Tags angezeigt werden.

Die anzuzeigende Datei wird mit der Option `file=""` angegeben. Sofern die Datei im Verzeichnis `/usr/share/doc/`*$package* liegt, ist die Angabe des Dateinamens ausreichend.

Wird kein Dateiname angegeben, so wird die Datei `/usr/share/doc/`*$package*`/`*$package*`.txt` angezeigt.

Bei abweichenden Verzeichnissen muss der absolute Pfad mit angegeben werden.

Der Aufbau der Paketdokumentation ist im entprechenden Kapitel (Seite 76) der verliegenden Entwicklerdokumentation beschrieben.

Zur Anzeige auf der Konsole wird automatisch der in der Basiskonfiguration eingestellte PAGER verwendet.

Beispiele:

```
<doc file="foo.txt">Read Foo Documentation</doc>
<doc file="/usr/share/doc/foo/bar.txt">Display bar-txt</doc>
```

<edit>-Tag

Mit dem `<edit>`-Tag wird eine Funktion zum Editieren der *eisfair* Konfiguration eingehängt. Es wird dabei immer die mit dem `<package>`-Tag gesetzte Konfiguration bearbeitet.

Bei der Option `package=""` kann optional der Name der Konfigurationsdatei unter `/var/config.d/` angegeben werden. Dieser entspricht dem Namen des Paketes.

Der im Menü anzuzeigende Text ist innerhalb des Tags anzugeben.

Auf der Konsole wird automatisch der in der Basiskonfiguration eingestellte Konfigurationseditor verwendet. Alternative Administrationsoberflächen bieten teilweise eigene Editoren.

Voraussetzung für die Verwendung des `<edit>`-Tags sind:

- Die Konfiguration steht in `/etc/config.d/`*$package* und entspricht den Richtlinien für eine *eisfair* Konfiguration.
- Es existiert eine Standardkonfiguration `/etc/default.d/`*$package*.
- Es wird die Gültigkeitsprüfung mittels `/etc/check.d/`*$package* durchgeführt.
- Es existiert ein spezifikationskonformes Apply-Skript `/var/install/config.d/`*$package*`.sh` zum Schreiben der „nativen“ Konfigurationsdateien des Pakets.
- `/etc/init.d/`*$package* kennt den Parameter „`restart`“ zum Neustarten des Pakets.

Beispiele:

```
<package>foo</package>
<edit>Edit Foo Configuration</edit>
<edit package="myfoo">Edit MyFoo Configuration</edit>
<edit stopstart>Stop process, edit configuration, start process</edit>
```

Die Verarbeitung des Parameters „restart“ in `/etc/init.d/`*$package* sieht im einfachsten Fall so aus:

Beispiel:

```
restart)
    $0 stop
    $0 start
    ;;
```

Sollte ein Neuschreiben der Konfiguration nur möglich sein, wenn das Paket vorher gestoppt wurde, dann kann dazu die Option `stopstart` genutzt werden. In diesem Fall wird nicht `/etc/init.d/`*$package* `restart` zum Neustarten des Pakets verwendet, sondern es wird als erstes das Paket mittels `/etc/init.d/`*$package* `stop` angehalten, bevor die Konfiguration neu geschrieben und danach das Paket wieder gestartet wird. Diese Vorgehensweise darf nur verwendet werden, wenn der normale Ablauf nicht möglich ist!

<init>-Tag

Funktionen in `/etc/init.d/`*$package* können über die standardisierte Schnittstelle `<init>` aufgerufen werden. Als Option wird die durchzuführende Aktion `task=""` übergeben.

Es wird dazu immer das Script unter `/etc/init.d/` aufgerufen, das dem in `<package>` gesetztem Paketnamen entspricht. Optional kann aber auch mittels der Option `package=""` das anzusteuernde Skript angegeben werden. Bei dieser Funktion können auch mehrere Pakete mit einem einzelnen Aufruf angesteuert werden, dazu sind diese durch ein Leerzeichen (Space) getrennt anzugeben.

Im Parameter `task=""` wird die Funktion sowie optional Aufrufparameter angegeben. Gängige Funktionen sind `start`, `stop` oder `status`. Ein Parameter wird durch ein Leerzeichen getrennt in `task=""` mit angegeben.

Beispiele:

```
<init task="start">Start Foo</init>
<init task="stop">Stop Foo</init>
<init package="foo" task="stop -force">Kill Foo/init>
<init task="status">Show Foo Status</init>
```

Tipp für fortgeschrittene Entwickler

Beim `<init>`-Tag können mit der Option `package=""` auch mehrere Pakete durch Leerzeichen getrennt angegeben werden. Die jeweiligen Aktionen werden dann für alle Pakete ausgeführt:

Beispiel:

```
<init package="foo bar"task="start">Foo und Bar starten</init>
```

<script>-Tag

Der Aufruf paketeigener Skripte erfolgt über das Tag <script>. Dort wird mit der Option `file=""` das auszuführende Skript angegeben.

Wenn dieses Skript unter `/var/install/bin/` liegt, dann reicht die Angabe des Namens (ohne absoluten Pfad). (empfohlen)

Alternativ kann auch ein Skript in einem anderen Verzeichnis aufgerufen werden. Dazu muss der absolute Pfad mit angegeben werden. Diese Methode funktioniert nicht mehr mit zukünftigen Webconf-Versionen (siehe Webconf).

Beispiele:

```
<script file="foo-dosomething">Do Something</script>
<script file="/usr/local/foo/doanything">Do Anything</script>
```

7.1.3 Menüs auf alternativen Benutzeroberflächen

Aktuell existieren mehrere alternative Benutzeroberflächen, die Menüinhalte darstellen können. Dies sind das klassische Menüsystem, die Curses-Oberfläche und das Web-Frontend "webconf". Teilweise unterscheiden sich die Menüsysteme in ihrer Funktion und in ihrem Verhalten, weshalb eine Möglichkeit existiert, die Sichtbarkeit von Menüeinträgen einzuschränken. Zu diesem Zweck kann bei den Tags `<menu>`, `<doc>`, `<script>` und `<init>` das optionale Attribut "ui" angegeben werden.

Gültige Werte für das Attribut sind bislang "classic" und "cui". Wird das Attribut nicht angegeben ist der entsprechende Menüeintrag immer sichtbar. Wird "ui" mit dem Wert "classic" angegeben ist der Eintrag lediglich im klassischen Menü sichtbar, während "cui" das selbe für das Curses-Menü leistet. Es ist auch möglich mehrere Werte durch Komma getrennt anzugeben.

Beispiele:

```
<doc file="foo.txt">This item is always visible</doc>
<script file="foo.cui.sh" ui="cui">Curses mode only</script>
<menu file="setup.foo.menu" ui="classic">Classic mode only</menu>
```

7.1.4 Dynamisches Verhalten in Menüs

Falls das bis hierher beschriebene statische Verhalten von Menüs nicht ausreicht, um den Anforderungen eines Paketes zu genügen, können Skripte zur Vor- und Nachbearbeitung von Menüaktionen eingebunden werden. Damit kann z.B. dynamisch ein Untermenü oder eine Dokumentationsdatei erstellt werden.

Zu diesem Zweck kennen die Menü-Tags `<menu>`, `<doc>`, `<edit>`, `<init>` und `<script>` die optionalen Attribute "pre" und "post". Als Wert für die beiden Attribute wird dabei der Verweis auf ein Programm oder ein Shell-Skript erwartet, das entweder mit einem absoluten Pfad oder nur mit seinem Dateinamen angegeben wird. Im zweiten Fall wird angenommen, dass die Datei vom Paketentwickler im Verzeichnis `/var/install/bin` abgelegt wurde.

Beispiele:

```
<menu file="setup.foo.dyn.menu" pre="foo_menu.sh">Dynamic Menu</menu>
<doc pre="foo_compose_mod_doc.sh">View Foo Modules Doc</doc>
```

Im Beispiel wird das Skript */var/install/bin/foo_menu.sh* aufgerufen, bevor das Untermenü geöffnet wird. Ein entsprechendes post-Skript würde entsprechend beim Verlassen des Untermenüs ausgeführt.

Hinweis

In fast allen Anwendungsfällen reicht das statische Verhalten des Menüs mit dessen Möglichkeiten vollkommen aus. Das dynamische Verhalten richtet sich in erster Linie an erfahrene Paketentwickler, welche eine Funktionalität benötigen, die in eisfair noch nicht vorgesehen ist.

An die pre/post-Skripte bzw. -Programme werden eine Reihe von Parametern übergeben, mit denen sich deren Verhalten steuern lässt. Die zu den jeweiligen Tags passenden Parameter sind in der folgenden Tabelle dargestellt:

Tag	$1	$2	$3	$4	$5
`<menu>`	"pre" "post"	*$package*	"menu"	*$menu_file*	
`<doc>`	"pre" "post"	*$package*	"doc"	*$doc_file*	
`<edit>`	"pre" "post"	*$package*	"edit"	*$config_file*	`$stopstart`
`<init>`	"pre" "post"	*$package*	"init"	*$init_file*	*$init_task*
`<script>`	"pre" "post"	*$package*	"script"	*$shell_script*	

Der Parameter *$stopstart* des *<edit>* Tags nimmt die Werte "apply" oder "apply-stopstart" an, je nachdem, ob der zu konfigurierende Dienst neu gestartet werden muss oder nicht.

Dateiverweise wie *$shell_script* werden immer mit einem absoluten Pfad an die pre/post-Skripte übergeben. Steht im Menü die Datei ohne Pfadangabe, dann wird der Pfad nach dem in der folgenden Tabelle dargestellten Schema erweitert.

Datei	Name und absoluter Pfad
menu file	`/var/install/menu/`*$menu_file*
doc file	`/usr/share/doc/`*$package/$doc_file*
config file	`/etc/config.d/`*$config_file*
init file	`/etc/init.d/`*$init_file*
shell script	`/etc/init.d/`*$shell_script*

Sollte ein pre/post-Skript eine Fehlerbedingung feststellen, so kann es durch einen Rückgabewert ungleich "0" (z.B. `exit 1`) diesen Fehler an das Menü signalisieren. Im Falle des pre-Skripts wird dadurch zudem die Ausführung der mit dem Menü-Tag verbundenen Aktion unterbunden. Im Falle von Fehlern sollte in jedem Fall über die Standardausgabe oder die Fehlerausgabe eine erklärende Fehlermeldung erfolgen.

Beispiel:

```
echo "Unable to write menu file "\$3"!"
exit 1
```

Hinweis

Die mit einem Tag verbundene Menüaktion wird auch dann nicht ausgeführt, wenn das in der "pre" Option angegebene Programm oder Skript nicht gefunden oder nicht ausgeführt werden konnte. Eine entsprechende Fehlermeldung wird in diesem Fall vom System erzeugt.

7.1.5 Untermenüs einhängen und entfernen

Um eigene Untermenüs ins *eisfair* Menü einhängen zu können, gibt es die Skripte `/var/install/bin/add-menu` und `/var/install/bin/del-menu`. Beide Skripte werden mit den selben Parametern aufgerufen, deshalb werden diese nur einmal gemeinsam erklärt.

```
/var/install/bin/add-menu MENUE UNTERMENUE MENUETEXT
/var/install/bin/del-menu MENUE UNTERMENUE
```

MENUE	gibt das Menü an, unter welches das neü Menü gehängt werden soll.
UNTERMENUE	gibt das Untermenü an, dass in MENUE eingehängt werden soll.
MENUETEXT	gibt den in MENUE anzuzeigenden Text an.

Beispiele:

```
/var/install/bin/add-menu         \
    setup.services.menu           \
    setup.services.foo.menu       \
    "Foo Configuration"

/var/install/bin/del-menu         \
```

```
setup.services.menu            \
setup.services.foo.menu
```

ACHTUNG

Hier hat sich die Aufrufsyntax gegenüber der alten Version geändert. Der alte Aufruf mit dem auszuführenden Skript als Parameter funktioniert aus Kompatibilitätsgründen weiter, darf aber bei neuen Releases nicht mehr verwendet werden!

Hinweis

Bis inklusive Base Version 1.0.7 wurde eine andere Syntax für die Menüstruktur verwendet. Dabei konnten aus dem Menü nur Shellscripts aufgerufen werden, die ggf. wieder ein Menü anzeigten.
Diese Menüsyntax ist veraltet und wird nur übergangsweise parallel unterstützt. Alternative Administrationsoberflächen werden mit der alten Menüstruktur nicht funktionieren, weshalb die Verwendung der alten Struktur in neün Releases nicht mehr zulässig ist.
Bei Verwendung der neuen Menüsyntax sind die alten Scripte wie foo-edit, foo-doc oder foo-bar nicht mehr nötig, da viele Funktionen jetzt standardisiert aufgerufen werden.

Sollte es aus einem Grund einmal nötig sein eine komplett neue Menüdatei anzulegen, so kann dies durch Aufruf des Skriptes `/var/install/bin/create-menu` erledigt werden.

```
/var/install/bin/create-menu MENUE MENUETITEL
```

MENUE	gibt das Menü an, welches erzeugt werden soll.
MENUETITEL	gibt den Menütitel an, welcher über dem Menü angezeigt werden soll.

7.2 Dokumentation

7.2.1 Dokumentation

Die eigentliche Dokumentation eines Pakets muss im Verzeichnis `/usr/share/doc/`*$package* abgelegt werden und eine im Unix-Textformat erstellte Datei Namens *$package*`.txt` sein.

In der Dokumentation müssen zumindest folgende Punkte genauer erklärt werden:

- Welche weiteren Pakete werden benötigt?
- Eventuelle Konfigurationsdateien und die in ihr definierten Variablen.
- Die Punkte im Setup-Menü, soweit sie dieses Paket betreffen.

- Die Funktionen des Pakets.
- Wie arbeitet dieses Paket mit anderen Paketen zusammen?

Zusätzliche Dokumentation, die den Originalquellen normalerweise beiliegt (man-Pages, info-Dateien, README, etc.), dürfen **nicht** mitgeliefert werden.

7.2.2 Changelog

Im Verzeichnis `/usr/share/doc/`*$package* ist eine Datei `changes.txt` anzulegen, in dem die Veränderungen, die von Version zu Version am Paket durchgeführt werden, kurz beschrieben werden.

Bewährt hat sich eine Form wie in folgendem Beispiel:

Beispiel für ein Changelog:

```
1.0.6
-----
- added file /bin/bigone
- deleted obsolete variable ONE in
  /etc/config.d/master
- renamed /etc/old to /etc/new
-
-

1.0.5
-----
- modified some comments in /etc/this
-
-
```

Dabei sollten die letzten änderungen jeweils oben in der Datei changes.txt stehen. Uralte änderungen können mit der Zeit auch entfallen.

8 Sonstiges

8.1 Loggen von Meldungen

Um Status- oder Fehlermeldungen mitzuloggen gibt es unter Linux den Dienst „syslogd“. Über diesen Daemon werden die Meldungen an ein oder mehrere konfigurierbare Ziele weitergeleitet. Diese Einstellungen werden über die „Base configuration“ vorgenommen.

Es empfiehlt sich daher, kurze Status- und Fehlermeldungen aus dem Paket über den syslogd mitzuloggen. Dies kann einfach über den Aufruf

```
/usr/bin/logger -t $PACKAGE             \
                -p $FACILITY.$LEVEL     \
                "Text der Meldung"

# FACILITY: auth, authpriv (for security information of a sensitive nature)
#           cron, daemon, ftp, kern, lpr, mail, news, syslog, user, uucp,
#           local0 .. local7
# LEVEL:    alert, crit, debug, emerg, err, info, notice, warning
```

erfolgen. Details zum Aufruf sowie zu den möglichen Werten kann der Dokumentation zum logger entnommen werden und steht an verschiedenen Stellen im Internet zur Verfügung (Tip: Benutzung einer Suchmaschine).

Die Logausgaben werden dabei üblicherweise in die Datei `/var/log/messages` geschrieben. Damit diese Datei nicht endlos wächst, gibt es den logrotate, der in regelmässigen Abständen die aktuelle logdatei mit gzip komprimiert und eine neue Logdatei beginnt. Nach einer gewissen Zeitspanne werden dann die ältesten Archive gelöscht.

Da sehr viele verschiedene Dienste den Service des syslogd nutzen, ist dieser Dienst nicht besonders gut für umfangreichere Logausgaben geeignet. Daher sollen diese in eine eigene Logdatei `/var/log/log.`*$PACKAGE* erfolgen.

Um zu verhindern dass auch diese Dateien unbegrenzt wachsen, kann auch jedes Paket den logrotate–Mechanismus für seine eigenen Logausgaben nutzen. Der logrotate wird dabei durch die Anlage einer Steuerdatei im Verzeichnis `/etc/logrotate.d/` gesteuert. Der Name dieser Steuerdatei muss wiederum dem Paketnamen entsprechen.

Beispiel für eine Logrotate–Konfigurationsdatei (Paket foo):

```
#-----------------------------------------------------------------------------
# /etc/logrotate.d/foo - foo logrotate configuration
#
```

```
# Creation:     2005-02-01  max
# Last Update:  2005-03-06  max
#
# Copyright (c) 2001-2015 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#-----------------------------------------------------------------------------

/var/log/log.foo {
    rotate 7
    daily
    compress
    missingok
    notifempty
    create 644 root root
    }
```

Details zu dieser Steuerdatei können der Dokumentation des logrotate entnommen werden.

8.2 Cronjobs

Um automatisch zeitgesteuert ein Script ausführen zu können, beispielsweise um einen Statusreport zu erstellen, gibt es unter Unix „Cronjobs“. Diese können von *eisfair* -Paketen mittels eines einfachen Mechanismus angelegt werden. Es muss lediglich eine Steuerdatei in `/var/cron/etc/root/` mit dem Namen des Pakets erstellt und die Aktualisierung des Cron-Daemon angestossen werden.

Diese Aktualisierung geschieht mit dem Aufruf:

```
# update crontab file
/var/install/config.d/cron.sh
```

Beispiel für eine Cron–Konfigurationsdatei (Paket foo):

```
#
# Do not edit this file, edit /etc/config.d/foo
# Creation date: Fri Mar 11 19:54:43 CET 2005
53 3 * * * /var/install/bin/foo-cron
12 */2 * * * /var/install/bin/foo-cron2
```

In diesem Beispiel wird das Script `/var/install/bin/foo-cron` immer Nachts um 3:53 sowie das Script `/var/install/bin/foo-cron2` jede 2. Stunde immer 12 Minuten nach der vollen Stunde gestartet.

Zur Konfiguration der Ausführungszeiten gilt folgende Reihenfolge: Minute Stunde Tag Monat Wochentag

Details dazu können der Dokumentation zu crontab entnommen werden.

8.3 Systemdateien aktualisieren

Manchmal ist es notwendig Einträge zu Systemdateien hinzuzufügen bzw. diese zu ändern. Damit die Dateien nicht direkt bearbeitet werden müssen, wurde ab dem base-Paket v1.0.8 eine einheitliche Schnittstellenfunktion realisiert. Initial wurden Funktionen für das Aktualisieren von `/etc/host.allow`, `/etc/host.deny`, `/etc/services` und `/etc/inittab` implementiert.

8.3.1 /etc/host.allow

Will ein Programm Einträge zu der Datei /etc/hosts.allow hinzufügen, so muss es eine Datei /etc/hosts.allow.*$package* schreiben, die wie folgt aussehen sollte:

```
#
# /etc/hosts.allow.nfsserver list file
# generated by /var/install/config.d/nfsserver.sh v1.0.1-3
#
portmap: 192.168.6.0/24
lockd:   192.168.6.0/24
rquotad: 192.168.6.0/24
mountd:  192.168.6.0/24
statd:   192.168.6.0/24
```

Um die Änderungen bzw. Ergänzungen abschliessend zu übernehmen, muss noch das Programm `/var/install/bin/update-hosts.allow` *$package* aufgerufen werden. Bei der Deinstallation des Paketes muss nur die Datei `/etc/hosts.allow.`*$package* gelöscht und erneut das update-hosts.allow-Script aufgerufen werden.

8.3.2 /etc/host.deny

Will ein Programm Einträge zur Datei `/etc/hosts.deny` hinzufügen, so muss es eine Datei `/etc/hosts.deny.`*$package* schreiben, die wie folgt aussehen sollte:

```
#
# /etc/hosts.deny.nfsserver list file
# generated by /var/install/config.d/nfsserver.sh v1.0.4-1
#
portmap: ALL
lockd:   ALL
```

```
mountd:  ALL
rquotad: ALL
statd:   ALL
```

Um die Änderungen bzw. Ergänzungen abschliessend zu übernehmen, muss noch das Programm `/var/install/bin/update-hosts.deny` *$package* aufgerufen werden. Bei der Deinstallation des Paketes muss nur die Datei `/etc/hosts.deny.`*$package* gelöscht und erneut das update-hosts.deny-Script aufgerufen werden.

8.3.3 /etc/services

Will ein Programm Einträge zur Datei `/etc/services` hinzufügen, so muss es eine Datei `/etc/services.`*$package* schreiben, die wie folgt aussehen sollte:

```
#
# /etc/services.antispam list file
# generated by /var/install/config.d/antispam.sh v1.2.2
#
spamd           783/tcp               # SpamAssassin daemon
```

Um die Änderungen bzw. Ergänzungen abschliessend zu übernehmen, muss noch das Programm `/var/install/bin/update-services` *$package* aufgerufen werden. Als Basis wird immer die Datei `/etc/services.std` verwendet deren Einträge dann ersetzt bzw. zu der neue Einträge hinzugefügt werden. Bei der Deinstallation des Paketes muss nur die Datei `/etc/services.`*$package* gelöscht und erneut das update-services-Script aufgerufen werden.

8.3.4 /etc/inittab

Will ein Programm Einträge zur Datei `/etc/inittab` hinzufügen, so muss es eine Datei `/etc/inittab.`*$package* schreiben, die wie folgt aussehen sollte:

```
#
# /etc/inittab.vbox list file
# generated by /var/install/config.d/vbox.sh v1.0.4-1
#
I6:2345:respawn:/usr/local/vbox/sbin/vboxgetty -d /dev/ttyI6
I7:2345:respawn:/usr/local/vbox/sbin/vboxgetty -d /dev/ttyI7
```

Um die Änderungen bzw. Ergänzungen abschliessend zu übernehmen, muss noch das Programm `/var/install/bin/update-inittab` *$package* aufgerufen werden. Als Basis wird immer die Datei `/etc/inittab.std` verwendet deren Einträge dann ersetzt bzw. zu der neue Einträge hinzugefügt werden. Bei der Deinstallation des Paketes muss nur die Datei `/etc/inittab.`*$package* gelöscht und erneut das update-inittab-Script aufgerufen werden.

8.3.5 /etc/sudoers

Ab der base Version 1.7.2 gibt es eine Änderung für /etc/sudoers.

Sudo kennt nun

```
# includedir /etc/sudoers.d
```

Will ein Programm Eintraege zur Datei `/etc/sudoers` hinzufügen, so muss es eine Datei `/etc/sudoers.d/`*$package* schreiben, die wie folgt aussehen sollte:

```
#
# /etc/sudoers.d/eisfax list file generated by eisfax Version 1.2.0
#
Cmnd_Alias EISFAX2PDF = /usr/bin/tiff2pdf
Cmnd_Alias EISFAXCHOWN = /bin/chown
Cmnd_Alias EISFAXCHMOD = /bin/chmod

fax ALL = NOPASSWD: EISFAX2PDF, EISFAXCHOWN, EISFAXCHMOD
```

Für diese Datei ist ein Filemod von 440 zwingend erforderlich.

```
chmod 0440 /etc/sudoers.d/$package
```

8.3.6 Debug-Modus

Wird die Option -debug an die genannten Scripte übergeben, so werden detailliertere Informationen über eventuell gefundene Fehler ausgegeben.

8.3.7 Beispiel

Die Implementierung wird am Beispiel der Konfiguration des nfsserver-Paketes gezeigt:

```
...

pgmname=`basename $0`

# set file names
configfile=$testroot/etc/config.d/nfsserver
generate_hostsallow=/etc/hosts.allow.nfsserver

# other parameters
nfsserver_version="v1.0.1-3"

...

echo "#"
echo "# $generate_hostsallow list file"
```

```
echo "# generated by $pgmname $nfsserver_version"
echo "#"

...

# set access rights
chown root $generate_hostsallow
chgrp root $generate_hostsallow
chmod 0640 $generate_hostsallow

# update file
/var/install/bin/update-hosts.allow $pgmname

...
```

8.3.8 Gruppen- und User-IDs

Um weiteres Chaos zu vermeiden, wurde jetzt eine Liste von UIDs und GIDs festgelegt, an die sich alle Packages halten sollten. Die Liste wurde aus den Einträgen von Debian Woody und SuSE gemischt, wobei im Zweifelsfall Debian das letzte Wort hatte.

Username	uid	gid
root	0	0
daemon	1	1
bin	2	2
sys	3	3
games	5	5
lp	7	7
mail	8	8
news	9	9
uucp	10	10
proxy	13	13
messagebus	18	18
fax	21	10
at	25	-
postgres	26	26
majordom	30	31
postgres	31	32
wwwrun	30	33
backup	34	34
operator	37	37
list	38	38
irc	39	39
ftp	40	49
gnats	41	41

Username	uid	gid
exim	42	42
named	44	44
mumble-server	46	46
postfix	51	51
mysql	60	60
man	62	62
zope	64	2
sshd	71	65
firebird	84	84
clamav	85	42
quassel-core	179	179
nobody	65534	65534
identd	100	65534

Gruppenname	gid
root	0
daemon	1
bin	2
sys	3
adm	4
tty	5
cdisk	6
lp	7
mail	8
news	9
uucp	10
proxy	13
kmem	15
messagebus	18
dialout	20
voice	22
cdrom	24
floppy	25
tape	26
sudo	27
audio	29
dip	30
majordom	31
postgres	32
video	33
backup	34
operator	37
list	38
irc	39

Gruppenname	gid
src	40
gnats	41
trusted	42
utmp	43
named	44
mumble-server	46
ftp	49
staff	50
postfix	51
maildrop	59
mysql	60
man	62
sshd	65
firebird	84
users	100
quassel	179
maschines	777
nobody	65533
nogroup	65534

Wer meint, dass in dieser Liste ein Eintrag fehlt, der solle sich in der Newsgroup spline.eisfair.dev melden.

9 eisfair spezifische Skripte

9.1 eislib – Funktionen zur Steuerung des Userinterfaces

In *eisfair* gibt es die include-Bibliothek „eislib“. Darin werden verschiedene standardisierte Funktionen zusammengefasst. Die Funktionalität wurde gegenüber den bisher verwendeten Skripten deutlich erweitert.

Aus Kompatibilitätsgründen werden die alten Skripte für eine übergangszeit weiterhin unterstützt. Die Verwendung von hier nicht dokumentierten Skripten in neuen Releases ist nicht mehr zulässig.

Die eislib wird am Anfang des paketeigenen Skriptes durch folgenden Aufruf eingebunden:

```
# include eislib
. /var/install/include/eislib
```

Beim Aufruf der Funktionen braucht nichts weiter beachtet zu werden.

Beispiel:

```
anykey
```

> **Hinweis**
>
> Es muss darauf geachtet werden, dass der Inhalt der Variablen `$IFS` möglichst umgehend nach deren Benutzung immer auf den Ursprungswert zurück gesetzt wird, um eventuellen Störungen vorzubeugen.

Beispiel:

```
while read line
do
    _ifs="$IFS"                         # Inhalt sichern
    IFS=':'                             # Inhalt neu zuweisen

    set - $line                         # Bearbeitung durchfuehren

    IFS="$_ifs"                         # Inhalt wieder herstellen

    if [ $3 -gt $gid -a $3 -lt 300 ]    # ...
    then
        gid=$3
    fi
```

```
done </etc/group
```

In *eisfair* Version 1.5.0 sind die im Folgenden aufgeführten Funktionen verfügbar:

9.1.1 anykey

anykey wartet an der Konsole auf die Eingabe „ENTER“. Beim Aufruf unter einem Webbrowser wird der Aufruf von anykey ignoriert, d.h. es erfolgt keine Unterbrechung des Skript-Ablaufs.

An der Konsole darf anykey nur zum Anhalten einer Bildschirmausgabe (bessere Lesbarkeit) benutzt werden. Eine Verwendung zur Steuerung eines Ablaufs (Bitte Diskette Einlegen und Enter drücken) ist nicht mit webconf kompatibel!

9.1.2 clrhome

clrhome löscht den Bildschirm, so dass eine übersichtlichere Anzeige möglich ist (z.B. mehrseitige Anzeigen). Unter einem Webbrowser wird stattdessen eine horizontale Trennlinie angezeigt.

9.1.3 progress_bar

Mittels `progress_bar` kann man auf der Konsole einen Fortschrittsbalken anzeigen. So kann der Anwender bei längeren Verarbeitungen über den aktuellen Status auf dem Laufenden gehalten werden. Im Webbrowser erfolgt keine Anzeige.

Syntax

```
progress_bar [--tty|--html|--file] current_step total_steps
```

```
options:
 --tty   use console colors
 --html  use html tags for colors
 --file  don't use any color tags
```

Als `current_step` wird der aktuelle Stand der Verarbeitung angegeben, `total_steps` ist die Gesamtzahl der durchzuführenden Verarbeitungsschritte. Um die Anzeige zu aktualisieren wird `progress_bar` einfach erneut aufgerufen, ohne dass zwischenzeitlich eine eigene Ausgabe erfolgt ist. Es wird dann die alte Anzeige mit dem neuen Fortschrittsbalken überschrieben.

Anzeigebeispiel (progress_bar 17 26):

```
 65% [================================>                  ] 17 / 26
```

9.1.4 mecho

mecho dient zur Ausgabe von Meldungen. Dabei kann als Parameter angegeben werden, welche Formatierungen verwendet werden sollen. Die Ausgabe erfolgt angepasst an die Aufrufvariante (Konsole oder HTML unter der Kontrolle eines Webbrowsers).

Derzeit werden folgende Formatierungen unterstützt:

`--std` **Ausgabe einer allgemeinen Meldung (Standard-Einstellung)**
Dies sollte der Standardfall bei der Ausgabe von Meldungen sein. Ein Beispiel dafür ist die Ausgabe von Statusmeldungen beim Starten eines Dienstes oder beim Verarbeiten einer geänderten Konfiguration.

`--stdbr` **Ausgabe einer allgemeinen Meldung (Breit - Weiss)**
Dies sollte der Standardfall bei der Ausgabe von Meldungen sein. Ein Beispiel dafür ist die Ausgabe von Statusmeldungen beim Starten eines Dienstes oder beim Verarbeiten einer geänderten Konfiguration.

`--info` **Ausgabe einer Information**
Eine wichtige Information. Dieser Typ wird für die Hervorhebung wichtiger Informationsmeldungen verwendet. Dem Anwender wird damit das Ergebnis „es ist alles in Ordnung“ signalisiert.

`--warn` **Ausgabe einer Warnung**
Warnungen werden mit dieser Option gesteuert. Eine Warnung signalisiert Probleme bei der Verarbeitung, wobei diese dennoch erfolgreich durchgeführt wurde.

Mit `--warn` kann der Anwender auch auf eine Gefahr aufmerksam gemacht werden (z.B. eine häufige Fehlerquelle bei der Konfiguration eines Paketes).

`--error` **Ausgabe einer Fehlermeldung**
Mittels der Formatierung `--error` wird eine klassische Fehlermeldung ausgegeben. Dies soll dem Anwender den Abbruch der Verarbeitung bzw. den Misserfolg einer Verarbeitung signalisieren.

`--link` **Ausgabe für einen Link (Farbe cyan)**
Ausgabe eines Zeichen z.B. für einen Link '>'

`--ok` **Ausgabe einer [OK] Meldung**
Mittels der Formatierung `--ok` wird eine OK Meldung ausgegeben. Die rechts am Rand der Arbeitsfläche plaziert ist und die Erfolgreiche Verarbeitung anzeigt.

```
Create configuration ...                                    [  OK  ]
```

`--fail` **Ausgabe einer [FAIL] Meldung**

Mittels der Formatierung `--fail` wird eine FAIL Meldung ausgegeben, die rechts am Rand der Arbeitsfläche plaziert ist und den Abbruch der Verarbeitung bzw. den Misserfolg einer Verarbeitung anzeigt.

```
Create configuration ...                                    [ FAIL ]
```

Syntax

```
mecho [-n] [--std|--stdbr|--info|--warn|--error|--link]
[--tty|--html|--file] message ...
```

```
options:
 -n       do not append newline

 --std    print standard message (default)
 --stdbr  print standard message (default - breit)
 --info   print info message
 --warn   print warning message
 --error  print error message
 --link   print message in - cyan
 --ok     print message [  OK  ]
 --fail   print message [ FAIL ]

 --tty    use console colors
 --html   use html tags for colors
 --file   don't use any color tags
```

Beispiel für OK und FAIL mit der Möglichkeit zur Fehlerbehandlung.

```
mecho --std " Create configuration ... "
_do_foo
retval="${?}"
if [ "${retval}" -eq 0 ]
then
    mecho --ok
else
    mecho --fail
    _do_bar
fi
```

Tipp für fortgeschrittene Entwickler

Mit den Optionen `[--tty|--html|--file]` kann die automatische Erkennung der Ausgabe-Umgebung gezielt ausgeschaltet werden. Man kann so bei einem manuellen Aufruf von der Konsole einfach den HTML-Code erzeugen.

9.1.5 echo_retval

Die Funktion 'echo_retval' liefert, in Abhängikeit vom Rückgabewert, [OK] oder [FAIL].

Beispiel für OK und FAIL ohne Eingriff für die Fehlerbehandlung.

```
mecho --std " Create configuration ... "
_do_foo
retval="${?}"
echo_retval "${retval}"
```

9.1.6 techo

Tabellen können mit Hilfe von techo bequem formatiert werden. Wie die anderen Funktionen der eislib funktioniert die Ausgabe von Tabellen sowohl an der Konsole als auch unter einem Webbrowser.

Dabei erfolgt die Ausgabe der Tabelle in drei Phasen:

Initialisierung Bei der Initialisierung `techo --begin` wird eine Liste der Spaltenbreiten übergeben. Diese sind in Zeichen anzugeben; bei der html-Ausgabe werden diese prozentual berücksichtigt.

Die Summe aller Spaltenbreiten einer Tabelle darf 80 nicht übersteigen, da sonst die Ausgabe auf dem Bildschirm nicht sichergestellt ist. In diesem Fall wird `techo` mit einer Fehlermeldung abgebrochen, die Zahlen müssen in ' ' eingefasst werden, damit ein expandieren vom '*' in der shell verhindert wird. Ab der eisfair base Version 1.5.3 ist die Spaltenbreite bei `techo --file` beliebig.

Tipp für fortgeschrittene Entwickler

Für jede Spalte kann angegeben werden, ob diese rechts- oder linksbündig ausgerichtet sein soll. Dazu wird der Spaltenbreite einfach ein „r“ bzw. ein „l“ angefügt. Ausserdem können Spalten mit variabler Breite definiert werden indem bei der Definition ein „*“ angefügt wird.

Ausgabe der Tabellenzeilen Jede Tabellenzeile wird mittels `techo --row` ausgegeben. Danach folgt eine Liste der einzelnen Spalteninhalte. Die gesamte Zeile oder auch jede Zelle kann in einer der vordefinierten Farben ausgegeben werden.

Für die Formatierung der gesamten Zeile muss die Angabe zwischen den Schlüsselwörtern `techo` und `--row` erfolgen. Für einzelne Zellen muss die Formatierung dieser vorangestellt werden.

Die Parameter zur Steuerung der Formatierung sind unter der Funktion mecho beschrieben.

Abschluss der Tabelle Der Abschluss der Ausgabe einer Tabelle erfolgt mit `techo --end`.

Syntax

```
techo [--tty|--html|--file] --begin width[align] width[align] ...
techo [--tty|--html|--file] [--std|--stdbr|--info|--warn|--error|--link]
--row [--std|--stdbr|--info|--warn|--error|--link] message ...
techo [--tty|--html|--file] --end

options:

 --tty   use console colors
 --html  use html tags for colors
 --file  don't use any color tags

message-options:

 --std    print standard message (default)
 --stdbr  print standard message (default - breit)
 --info   print info message
 --warn   print warning message
 --error  print error message
 --link   print message in - cyan
```

Beispiel:

```
techo --begin '3 3r 15 10 28* 20'
techo --row "" 1 foo foo "This is a somehow longer text, since I
need a very long line of text" foo
techo --row "->" 2 --info "bar foo" --warn "foo bar" foo
techo --row "" 12 foofoo foo foo --error foobar
techo --info --row "" 24 foo foobar foo --error foobar
techo --end
```

Tipp für fortgeschrittene Entwickler

Die Funktionen der eislib funktionieren nicht nur auf der Konsole und unter einem Webbrowser, sondern auch bei der Umleitung der Ausgabeströme in eine Datei oder Variable.

Beispiel:

```
(
techo --begin '4 40'
techo --row ""   "Zeile 1"
techo --row ""   "Zeile 2"
techo --row "->" "Zeile 3"
techo --row ""   "Zeile 4"
techo --end
) >/tmp/foo.txt
```

9.1.7 eistime

Ueber das Skript `eistime` werden Datum und Zeit in der ISO-8601-Notation systemweit zur Verfügung gestellt.

Syntax

${EISDATE}	=	2006-04-25
${EISTIME}	=	21:16:38

Beispiel:

```
cat  >${HOSTS_HFAXD_FILE} <<EOF
# ---------------------------------------------------------------------
#  hosts.hfaxd file generated by ${PACKAGE_NAME} version: ${VERSION}
#
#  Do not edit this file, edit ${CONFIG_FILE}
#  Creation Date: ${EISDATE} Time: ${EISTIME}
# ---------------------------------------------------------------------
EOF
```

Ergibt:

```
# ---------------------------------------------------------------------
#  hosts.hfaxd file generated by eisfax version: 1.1.20
#
#  Do not edit this file, edit /etc/config.d/eisfax
#  Creation Date: 2006-04-25 Time: 21:16:38
# ---------------------------------------------------------------------
```

9.1.8 refresh_screensize

Die Funktion 'refresh_screensize' wird mit dem laden der eislib ausgeführt und kann jeder Zeit wieder aufgerufen werden.

Der Rückgabewert ist: _EISLIB_SCREENSIZE_X actual number of screen columns _EISLIB_SCREENSIZE_Y actual number of screen lines

9.1.9 check_screensize

Die Funktion 'check_screensize' uberprüft die grösse des Terminal. Der Rückgabewert von 1 bedeutet ein zu kleines Terminal.

Default: _EISLIB_SCREENSIZE_X_MIN = 80 _EISLIB_SCREENSIZE_Y_MIN = 24 _EISLIB_SCREENSIZE_Y_FILE = 999999

9.2 inetlib – Funktionen für Parameter von Interfaces

`get_interfaces()`	get list of interfaces
`get_interface()`	get name of interface
`get_ipaddr()`	get IP address of interface
`get_netmask()`	get Netmask of interface
`get_broadcast()`	get Boadcast address of interface
`get_network()`	get Network of interface

Diese Funktionen dienen dazu, die aktuellen Parameter eines Interfaces (z.B. eth0, eth1 oder auch ppp0) zu ermitteln.

Eigentlich stehen diese Angaben über die entsprechenden Variablen

`IP_ETH_%_NAME`

`IP_ETH_%_IPADDR`

`IP_ETH_%_NETWORK`

`IP_ETH_%_NETMASK`

aus der Konfigurationsdatei `/etc/config.d/base` zur Verfügung.

Werden die Parameter für Interfaces aber dynamisch vergeben (z.B. bei dhcp oder beim DSL Paket für ppp0), so stimmen die Angaben aus `/etc/config.d/base` nicht mehr mit den aktuellen Parametern der Interfaces überein.

Die aufgeführten Funktionen ermitteln die Parameter, indem sie die Ausgabe des Befehls `ifconfig` auswerten.

9.2.1 get_interfaces

Die Funktion `get_interfaces` gibt eine Liste der Interfaces zurück. Dabei werden nur solche Interfaces ermittelt, die mit eth oder ppp beginnen. Das Loopback-Interface lo ist z.B. nicht in der Liste beinhaltet.

Beispiel:

```
IF=`get_interfaces`
mecho --info "Vorhandene Interfaces $IF"
```

Ausgabe ist beispielsweise:

```
Vorhandene Interfaces eth0 eth0:0
```

Hier wird die Liste der vorhandenen Interfaces ausgegeben.

9.2.2 weitere get-Funktionen

Die weiteren Funktionen geben die entsprechenden Parameter eines Interfaces aus. Als Parameter kann entweder die Nummer der `IP_ETH_%`-Variablen aus `/etc/config.d/base`

angegeben werden oder der Name eines Interfaces.

Für `get_interface` macht natürlich nur die Nummer einen Sinn.

Aufrufe (hier für get_ipaddr):

```
get_ipaddr 1
```

Hier wird die aktuelle IP-Adresse des Interfaces ausgegeben, welche über `IP_ETH_1_NAME` identifiziert wird. Ist `IP_ETH_1_NAME` leer, so wird das Interface eth0 genutzt. Achtung: eth0 und nicht eth1.

```
get_ipaddr eth0:0
```

Hier wird die aktuelle IP-Adresse des Interfaces eth0:0 ausgegeben.

Existiert ein angegebenes Interface nicht, so wird kein Text bzw. leerer Text ausgeben. Dies gilt auch bei ungültigem Parameter, wie z.B. ett4.

Ein umfangreicheres Beispiel:

```
# include eislib
. /var/install/include/eislib

# include inetlib
. /var/install/include/inetlib

# Test the library

mecho --info "Test inetlib functions"
mecho "The interfaces are retrieved using get_interfaces."
mecho

for VAL in $(get_interfaces)
do
    mecho --info "Working for interface: $VAL"
    mecho "IP address: $(get_ipaddr $VAL)"
    mecho "Netmask   : $(get_netmask $VAL)"
    mecho "Broadcast : $(get_broadcast $VAL)"
    mecho "Network   : $(get_network $VAL)"
    mecho
done

mecho --info "Check function get_interface"
mecho "Use parameter 1, 2 and bad"
mecho

for VAL in 1 2 bad
do
    mecho --info "Working for interface: $VAL"
```

```
    mecho "Interface $VAL : $(get_interface $VAL)"
done
```

Hier wird die Liste der Interfaces ermittelt und zu jedem Interface werden alle Parameter ausgegeben.

Ausgabe kann beispielsweise sein:

```
Test inetlib functions
The interfaces are retrieved using get_interfaces.

Working for interface: eth0
IP address: 192.168.1.252
Netmask   : 255.255.255.0
Broadcast : 192.168.1.255
Network   : 192.168.1.0

Working for interface: eth0:0
IP address: 192.168.1.251
Netmask   : 255.255.255.0
Broadcast : 192.168.1.255
Network   : 192.168.1.0

Check function get_interface
Use parameter 1, 2 and bad

Working for interface: 1
Interface 1 : eth0
Working for interface: 2
Interface 2 : eth0:0
Working for interface: bad
Interface bad :
```

9.3 configlib – Funktionen zum Handling der Konfigurationsdateien

Diese Bibliothek beinhaltet Funktionen zur Behandlung der *eisfair* Konfigurationsdateien.

Das Einbinden dieser Bibliothek erfolgt analog zur eislib:

```
# include configlib
. /var/install/include/configlib
```

9.3.1 printgpl

Die Funktion `printgpl()` gibt einen GPL-Header für eine *eisfair* -Konfigurationsdatei aus.

Beispiel:

```
printgpl --conf                 \
         "foo"                  \
         "2005-03-25" "frank"   \
         "2001-2013 the eisfair team, team(at)eisfair(dot)org"
```

Im Beispiel wird der GPL-Header für das angegebene Paket ausgegeben.

```
 printgpl [Options] PACKAGE CREATED CREATOR COPYRIGHT

 Options:
   [--conf]      - print the GPL Header for config
   [--check]     - print the GPL Header for check
   [--check_exp] - print the GPL Header for check.exp
   [--check_ext] - print the GPL Header for check.ext

 PACKAGE         - Name of the package
 CREATED         - Package creation Date
 CREATOR         - Original package author
 COPYRIGHT       - Copyright String (optional) wenn nicht gesetzt default:
     "Copyright (c) 2001-2011 the eisfair team, team(at)eisfair(dot)org"
```

Das sieht dann in unserem Beispiel so aus:

Beispiel:

```
# ----------------------------------------------------------------------
# /etc/config.d/foo - configuration file for foo
#
# Creation:     2005-03-25  frank
# Last Update:  2006-02-10  max
#
# Copyright (c) 2001-2013 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
# ----------------------------------------------------------------------
```

9.3.2 check

Bei einem erweiterten „Erweitertes update (Seite 31)“ `$package-update.sh` Skript können auch die folgenden Header benutzt werden.

—check

Der Parameter `--check` gibt einen GPL-Header für eine *eisfair* -check-datei aus.

Beispiel:

```
printgpl --check                \
         "foo"                  \
         "2005-03-25" "frank"   \
         "2001-2013 the eisfair team, team(at)eisfair(dot)org"
```

Beispiel:

```
# ----------------------------------------------------------------------
# /etc/check.d/foo - eischk file for foo
#
# Creation:
```

Im Beispiel wird der GPL-Header für das angegebene Paket ausgegeben.

Das Ergebnis entspricht dem vorangegangenen Beispiel `printgpl`.

—check_exp

Der Parameter `--check_exp` gibt einen GPL-Header für eine *eisfair* -check.exp-datei aus.

Beispiel:

```
printgpl --check_exp            \
         "foo"                  \
         "2005-03-25" "frank"   \
         "2001-2013 the eisfair team, team(at)eisfair(dot)org"
```

Beispiel:

```
# ----------------------------------------------------------------------
# /etc/check.d/foo.exp - eischk exp file for foo
#
# Creation:
```

Im Beispiel wird der GPL-Header für das angegebene Paket ausgegeben.

Das Ergebnis entspricht dem vorangegangenen Beispiel `printgpl`.

—check_ext

Der Parameter `--check_ext` gibt einen GPL-Header für eine *eisfair* -check.ext-datei aus.

Beispiel:

```
printgpl --check_ext            \
         "foo"                  \
         "2005-03-25" "frank"   \
         2001-2013 the eisfair team, team(at)eisfair(dot)org"
```

Beispiel:

```
# ----------------------------------------------------------------------
# /etc/check.d/foo.ext - eischk ext file for foo
#
# Creation:
```

Im Beispiel wird der GPL-Header für das angegebene Paket ausgegeben.

Das Ergebnis entspricht dem vorangegangenen Beispiel `printgpl`.

9.3.3 printgroup

Mit der Funktion `printgroup()` kann man einen Gruppenheader für eine *eisfair* -Konfigurationsdatei erzeugen. Dabei kann nach dem Gruppennamen noch optional ein Kommentar angegeben werden:

Beispiel:

```
printgroup "Simple group"
printgroup "General settings" "General settings for configuration"
```

Beispiel: Für eine Zeile

```
# ----------------------------------------------------------------------
# Simple group
# ----------------------------------------------------------------------
```

Beispiel: Für eine Zeile mit Kommentar

```
# ----------------------------------------------------------------------
# General settings
# general settings for eisfair configuration
# ----------------------------------------------------------------------
```

Standard ist vor und nach dem Gruppenheader jeweils eine Leerzeile.

Diese Leerzeile kann gesteuert werden. Dafür gibt es die folgenden Parameter. Hierbei lässt sich durch Angabe von '0' die Leerzeile auch unterdrücken.

Parameter:

```
-b x     print x newline before Groupheader
-a x     print x newline after Groupheader
```

Beispiel:

```
printgroup -b 0 "Simple group"
printgroup -b 2 -a 0 "General settings" "General configuration settings"
```

9.3.4 printcustomgroup

Mit der Funktion `printcustomgroup()` kann man einen Gruppenheader für eine *eisfair* -Konfigurationsdatei erzeugen und einen mehrzeiligen Kommentar angeben:

Beispiel:

```
printcustomgroup 'General settings' << !EOC
comment line
comment line
comment line
!EOC
```

Beispiel:

```
# ----------------------------------------------------------------------
# General settings
#
# more comment
# more comment
# etc
# ----------------------------------------------------------------------
```

Standard ist vor und nach dem Gruppenheader jeweils eine Leerzeile.

Diese Leerzeile kann gesteuert werden. Dafür gibt es die folgenden Parameter. Hierbei lässt sich durch Angabe von '0' die Leerzeile auch unterdrücken.

Parameter:

```
-b x     print x newline before Groupheader
-a x     print x newline after Groupheader
```

Beispiel:

```
printcustomgroup -b 0 'General settings' << !EOC
comment line
comment line
comment line
!EOC
```

Wenn dieser mehrzeilige Kommentar tabellarisch gestaltet werden soll und dabei am Zeilenanfang Leerzeichen vorkommen, müssen diese am Beginn der Zeile mit der '#' Raute geschützt werden.

9.3.5 printvar

Die Funktion `printvar()` dient zur formatierten Ausgabe von Konfigurationsvariablen und Kommentaren, sofern eine *eisfair* -Konfigurationsdatei selbst geschrieben werden muss.

Beispiel:

```
printvar 'FOO_VARIABLE'            "This is a simple Variable"
printvar ''                        "This is a comment without a Variable"
printvar 'FOO_ARRAY_'$idx'_OPTION' "Simple printing of Arrays"
```

Als Beispiel sind hier drei gängige Aufrufvarianten angegeben. Als erstes die einfache Ausgabe einer Variable mit einem Kommentar. Sollte die Variable `FOO_VARIABLE` den Inhalt „`foobar`“ haben, so sieht die Ausgabe wie folgt aus:

```
FOO_VARIABLE='foobar'                  # This is a simple Variable
```

Das Kommentarzeichen „`#`“ wird dabei unabhängig von der Länge des Variablennamens und des Inhalts immer an Position 39 ausgegeben. Sollten der Variablenname und der Inhalt länger als 37 Zeichen sein, so wird der Kommentar automatisch in der Folgezeile an Position 39 ausgegeben.

Um längere Kommentare mehrzeilig ausgeben zu können, kann `printvar` auch ohne einen Variablennamen aufgerufen werden. Es wird dann nur der übergebene Kommentar ab Position 39 ausgegeben. Das ist im zweiten Beispiel dargestellt.

Auch Variablenfelder können einfach mit `printvar` ausgegeben werden, dazu ist einfach der Variablenname inclusive des Zählers an `printvar` zu übergeben (Beispiel 3).

9.3.6 printend

Mit der Funktion `printend()` wird eine *eisfair* -Konfigurationsdatei abgeschlossen, d.h. es wird die Fusszeile „`End`“ausgegeben:

Beispiel:

```
printend
```

9.4 Weitere verfügbare Skripte

9.4.1 /var/install/bin/ask

Das Skript `ask` dient zur interaktiven Abfrage einer Eingabe. Aus diesem Grund darf dieses Skript nicht im Zusammenhang mit webconf verwendet werden!

```
/var/install/bin/ask [--tty --info --warn --error] QUESTION [DEFAULT] [PATTERN

QUESTION  -  Question to be asked.
DEFAULT   -  Default answer.
PATTERN   -  Possible answers:
```

```
empty     -  y(es) or n(o)
*         -  string input
+         -  string input (not empty!)
min-max   -  numerical input (range given)
WORD|WORD -  a list of possible values
             the first will be included into the question
             the last will be returned if choosen
WORD=WORD -  a list of possible values
             all will be included into the question
             the first will be returned if choosen
^$        -  ENTER

Return Values:

Exitcode:    Zero-Based Index of the Choosen PATTERN
             255 ask was aborted by typing CTRL-C

Output:      If Output is redirected, the output is either a part
             of the matching pattern (if PATTERN is given) or the
             entered text


--info     print info message     (green)
--warn     print warning message  (yellow)(bright)
--error    print error message    (red)(brightinvers)

--tty      use console colors
```

Beispiele:

```
ask "Kernel loeschen"
ask "Kernel loeschen" "no"
ask "Cronzyklus" "" "d|daily" "w|weekly" "m|monthly"
ask "Hostname" "eistest" "*"
ask "Hostname" "" "+" >result
answer=$(ask "Weiter" "y")
ask "Anzahl" "3" "1-7" >result
answer=$(ask "Select" "" "1-15" "^$=Return" "0=Exit")
ask --tty --info "Kernel loeschen" "no"
```

Um den Rückgabewert von 'ask' bei einem Abruch mit CTRL-C abfangen zu können, hat sich die folgende Variante bewährt:

```
_ask_tmpfile=$(/bin/mktemp -t ask.XXXXXXXXX)
/var/install/bin/ask "Select" "" "1-8" "^$=Return" "0=Exit" > ${_ask_tmpfile}
rc=${?}
read answer < ${_ask_tmpfile}
rm -f ${_ask_tmpfile}
```

```
# if ask break, ask returned 255
if [ ${rc} = 255 ]
then
    answer=0
fi

case "${answer}" in
    '')
        exit 0
    ;;
    0)
        exit 127
    ;;
    *)
        tue das andere
    ;;
esac
```

9.4.2 /var/install/bin/choose

Ueber den Aufruf von `choose` kann eine Auswahlliste dargestellt werden. `choose` übernimmt dazu die gesamte Oberflächendarstellung.

- Darstellung in Tabellenform (mittels `techo`)
- Automatische Nummerierung der Zeilen
- Rücksprung ohne eine Auswahl
- Vorwärts- und rückwärtsblättern
- Automatische Anpassung an Bildschirmgrösse (horizontal und vertikal)
- Direktanzeige einer gewünschten Zeile beim Erstaufruf möglich
- Deaktivieren einzelner Zeilen

Sofern im Environment `SCROLL="yes"` gesetzt ist, wird auf das Blättern verzichtet. Es werden dann immer alle Zeilen ausgegeben und der Anwender muss manuell scrollen.

In diesem Fall sind verschiedene der obigen Features natürlich nicht verfügbar.

choose kann nicht in Verbindung mit webconf genutzt werden.

Syntax:

```
/var/install/bin/choose ARRAY [START_POSITION]
```

Die übergabe der Daten an `choose` erfolgt analog der *eisfair* Konfigurationsdateien als mehrdimensionales Array. Hierfür wird im Folgenden `MYARRAY` verwendet.

Neben den eigentlichen Inhalten müssen folgende Werte übergeben werden:

Beispiel:

```
MYARRAY_TITLE="Mein Titel"        # Titelzeile
MYARRAY_SUBTITLE="Untertitel"     # optional ein Untertitel
MYARRAY_QUESTION="Select Item"    # Frage
MYARRAY_COLS="10 32r 10*"         # Spaltendefinition (analog techo)
MYARRAY_ROWS=345                  # Anzahl Zeilen
```

Bei der Spaltendefinition darf die erste Spalte mit den Auswahlnummern nicht mit angegeben werden. Diese wird von choose automatisch generiert.

Die Inhalte werden als Array übergeben:

Beispiel:

```
MYARRAY_1_1="Test"
MYARRAY_1_2="foo"
MYARRAY_1_3="1234"
MYARRAY_2_1="Beispiel"
MYARRAY_2_2="bar"
MYARRAY_2_3="5678"
```

Hier muss besonders darauf geachtet werden, dass die Anführungszeichen korrekt gesetzt werden:

Beispiele:

```
eval MYARRAY_$row_1=\"foo\"
eval MYARRAY_$row_1=\"'--info \"'foo bar foo bar'\"'\"
eval MYARRAY_$row_1=\"'--warn \"'$FOOBAR'\"'\"
```

ACHTUNG

Alle Werte müssen per export an Subshells übergeben werden!

Nun kann choose aufgerufen werden:

Beispiel:

```
_ask_tmpfile=$(/bin/mktemp -t XXXXXXXXXXXXX)
/var/install/bin/choose MYARRAY 89  > ${_ask_tmpfile}
rc=${?}
read answer < ${_ask_tmpfile}
rm -f ${_ask_tmpfile}

[ $rc = 255 ] && exit 255
```

In diesem Beispiel wird mit der Anzeige direkt auf der Seite begonnen, auf der sich das im Array in Zeile 89 übergebene Element befindet.

Rückgabewerte:

```
""                  Return ohne Auswahl; Ruecksprung in vorherige Ebene
"1" - "9999"        Ausgewaehltes Element
"0"                 Ende; sofortiges Beenden der Anwendung
```

über die Angabe `MYARRAY_ACTIVE="no"` kann man einzelne Spalten aus der Auswahllsite ausblenden. Damit wird diese Spalte in der Auswahlliste nicht berücksichtigt. Das ist z.B. bei jeglicher Art von Aufgabenlisten praktisch, da man so die bereits abgearbeiteten Punkte einfach aus der Liste entfernen kann, ohne diese komplett umsortieren oder gar neu aufbauen zu müssen.

Um auf jeder Seite wiederkehrende Spaltenüberschriften ausgeben zu können, werden diese per

Spaltenüberschriften

```
MYARRAY_CAPTION_1='--info "Spalte 1"'
MYARRAY_CAPTION_2='--info "Spalte 2"'
```

übergeben. Die Verwendung von Spaltenüberschriften ist optional.

Mittels `MYARRAY_FLAGS` können die folgenden optionalen Parameter übergeben werden:

- `--spread`
- `--indent` *12*
- `--showexit`
- `--list`

Bei der Angabe von `--spread` werden zwischen den einzelnen Zeilen jeweils eine Leerzeile ausgegeben und die gesamte Liste wird auf dem Bildschirm vertikal zentriert. Diese Funktion ist nur verfügbar, wenn die gesamte Liste auf einer Seite dargestellt werden kann. Ansonsten wird dieser Parameter ignoriert und es erfolgt eine normale Ausgabe.

Mittels `--indent` *12* kann ein grösserer oder kleinerer linker Einzug angegeben werden.

Bei der Angabe von `--showexit` wird auf jeder Seite als letzter Punkt die Funktion „0“ ausgegeben:

Bei der Angabe von `--list` werden keine Nummerischen Werte ausgegeben und es gibt auch keine Auswahl fuer Numerische Werte. Es ist dann eine Anzeige.

```
--showexit

    1. Gallia est omnis divisa in partes tres, quarum unam incolunt
    2. Belgae, nostra Galli appellantur. hi omnes lingua, institutis,
    3. legibus inter a Belgis Matrona et Sequana dividit. horum omnium
    4. fortissimi sunt
    0. Return


  145. agmine proelio nostros lacessere coeperunt. Caesar suos a proelio
  146. pabulationibusque prohibere. ita dies circiter quindecim iter
  147. fecerunt, non amplius quinis aut senis milibus passuum interesset.
    0. Exit
```

Um eigene Werte, die nicht angezeigt werden sollen, im Array abzuspeichern muss der Bereich MY (`MYARRAY_MY_FOO`) verwendet werden. Dadurch kann es zu keinen überschneidungen der Namen bei zukünftigen Erweiterungen der übergabestruktur kommen.

9.4.3 /var/install/bin/check-version

überprüft ein installiertes Paket auf seine Version

```
check-version package_name check_version
check-version -diff version-1 version-2

package_name   Paketname
check_version  Versionsnummer z.B. 1.0.1

Rueckgabewerte:

new            Abgefrage Version ist aelter als die installierte
old            Abgefrage Version ist neuer als die installierte
installed      Paket ist installiert
not-installed  Paket ist nicht installiert
```

Beispiel:

```
if [ "$(/var/install/bin/check-version foo 1.0.0)" = "not-installed" ]
then
    mecho --error "Required foo package not installed."
    mecho --error "Installation aborted."
    exit 1
fi
```

9.4.4 /var/install/bin/doc

Zeigt die Dokumentation eines Paketes an. Dabei ist der komplette Pfad zur Dokumentationsdatei mit zu übergeben.

```
doc file
doc file message

file           Dateiname der Dokumentationsdatei
message        Kopfzeile
```

Beispiel:

```
/var/install/bin/doc /usr/share/doc/foo/foo.txt "View foo documentation"
```

9.4.5 /var/install/bin/edit

Startet den Editor zum Editieren der Konfiguration

```
edit [--apply|--apply-stopstart] file
edit [--apply|--apply-stopstart] package message

--apply           Aenderungen werden durch einen Restart sofort
```

```
                  uebernommen.
--apply-stopstart Im Gegensatz zum Schalter --apply wird der Dienst hier
                  erst gestoppt, dann wird die Konfiguration geschrieben
                  und erst dann erfolgt der erneute Start des Dienstes.
file              Dateiname der Konfigurationsdatei
package           Name des Paketes
message           Kopfzeile
```

9.4.6 /var/install/bin/add-menu

add-menu dient zur Erstellung eines Menüeintrages (Untermenü) für das Setup

```
add-menu [--menu] menu-file sub-menu-file comment
add-menu --script menu-file script-file comment

--menu         Untermenue (<menu>-Tag) erstellen   [default]
--script       Skriptaufruf (<script>-Tag) anstelle des Untermenues
               erstellen

menu-file      Dateiname relativ zum Menueverzeichnis /var/install/menu

               Der neue Eintrag wird an dieses Menue angefuegt.

sub-menu-file  Dateiname relativ zum Menueverzeichnis /var/install/menu

               Der neue Eintrag
                    <menu file="sub-menu-file">comment</menu>
               wird am Ende des Menues "menu-file" angefuegt, sofern
               ein solcher Eintrag noch nicht im Menue vorhanden ist.

script-file    Dateiname relativ zu /var/install/bin oder absoluter Pfad

               Der neue Eintrag
                    <script file="script-file">comment</script>
               wird am Ende des Menues "menu-file" angefuegt, sofern
               ein solcher Eintrag noch nicht im Menue vorhanden ist.

comment        Text, der im Menue angezeigt werden soll
```

9.4.7 /var/install/bin/create-menu

create-menu dient zum Anlegen einer komplett neuen Menüdatei

```
/var/install/bin/create-menu menu-file menu-title

menu-file      Dateiname relativ zum Menueverzeichnis /var/install/menu.
```

```
menu-title     Menuetitel der ueber dem Menue angezeigt werden soll.
```

9.4.8 /var/install/bin/del-menu

del-menu dient zur Entfernung eines Menüeintrags aus dem Setup

```
del-menu menu-file sub-menu-file
del-menu menu-file script-file

menu-file      Dateiname relativ zum Menueverzeichnis /var/install/menu

sub-menu-file  Dateiname relativ zum Menueverzeichnis /var/install/menu

               Die Zeile <menu file="sub-menu-file"> mit dem angegebenen
               Untermenue wird aus dem Menue entfernt, sofern es diese
               gibt.

script-file    Dateiname relativ zu /var/install/bin oder absoluter Pfad

               Die Zeile <script file="script-file">comment</script>
               mit dem angegebenen Skriptaufruf wird aus dem Menue
               entfernt, sofern es diese gibt.
```

9.4.9 /var/install/bin/add-user

Fügt einen neuen Benutzer zum System hinzu. Wenn das Skript ohne Parameter aufgerufen wird, werden die Daten einzeln abgefragt. Zum Anlegen eines Benutzers ohne den Eingabedialog können folgende Parameter übergeben werden:

```
add-user [-d|-l] user encrypted-password uid gid name home shell

-d             zum erstellen eines Benutzers der sich ohne Passwort
               anmelden darf
-l             zum Sperren des Benutzerpasswortes
user           Benutzername
encrypted-password Passwort in kodierter Form
uid            Benutzer ID Nummer
gid            Gruppen ID Nummer
name           Beschreibung de Benutzers
home           Basisverzeichnis des Benutzers
shell          Kommandointerpreter des Benutzers
```

Beispiel für das Anlegen eines Benutzers:

```
/var/install/bin/add-user \
    foo sdz8evwesdfs 2005 100 "User foo" /home/foo /bin/bash
```

Beispiel für das Anlegen eines Systembenutzers ohne Anmeldeberechtigung:

```
/var/install/bin/add-user \
    firebird '*' 84 84 "Firebird SQL deamon" /var/firebird /bin/bash
```

Beispiel für das Anlegen eines Benutzers, der sich ohne Passwort anmelden darf:

```
/var/install/bin/add-user \
    -d foo '' 2005 100 "User foo" /home/foo /bin/bash
```

9.4.10 /var/install/bin/modify-user

ändert diverse benutzerspezifische Parameter, wie den Kommentar (-c), das Home-Verzeichnis (-d), das Ablaufdatum eines Benutzerkontos (-e), der Anzahl der Tage, nach denen das Benutzerkonto gesperrt wird (-f), die Benutzergruppe (-g) und die verwendete System-Shell (-s). Wird beim Aufruf des Skriptes keine Option angegeben, so wird ein Menü angezeigt, über welches die änderungen interaktiv durchgeführt werden können.

```
modify-user
modify-user -c login-name comment
modify-user -d login-name home-directory [-m|-r]
modify-user -e login-name YYYY-MM-DD
modify-user -f login-name number-of-days
modify-user -g login-name group-name
modify-user -s login-name user-shell

login-name     Loginname des zu bearbeitenden Benutzerkontos.

comment        Kommentar.

home-directory Name des Home-Verzeichnises des Benutzers in /home.
               Durch die zusaetzliche Angabe einer der folgenden
               Optionen kann das Verhalten dieser Funktion beinflusst
               werden.

               -m - Das Ursprungsverzeichnis und dessen Inhalt wird
                    verschoben.

               -r - Es wird ein neues Home-Verzeichnis angelegt und das
                    Ursprungsverzeichnis wird geloescht.

               '' - Es wird ein neues Home-Verzeichnis angelegt, das
                    Ursprungsverzeichnis bleibt unveraendert erhalten.

YYYY-MM-DD     Ablaufdatum des Benutzerkontos. Wird 'never' eingegeben,
               so kann das Benutzerkonto unbefristet genutzt werden.

number-of-days Anzahl der Tage nach denen ein Benutzerkonto deaktiviert
```

```
                wird wenn dessen Kennwort abgelaufen ist. Bei Eingabe von
                '0' geschieht dies umgehend, wird 'never' eingegeben, so
                wird diese Funktion deaktiviert.

group-name      Name der Benutzergruppe.

user-shell      Name der System-Shell
```

9.4.11 /var/install/bin/remove-user

Entfernt einen Benutzer vom System. Systembenutzer können dabei nur durch Setzen des Schalters "-f"entfernt werden.

```
remove-user [-f] user do-remove-homedir

-f              Systembenutzer loeschen
user            Benutzername
do-remove-homedir 'y' oder 'n' zum Loeschen des Basisverzeichnisses
```

Beispiel für das Entfernen eines Benutzers mit Löschen des Basisverzeichnisses:

```
/var/install/bin/remove-user foo y
```

Beispiel für das Entfernen eines Systembenutzers ohne Löschen des Basisverzeichnisses:

```
/var/install/bin/remove-user -f firebird n
```

9.4.12 /var/install/bin/add-group

Fügt eine neue Gruppe zum System hinzu. Wenn das Skript ohne Parameter aufgerufen wird, werden die Daten einzeln abgefragt.

```
add-group group gid

group           Name der Gruppe
gid             Gruppen ID Nummer
```

Beispiel für das Anfügen einer Gruppe:

```
/var/install/bin/add-group foo 2001
```

9.4.13 /var/install/bin/remove-group

Entfernt eine Gruppe vom System. Systemgruppen können dabei nur durch Setzen des Schalters "-f" entfernt werden.

```
remove-group [-f] group

-f              Systemgruppe loeschen
```

```
group          Gruppenname
```

Beispiel für das Entfernen einer Gruppe:

```
/var/install/bin/remove-group foo
```

9.4.14 /var/install/bin/install-local-package

über dieses Skript ist es möglich, interaktiv Pakete zu installieren, die sich auf der lokalen Festplatte befinden, ohne dass zuvor manuell eine eis-list.txt-Datei erstellt werden muss.
Nach Eingabe des Verzeichnisnamens, in dem sich die Paket- und .info-Dateien befinden, wird dynamisch eine eis-list.txt-Datei erzeugt, welche dann menügeführt die Installation der Pakete ermöglicht. Das Skript merkt sich die zuletzt verwendeten Verzeichnisse und bietet diese im interaktiven Modus zur Auswahl an.
Zusätzlich ist es möglich, dem Skript über die Kommandozeile ein Verzeichnis mitzugeben bzw. das Löschen der temporären index.txt oder eis-list.txt-Dateien auf Wunsch zu verhindern (-keep-all). Wer diese Installationsfunktion fest in das existierende eisfair-Installationsmenü aufnehmen möchte, kann dies über den Switch '-add-menu' bewerkstelligen. Der Switch '-del-menu' entfernt den Menüeintrag wieder.
Falls man das Skript nur für die Generierung der index.txt und eis-list.txt-Dateien verwenden möchte, kann man über den Switch '-no-install' den Aufruf der Installationsroutine verhindern.

Die verschiedenen Startvarianten im überblick:

```
install-local-package
  oder
install-local-package [--keep-all][--no-install] Verzeichnisname
  oder
install-local-package --add-menu
  oder
install-local-package --del-menu

Optionen: --add-menu   - Menueeintrag zum Installationsmenue hinzufuegen
          --del-menu   - Menueeintrag aus Installationsmenue entfernen
          --keep-all   - eis-list.txt und index.txt-Dateien nicht
                         loeschen
          --no-install - Nach dem Generieren der eis-list.txt und
                         index.txt-Dateien das Installationsskript
                         nicht starten.
```

9.4.15 /var/install/bin/sort-menu

Dieses neue Skript erlaubt die alphabetische Sortierung von Menü-Einträgen einer angegebenen Menüdatei.

```
sort-menu menu-file-name

menu-file-name     Name einer Menuedatei
```

9.4.16 /var/install/bin/check-package

Mit diesem Skript lässt sich überprüfen, ob ein Paket vorhanden ist. Ist das nicht der Fall, wird die Installation im Dialog eingeleitet.

Intern arbeitet dieses Skript mit 'check-version' zusammen.

Die Aufruf-Syntax

```
check-package
             -p, --setpackage      package_name
             -v, --setversion      package_version
             msg_do_now            write a message do now

Example:  check-package -p perl -v 1.2.0 "SUBVERSION_TOOLS_USE_PERL"
```

Die Angabe von -v, —setversion ist nicht zwingend, es kann auch nur auf das Paket geprüft werden.

Beispiel für die Prüfung eines Paketes und das Auswerten des Rückgabewertes.

Als Parameter dient in diesem Beispiel EISFAX_USER_1_PRN_INFO='yes' Für das Drucken der Information wird das Paket 'a2ps' benötigt.

```
prn_info='${EISFAX_USER_1_PRN_INFO}'

case ${prn_info} in
    yes)
      /var/install/bin/check-package -p "a2ps" "EISFAX_USER_1_PRN_INFO"

        case ${?} in
            0)
                : # alles OK (nun in die configfiles eintragen)
            ;;

            *)
                # nicht OK
                # Paramneter zuruecksetzen

      sed "s/^EISFAX_USER_1_PRN_INFO=.*/EISFAX_USER_1_PRN_INFO='no'/" \
                          /etc/config.d/eisfax >/tmp/CONFIG_FILE_TMP
                mv /tmp/CONFIG_FILE_TMP /etc/config.d/eisfax
                chmod 0600 /etc/config.d/eisfax

                # Fehlermeldung ausgeben

                mecho
                mecho --info " EISFAX_USER_1_PRN_INFO="
```

```
                mecho --info " a2ps
                mecho --warn " was not installed and so"
                mecho --warn " not added to the config files."
                mecho --warn " Please solve that first"
                mecho
                anykey
            ;;
        esac
    ;;

    *)
        : # nothing to do
    ;;
esac
```

9.4.17 /var/install/bin/check-folder

Mit diesem Skript lässt sich überprüfen, ob ein Folder/Verzeichnis vorhanden ist. Ist das nicht der Fall, wird das Verzeichnis im Dialog angelegt.

Die Aufruf-Syntax:

```
  check-folder
              -f, --setfolder       /x/y/z
              msg_do_now            write a message do now

  Example:
check-folder -f /tmp/myfolder/attention "EISFAX_USER_${idx}_COPY_TO_PATH"
```

9.4.18 /etc/init.d/functions

Dieses Skript stellt Funktionen für das Init-Skript bereit. Es wird analog der eislib in das Init-Skript eingebunden. Das benutzen der eislib im Init-Skipt ist nicht erwünscht.

Beispiel für ein Init-Script:

```
#!/bin/sh
# ----------------------------------------------------------------------
# /etc/init.d/foo - init script for foo
#
# Creation:     2010-12-30  hbfl
# Last Update:  2010-12-30  hbfl
#
# Copyright (c) 2001-2013 the eisfair team, team(at)eisfair(dot)org
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
```

```
# ----------------------------------------------------------------------

# include functions
. /etc/init.d/functions

# include configuration
. /etc/config.d/foo

_do_start ()
{
    if ${forcestart:-false}
    then
        START_FOO=yes
    fi

    ### check if starting foo is allowed
    if [ "$START_FOO" = 'yes' ]
    then
        boot_mesg " * Starting foo ..."
        loadproc foo
    fi
}

_do_stop ()
{
    boot_mesg " * Stopping foo ..."
    killproc foo
}

_do_status ()
{
    statusproc foo
}
_do_usage ()
{
    echo "Usage: $0 [-q|--quiet] {start|forcestart|stop|restart}"
{

#-----------------------------------------------------------------------
# main
#-----------------------------------------------------------------------

while [ ${#} -gt 0 ]
do
    case "${1}" in
        -q|--quiet)
            _quiet=true
            shift
        ;;
        *)
            _action=${1}
            shift
        ;;
```

```
    esac
done

case "$(_action}" in
    start)
        _do_start
    ;;
    forcestart)
        forcestart=true
        _do_start

    ;;
    stop)
        _do_stop
    ;;
    restart)
        _do_stop
        sleep 3
        _do_start
    ;;
    status)
        _do_status
    ;;
    *)
        _do_usage
        exit 1
    ;;
esac

exit 0
```

Die Funktionen im Überblick

`boot_mesg` **Ausgabe einer allgemeinen Meldung**

Dies sollte der Standardfall bei der Ausgabe von Meldungen sein. Ein Beispiel dafür ist die Ausgabe von Statusmeldungen beim Starten oder Stoppen eines Dienstes. .

```
 * Starting foo ...
```

`loadproc` **Angabe des Programms das gestartet werden soll**

`killproc` **Angabe des Programms das gestoppt werden soll**

Dabei wird automatisch das Ergebnis [OK] [FAIL] [WARN] am rechten Rand der Arbeitsfläche hinter der boot_mesg Ausgabe gesetzt.

Beispiel:

```
boot_mesg " * Starting foo ... "
```

```
loadproc foo
```

```
* Starting foo ...                                          [  OK  ]
```

`statusproc` **Angabe des Programms für das der Status ausgegeben werden soll**

.

Unhabhängig von load- killproc lassen sich auch die Ausgaben von [OK] [FAIL] [WARN] erzeugen.

`echo_ok` **Ausgabe von [OK]**

.

`echo_failure` **Ausgabe von [FAIL]**

`echo_warning` **Ausgabe von [WARN]**

Diese Ausgaben erfolgen jeweils eine Zeile Höher an den rechten Rand der Arbeitsfläche.

Beispiel:

```
boot_mesg " Do foo ... "
_do_foo
retval="${?}"
if [ "${retval}" -eq 0 ]
then
    echo_ok
else
    echo_failure
    _do_bar
fi
```

Das Skript functions unterdrückt die Ausgaben auf die Console, wenn der Parameter _quiet=true gesetzt ist.

9.4.19 /var/install/bin/convert-encoding

Mit diesem Skript lässt sich die Kodierung in einem Skript an das vorhandene Encoding auf dem System anpassen. Wenn das Skript in UTF-8 Kodiert ist und das System UTF-8 Kodierung hat, wird nichts ausgeführt, wenn das System in ISO-8859-15 laeuft wird das Skript nach ISO-8859-15 Kodiert.

Die Aufruf-Syntax:

```
  convert-encoding
                   /pfad/zum/skript/

  Example:
convert-encoding  /tmp/myfolder/myskript
```

10 Curses basierte Programme

10.1 Übersicht

Eisfair nutzt neben umfangreichen Scriptdateien auch Programme, welche als Benutzerinterface die Funktionen der ncurses Programmbibliothek verwenden. Diese sind im Verzeichnis `/var/install/bin` installiert und an der Dateiendung .cui erkennbar.

10.2 Documentation Viewer - show-doc.cui

Zur Anzeige von Dateien im Textformat dient das Programm show-doc.cui. Es findet derzeit hauptsächlich Verwendung für die Präsentation der Paketdokumentation. Eine über die Funktionstaste F7 erreichbare Suchfunktion erleichtert dabei das Auffinden von Textstellen.

Ein weiterer Anwendungsbereich für das Programm show-doc.cui ist die Verwendung als Protokoll-Viewer. Mit der eingebauten 'tail' Funktion zeigt das Programm neue Daten am Ende einer Textdatei in Echtzeit an.

Beim Aufruf von show-doc.cui wird der Name der Textdatei übergeben, die angezeigt werden soll. Ein *.gz oder *.bz2 Archiv wird dabei automatisch extrahiert. Zudem sind eine Reihe von Kommandozeilenoptionen verfügbar, um das Programm zu beeinflussen:

```
/var/install/bin/show-doc.cui [Optionen] Dateiname
     -c, --color           Programm im Farbmodus ausfuehren
         --nocolor         Programm im S/W-Modus ausfuehren
     -m, --mouse           Maus als Eingabegeraet verwenden
         --nomouse         Keine Maus fuer die Eingabe verwenden
     -v, --version         Programmversion anzeigen
     -t, --title=<titel>   Programmtitel setzen
     -f, --follow          'tail'-Funktion aktivieren
     -h, --help            Hilfe anzeigen
```

Beispiel für die Anzeige einer Datei:

```
/var/install/bin/show-doc.cui /usr/share/doc/foo/foo.txt \
                              --title="foo program options"
```

Neben der eigentlichen Textdatei, die betrachtet werden soll, kann im selben Verzeichnis eine Datei bereitgestellt werden, die ein Inhaltsverzeichnis zur Textdatei enthält. Dazu muss

die Datei den selben Namen wie die anzuzeigende Originaldatei tragen, wobei deren Endung (z.B. ".txt") gegen die Endung ".toc" auszutauschen ist. Besitzt die Originaldatei keine Endung, dann wird an deren Namen die Endung ".toc" angehängt.

Wird beim Öffnen eines Dokuments eine solche ".toc"-Datei gefunden, kann in show-doc.cui das Inhaltsverzeichnis mit der F4-Taste aufgerufen werden. Darin ist dann die direkte Navigation zum jeweiligen Kapitel möglich.

Der Inhalt der Indexdatei folgt dem XML-Format, wobei alle Einträge mit dem Tag "index" eingeschlossen werden. Der "index"-Tag kennt zudem das Attribut "title", das die Überschrift des Inhaltsverzeichnisses festlegt.

Unterhalb des "index" Tags werden die Sprungziele mit dem Tag "entry" eingeleitet. Im Datenbereich zwischen dem öffnenden und dem schließenden Tag steht dabei der Text der als Kapitelüberschrift angezeigt werden soll. Zudem existieren die beiden Attribute "line" und "level". Während "line" die Textzeile in der Originaldatei angibt, ab der das jeweilige Kapitel beginnt, gibt "level" die Ebene der Überschrift und damit deren Einrückung im Index an.

Beispiel für eine .toc Datei:

```
<!-- PostgreSQL Table Of Contents -->
<index title="PostgreSQL Eisfair Dokumentation">

    <entry line="36" level="1">1 Allgemein</entry>
    <entry line="66" level="1">2 Installation</entry>
    <entry line="95" level="1">3 Konfiguration</entry>
    <entry line="102" level="2">3.1 Allgemeine Einstellungen</entry>
    <entry line="140" level="2">3.2 Zugriffstabelle</entry>
    <entry line="197" level="2">3.3 Sicherung</entry>
    ...

</index>
```

10.3 File and directory listing tool - list-files.cui

Vielfach wird das Auflisten und Auswählen bestimmter Dateien und Verzeichnisse benötigt. Dabei müssen diese mitunter, an Hand von charakteristischen Merkmalen, aus größeren Beständen herausgefiltert werden. Ein optionaler Abfragedialog leitet nach der Eintragsauswahl das Abarbeiten eines externen Scripts ein.
Viele Schalter und Parameter sollen eine möglichst flexible Verwendung des Programms erlauben:

```
--title=Titel
      Titel des Auswahldialogs
--column=Spaltenueberschrift
      Ueberschrift der ersten Spalte
--path=/etc
```

```
      zu durchsuchendes Verzeichnis
      (Unterverzeichnisse werden nicht durchsucht)
--filter="*.conf,*.cfg,*.cnf"
      Optionale Liste mit Platzhaltern zur Einschraenkung
      der aufgelisteten Eintraege. Zulaessig sind:
      ?     1 Zeichen
      *     beliebige Zeichenanzahl
      [0-9] Zeichenmenge
--only=1
      Optionale Beschraenkung der Anzeige auf:
      1=Dateien
      2=Verzeichnisse
      3=Links
      4=Sockets
--quest="Restore config from %s ?"
      Optionale Abfrage vor der Scriptausfuehrung. Mit
      %s kann hier der gew"ahlte Dateiname angezeigt werden.
--script=/run.sh
      Dateiname des Scripts, welches nach der Auswahl ausgefuehrt
      wird. Der ausgewaehlte Name wird als 1. Parameter uebergeben.
-d    zeigt zusaetzlich Datum und Uhrzeit der letzten Dateiaenderung
-m    zeigt zusaetzlich die Dateiatribute an
-n    zeigt zusaetzlich die Groesse an
-u    zeigt zusaetzlich User und Group der Eintraege an
-w    wartet nach Abarbeitung des Scripts auf einen Tastendruck
--helpfile=/var/install/help/foo
      Dateiname der XML-Hilfedatei
--helpname=FOO_MENU_FILEDIALOG
      XML-Tag der zugehoerigen Hilfe <help name='FOO_MENU_FILEDIALOG'>
--helpview
      blendet die Hilfe nach dem Programmstart sofort ein
```

Beispiel an Hand des MySQL Datenbank Restore Dialogs:

```
/var/install/bin/list-files.cui \
  --title="Select SQL file" \
  --column="Restore from:" \
  --path=/var/lib/mysql_backup
  --filter="*-????????-[0-2][0-9].sql.gz" \
  --only=1 \
  -d -n -w \
  --quest="Restore database from %s ?" \
  --script="/usr/local/mysql/bin/mysqlrestore.sh $user $pass /var/lib/mysql" \
  --helpfile=/var/install/help/mysql \
  --helpname=MYSQL_MENU_RESTORE \
  --helpview
```

10.4 Curses Frontend für Shell-Scripte - shellrun.cui

Mit Hilfe des Programms shellrun.cui lassen sich Shell-Scripte mit Curses-Oberfläche schreiben. Das Konzept ist dem Programm "dialog" ähnlich, geht jedoch in sofern darüber hinaus, als dass die Shell-Skripte unter shellrun.cui interaktiv mit der Oberfläche in Verbindung stehen und diese dynamisch beeinflussen können.

Allerdings ist die Script-API des Programms recht komplex und darum in eine separate Dokumentation [libcuidoc] ausgelagert.

Der Aufruf von shellrun.cui kann mit den folgenden Kommandozeilen- Argumenten erfolgen:

```
/var/install/bin/shellrun.cui [Optionen] Script-Datei
     -o  --args=<Argumente> Argumente, die an die Skriptdatei
                            uebergeben werden sollen
         --debug            Debug-Modus
     -c, --color            Programm im Farbmodus ausfuehren
         --nocolor          Programm im S/W-Modus ausfuehren
     -m, --mouse            Maus als Eingabegeraet verwenden
         --nomouse          Keine Maus fuer die Eingabe verwenden
     -v, --version          Programmversion anzeigen
     -t, --title=<titel>    Programmtitel setzen
     -f, --follow           'tail'-Funktion aktivieren
     -h, --help             Hilfe anzeigen
```

11 Verwandte Themen

11.1 Einrichtung eines Downloadservers

eisfair benutzt wget, um Pakete von einem Server herunterzuladen. Dies kann ein ftp- oder ein http-Server sein. Als URL muss eine Text-Datei angeben werden, in der das Inhaltsverzeichnis der verfügbaren Pakete aufgeführt ist. Per Default heissen diese Dateien eis-list.txt.

Hier ein Beispiel von http://download.eisfair.org/packages/eis-list_web.txt

```
#<comment> Package list of download.eisfair.org
#<comment> Copyright Frank Meyer <frank(at)eisfair(dot)org>
#<welcome> Welcome to the official eisfair package server www.eisfair.org
#<index>   index.txt
#<empty>
#<message> Web packages (Apache / Squid):
#<empty>
#<info>    apache.tar.gz.info
#<info>    apache_php4.tar.gz.info
#<info>    apache_htdig.tar.gz.info
#<info>    apache_webalizer.tar.gz.info
#<info>    apache_mod_gzip.tar.bz2.info
#<info>    php_cgi.tar.gz.info
#<empty>
#<info>    squid.tar.gz.info
#<empty>
#<message> Last database update: 2004-05-13 19:52:10
```

Zu den Tokens:

`<comment>`	Kommentar
`<welcome>`	Titelzeile auf dem eisfair-Server (in grün)
`<index>`	Referenzierung der index-Datei für `<require-package>`
`<version>`	Zeigt „Version xxxx“
`<empty>`	Erzeugt Leerzeile
`<message>`	Gibt Zeile als Meldung aus (eingerückt)
`<link>`	Verweist auf weitere Package-Liste
`<info>`	Bietet ein Package zum Download an

Auf Pakete wird mit dem `<info>`-Tag verwiesen. Diese Verbindung erfolgt über den Namen der Info-Datei (Seite 15), aus der dann der Name des eigentlichen Paketes extrahiert werden kann.

Der Name der Info-Datei kann absolut wie auch relativ angegeben werden. Ebenfalls gültig ist folgende Angabe:

```
http://mirror.eisfair.de/eisfair/bard.tar.gz.info
```

Der Link kann also auch auf einen ganz anderen Server verweisen. Relative Verweise haben den Vorteil, dass man die Datei eis-list.txt plus Pakete auf einen lokalen Server ohne Änderung kopieren kann, um dann von dort einen *eisfair* -Rechner zu installieren.

11.1.1 index-Datei

Über die index-Datei wird bei Verwendung des `<require-package>`–Tags das zu installierende Paket ermittelt. Dazu wird zuerst geprüft, ob in der index-Datei eine Version des benötigten Pakets verzeichnet ist, die den Status „stable“ hat und deren Version mindestens die benötigte Version ist. Wird es solches Paket gefunden, so wird dieses installiert.

Sofern kein entsprechendes Paket im Status „stable“ gefunden wurde, wird die neueste „testing“ oder „unstable“ Version des Pakets installiert.

Beispiel für eine index-Datei:

```
# name      version    status     path
foo         1.2.2      stable     http://mein.server.de/foo.tar.gz.info
foo         1.3.4      testing    http://mein.server.de/dev/foo.tar.gz.info
bar         2.1.12     stable     http://mein.server.de/bar.tar.gz.info
```

In der ersten Spalte der index-Datei ist der Paketname angegeben, in den weiteren Spalten folgen die Paketversion, der Paketstatus und der 'absolute' Pfad zur Paket-Info-Datei.

[developer] http://www.pack-eis.de/?p=developer [pack-eis] http://www.pack-eis.de [libcuidoc]