Kapitel

Dieter Müller, Dr. Greg Perry, and Jonathan Potter

COLLABORATORS			
	TITLE : Kapitel		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Dieter Müller, Dr. Greg Perry, and Jonathan Potter	May 31, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Kapitel

1.1 Inhalt dieses Kapitels

1.2 ARexx

16.1 ARexx

Der ARexx-Portname von Opus 5 ist DOPUS.x, wobei x der Aufrufzähler des Programms ist (der erste und meistbenutzte dürfte DOPUS.1 sein). Da ARexx-Skripte, die von Opus aus aufgerufen werden, nicht automatisch die Befehlsadresse erben, empfiehlt sich der Einsatz der Befehlssequenz {Qp} in der aufrufenden Opus-Funktion (dies wird an anderer Stelle im Handbuch

ausführlicher beschrieben).

Wenn ein Befehl einen Wert oder eine Information zurückgibt, werden diese Daten generell in der Variable RESULT zurückgegeben. Einzige Ausnahme bildet hier die Befehle "dopus getstring" und "lister getstring (lesen Sie weiter unten mehr), die ihre Informationen in der Variablen DOPUSRC zurückgeben. Fehlercodes werden generell in der Variable RC zurückgegeben.

Der ARexx-Befehlssatz ist sehr umfassend und flexibel. Sie haben fast vollständige Kontrolle über Dateilister im Textmodus, zusammen mit der Möglichkeit Funktionen und Befehle aufzurufen. Sie können Ihre eigenen Befehle mittels ARexx zu Opus hinzufügen, die dann automatisch geladen werden - diese Befehle erscheinen, als seien sie fest in Opus eingebaut. Sie können sogar die Standardbefehle durch Ihre eigenen ersetzen.

Es existiert weiterhin eine leistungsfähige Möglichkeit zur Nutzung benutzerdefinierter Handler, die vollständig an die Bedürfnisse angepasst sind. Dies erlaubt ARexx-Programmen, Nachrichten von Opus für eine Reihe von Benutzeraktionen, einschließlich Dateilister- und Piktogrammereignissen, zu empfangen. Lesen Sie auch den Abschnitt über die

benutzerdefinierten Handler
für weitere Informationen darüber.

Sie können von hier direkt alle Abschnitte dieses Kapitels anwählen. Der mit einem (*) gekennzeichnete Abschnitt ist der, in dem Sie sich aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des Anzeigeprogramms.

1.3 ARexx-Befehle

16.2 ARexx-Befehle

Der Einfachheit halber ist der ARexx-Befehlssatz von Directory Opus hierarchisch aufgebaut mit nur drei Haupt- oder Basisbefehlen:

dopus, lister und command.

(Die in eckigen Klammern angegebenen Parameter sind optionale Parameter.)

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen.

1.4 Der Basisbefehl 'dopus'

16.2.1 Der Basisbefehl "dopus"

Der erste Basisbefehl ist dopus. Dieser Befehl ist für alle generellen
Zwecke gedacht und erlaubt die Ausführung von Funktionen, die nicht in die
anderen Kategorien fallen.

```
                dopus remappicon

                dopus remtrap

                dopus request

                dopus screen

                dopus send

                dopus setappicon

                dopus version
               Befehl: dopus addappicon
```

```
Syntax: dopus addappicon <port> <label> <id> [pos <x> <y>] [icon <filename>]
         [quotes] [info] [snap] [close] [local] [locked] [menu <stem>]
         [base <base>]
```

Dies erlaubt es Ihnen von ARexx aus Ihre eigenen AppIcons zum Bildschirm von Opus (und optional zu dem der Workbench) hinzuzufügen. Sie können das zu benutzende Piktogramm spezifizieren, den Namen des Piktogramms und dessen Position auf dem Bildschirm, sowie einige weitere Parameter. Sie können außerdem die Objekte für das Popup-Menü des neuen Piktogramms definieren. Opus sendet Nachrichten an den von Ihnen spezifizierten Messageport - sehen Sie sich dazu die mitgelieferten Beispielskripte an, die Ihnen das Empfangen und Verarbeiten dieser Nachrichten näher erklären.

Die Parameter für "addappicon" sind:

```
 port     - Der Name des Ports, an den die Nachrichten gesendet werden.
 label    - Der Name des Piktogramms (Der Text, der darunter steht).
 id       - Ihre eigene Identifikation für das Piktogramm. Diese wird in
            den Nachrichten zurückgegeben.
 pos      - Position des Piktogramms (x/y-Koordinaten)
 icon     - Optionaler Pfadname des zu nutzenden Piktogramms (ohne die
            ".info"-Ednung)
 quotes   - Geben Sie dieses Schlüsselwort an, wenn Sie wollen, daß
            Dateinamen in Anführungszeichen übergeben werden, wenn Sie an
            den Messageport gesendet werden.
 info     - Wenn Sie wollen, daß "Information" bei diesem Piktogramm
            funktioniert.
 snap     - Wenn Sie wollen, daß "Fixieren" bei diesem Piktogramm
            funktioniert.
 close    - Wenn Sie wollen, daß "Schließen" statt "Öffnen" im Popup-Menü
            stehen haben möchten.
 local    - Wenn Sie wollen, daß das Piktogramm nur auf der Opus-Oberfläche
            und nicht auf der Workbench erscheint.
 locked   - Das Piktogramm erscheint fest an seiner Position, d.h. es kann
            nicht verschoben werden.
 menu     - Name einer Stammvariable, die Ihre eigenen Menüobjekte enthält.
 base     - Erlaubt Ihnen die Spezifikation einer Basis-ID für Nachrichten,
            die vom Popup-Menü gesendet werden.
```

Der Parameter "menü" erlaubt Ihnen die Spezifikation Ihrer eigenen Menüobjekte für das Popup-Menü des Piktogramms. Die Stammvariable muß das folgende Format vorweisen:

```
  stem.COUNT - Anzahl der Objekte
  stem.0 - Objekt 1
  stem.1 - Objekt 2
  usw.
```

Wenn Sie "---" als Objekt spezifizieren, so erscheint an dieser Stelle ein Menüseparator. Wenn Sie eine Nachricht erhalten, daß der Benutzer eines dieser Menüobjekte angewählt hat, so enthält die Nachricht die ID dieses Objektes. Dies ist der Wert entsprechend der Position des Objektes in der Stammvariable (z.B. 0 für Objekt 1, 1 für Objekt 2, usw.). Wenn Sie eine Basis-ID ("base") definiert haben, so wird diese zu der eigentlichen ID addiert.

Der Befehl gibt ein AppIcon-Handle in RESULT zurück, wenn er erfolgreich ausgeführt wurde. Dieses Handle kann auch an "dopus setappicon" weitergeleitet werden, um das Piktogramm zu verändern, und es muß an "dopus remappicon" weitergeleitet werden, damit das AppIcon nach Beendigung der gewünschten Operation entfernt werden kann. Wenn Sie keine Piktogrammdatei spezifizieren, so wird das Standardpiktogramm vom Typ "Tool" (Werkzeug) benutzt.

Nachrichten vom AppIcon werden an den von Ihnen spezifizierten Messageport gesendet. Die Nachrichten sind in derselben Weise strukturiert wie die, die von Dateilistern gesendet werden, so daß kein Grund besteht, daß Sie nicht denselben Code für beide benutzen können. Für weitere Informationen sehen Sie bitte auch den Abschnitt über benutzerdefinierte Handler.

Befehl: dopus addtrap

Syntax: dopus addtrap [abort|<command>] <handler> [port <portname]

Dieser Befehl erlaubt Ihrem Skript das Abfangen des Abbruchknopfes der Fortgangsanzeige oder jedes internen Opus-Befehls. Spezifizieren Sie das Schlüsselwort "abort", um die Abbruchnachricht abzufangen, oder den Namen des internen Befehls, der abgefangen werden soll. <handler> ist der Name des Messageports Ihres benutzerdefinierten Handlers. Wenn Sie mit dem Schlüsselwort "port" den Namen eines Messageports übergeben, so wird die Nachricht zu diesem Port anstatt zum üblichen Handlerport geschickt. Dies kann sehr nützlich sein, wenn es im Zusammenhang mit "abort" genutzt wird, und Ihr Handler gerade "busy" ist, weil er etwas synchron ausführt, während auf Abbruch gedrückt wird. Wenn der Port für einen Abbruch ein separater Prozess ist, kann er den den Haupthandlerprozess z.B. mittels Signalen unterbrechen. Der Befehl "dopus remtrap" dient dazu, abgefangene Funktionen wieder zu entfernen. Lesen Sie Weiteres in der Sektion über die

```
                benutzerdefinierten Handler
                 zu den Messageports.
```

Befehl: dopus back

Syntax: dopus back

Hauptfenster (und Bildschirm) von Opus 5 werden ganz nach hinten gelegt.

Befehl: dopus command

Syntax: dopus command <name> program <scriptname> [desc <description>]
[template <template>] [source] [dest]

Dieser Befehl bietet die Möglichkeit, neue interne Befehle zu Opus hinzuzufügen oder bereits existierende zu ersetzen. Er wird generell aus der "init"-Funktion eines Opus-ARexx-Moduls aufgerufen und der Parameter "program" ist der Name dieses Moduls. Das Feld "program" ist unabdingbar und Opus wird den Skriptnamen, den Sie hier angeben, jedesmal ausführen, wann immer diese Funktion aufgerufen wird. Lesen Sie mehr darüber bei den

ARexx-Modulen
und den Beispielskripten.

Befehl: dopus error

Syntax: dopus error <code>

Dieser Befehl wird genutzt, um aussagekräftigere Fehlermeldungen zu erhalten, wenn ihm ein Opus-ARexx-Fehlercode übergeben wird.

Beispiel:

```
> dopus error 1
                ---> Datei vom Filter abgelehnt
> dopus error 10
                ---> Ungültiges Listerhandle
```

Befehl: dopus front

Syntax: dopus front

Damit werden Hauptfenster (und Bildschirm) von Directory Opus 5 ganz nach vorne geholt.

Befehl: dopus getfiletype

Syntax: dopus getfiletype <filename> <id>

Dieser Befehl untersucht den Dateityp der angegebenen Datei. <filename> ist der volle Name der Datei einschließlich Pfad. Standardmäßig wird bei Erkennung des Dateityps die Dateitypbeschreibung als String in RESULT zurückgegeben. Geben Sie das Schlüsselwort "id" an, wird stattdessen die Identifikation des Dateityps zurückgegeben.

Beispiel:

```
> dopus getfiletype RAM:Testdatei.lha
                                    ---> LHA Archiv
> dopus getfiletype RAM:Bild.jpg id
                                    ---> JPEG
```

Befehl: dopus getstring

Syntax: dopus getstring <text> [secure] [<length>] [<default>] [<buttons>]

Dieser Befehl erlaubt dem Benutzer die Übergabe eines Textstrings. <text> ist ein String der im Requester als Text angezeigt wird und sollte, wenn Leerzeichen in ihm enthalten sind, in Anführungszeichen eingeschlossen sein. Das Schlüsselwort "secure" sorgt dafür, daß jeder eingegebene Buchstabe nur durch einen Stern (*) dargestellt wird, so daß sich hiermit auch Passwortabfragen realisieren lassen. <length> ist die maximal akzeptierte Eingabelänge und ist, wenn nicht spezifiziert, auf 80 Zeichen voreingestellt. <default> ist der Vorgabewert, der im Requester als Vorgabetext erscheint. <buttons> sind die Knöpfe, die Sie in dem Requester zur Verfügung haben wollen; die Knöpfe sollten dabei durch vertikale Striche ("|") abgetrennt sein. Wird der Parameter "buttons" weggelassen, so hat der Requester nur einen Knopf names "OK".

Beispiel:

```
 > dopus getstring '"Bitte Text eingeben" 40 "" OK|Abbruch'
```

Dieses Beispiel würde einen Requester mit dem Text "Bitte Text eingeben" anzeigen, eine maximale Eingabelänge von 40 Zeichen akzeptieren, keinen Vorgabetext zeigen und zwei Knöpfe, die mit "OK" und "Abbruch" beschriftet sind, anbieten.

Der Eingabestring wird, falls eingegeben, in RESULT zurückgeliefert. Die Grundzahl des gewählten Knopfes wird in der speziellen Variable DOPUSRC zurückgegeben. Im obigen Beispiel würde, wenn der Benutzer auf "OK" klickte, DOPUSRC den Wert 1 enthalten. Hätte er auf "Abbruch" geklickt, so würde 0 zurückgegeben. Nur dieser Befehl und der Befehl "lister getstring" nutzen momentan die Variable DOPUSRC. Dies kann sich in der Zukunft mit weiteren Entwicklungen natürlich noch ändern.

Bitte beachten Sie, daß in früheren Version von Opus 5 die Variable RESULT nicht gelöscht wurde, wenn ein leerer String eingegeben wurde. Achten Sie darauf, ob von diese Änderung Ihre eigenen Skripte betroffen sind oder nicht.
Befehl: dopus read

```
Syntax: dopus read <filename>
         dopus read <handle> <filename>
         dopus read <handle> <quit>
```

Diese Befehle wurden implementiert, um Ihnen größere Kontrolle über den Textanzeiger von Opus zu geben. Grundsätzlich erlauben Sie Ihnen, einen Textanzeiger zu öffnen und darin einen nach dem anderen Texte zu betrachten, bevor er wieder geschlossen wird. Sie können das Textanzeigerhandle in derselben Weise nutzen wie das Dateilisterhandle.

Beispiel:

```
 > dopus read RAM:Datei1.TXT
                                    ---> 121839492
 > dopus read 121839492 RAM:Datei2.TXT
```

```
                                    ---> 121839492
 > dopus read 121839492 quit
                                    ---> 0
```

Befehl: dopus remappicon

Syntax: dopus remappicon <handle>

Hiermit entfernen Sie ein AppIcon, das Sie vorher mit dem Befehl "dopus addappicon" hinzugefügt haben.

Befehl: dopus remtrap

Syntax: dopus remtrap [abort|<command>] <handler>

Schaltet das Abfangen des Abbruchknopfes der Fortgangsanzeige oder eines internen Befehls, wie es mit "dopus addtrap" initiiert wurde, wieder aus. Wenn Sie für <command> ein "*" eingeben, so werden alle Traps, die für diesen Handler definiert wurden, entfernt. <handler> ist der Name des Messageports, wie er bei "dopus addtrap" spezifiziert wurde.

Befehl: dopus request

Syntax: dopus request <text> <buttons>

Dieser Befehl verlangt vom Benutzer eine Auswahl. <text> ist der angezeigte Textstring und <buttons> sind die Knöpfe, die Sie im Requester erscheinen lassen wollen. Knöpfe werden voneinander auch hier durch einen vertikalen Strich abgetrennt.

Beispiel:

```
 > dopus request '"Bitte wählen Sie" "Option1|Option2|Option3"'
```

Dies würde einen Requester mit dem Text "Bitte wählen Sie" ausgeben und Ihnen drei Knöpfe zur Auswahl bieten, die mit "Option1", "Option2" und "Option3" beschriftet sind.

Der Wert des angewählten Knopfes wird in RC zurückgegeben. Der letzte Knopf (in diesem Fall "Option3") ist immer als "Abbruch"-Knopf gedacht und bekommt daher automatisch den Wert 0 zugewiesen. Die bei diesem Beispiel zurückgegebenen Werte sind also 1, 2, oder 0.

Befehl: dopus screen

Syntax: dopus screen

Dieser Befehl gibt Informationen über den Bildschirm von Opus im folgenden Format zurück:

```
 <name> <width> <height> <depth> <barheight>
```

Wenn Opus verborgen ist, hat es natürlich keinen aktuellen Bildschirm. In

diesem Fall wird in RC der Wert 5 zurückgegeben. Sie können so auch herausfinden, ob Opus verborgen ist oder nicht.

Beispiel:

```
 > dopus screen
                 ---> DOPUS.1 640 512 2 10
```

Der Wert <barheight> ist nützlich, wenn Sie Dateilister oder andere Fenster direkt unterhalb der Titelzeile des Bildschirms öffnen wollen.

Befehl: dopus send

Syntax: dopus send <port name> <string>

Dieser Befehl richtet bei Opus selbst nichts aus, erleichtert es Ihnen aber, z.B. einen String als Nachricht an einen anderen ARexx-Task zu senden. Der String wird in Arg0 der Nachricht übergeben, die an den genannten Port gesendet wird.

Befehl: dopus setappicon

Syntax: dopus setappicon <handle> <item>

Dies erlaubt Ihnen die Manipulation von AppIcons, die Sie mit dem Befehl "dopus addappicon" hinzugefügt haben. Gültige Werte für <item> sind:

 text <text>

  Ändert die Beschriftung des Piktogramms.

 busy [on|off]

  Ändert den Piktogrammstatus auf "busy" oder nicht-"busy".

 locked [on|off]

  Sperrt oder entsperrt die Position des Piktogramms.

Wenn ein Piktogramm "busy" ist, kann es durch den Benutzer nicht angewählt werden. Es wird dann nicht reagieren auf Doppelklicks, Popup-Menü-Ereignisse, Nehmen & Ablegen usw. Das Piktogramm wird, wenn es "busy" ist, als Geisterpiktogramm dargestellt, d.h. es wird mit einem Punktraster belegt, um diesen Status zu kennzeichnen.

Wenn ein Piktogramm gesperrt ist, kann seine Position nicht verändert werden und es kann nicht manuell durch den Benutzer verschoben werden, genausowenig, wie ein "CleanUp" einen Effekt darauf hätte.

Befehl: dopus version

Syntax: dopus version

Dieser Befehl gibt einen String im Format <version> <revision> zurück und kann in ARexx-Skripten nützlich sein, um festzustellen, ob bestimmte Fähigkeiten überhaupt verfügbar sind.

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen. Der mit einem (*) gekennzeichnete Unterpunkt ist der, in dem Sie sich aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des Anzeigeprogramms.

1.5 Der Basisbefehl 'lister'

16.2.2 Der Basisbefehl "lister"

Der nächste Basisbefehl, lister, erlaubt Ihnen die Kontrolle über Dateilister und die Einträge in Dateilistern.

lister add

lister addstem

lister copy

lister clear

lister clear

lister close

lister empty

lister getstring

lister new

lister query

lister read

lister refresh

lister remove

lister request

```
                lister set

                lister select

                lister wait

                lister iconify
               Befehl: lister add

Syntax: lister add <handle> <name> <size> <type> <seconds> <protect>
         <comment>

Dieser Befehl fügt dem spezifizierten Dateilister einen Eintrag hinzu.

 <name> ist der volle Name des Eintrags.
 <size> ist die Größe dieses Eintrags.
 <type> definiert den Typ des Eintrags (<0 für eine Datei und >0 für ein
        Verzeichnis).
 <seconds> ist das Datum des Eintrags in Sekunden seit dem 1.1.1978.
 <protect> sind die Schutzbits der Datei im ASCII-Format
 <comment> ist der Dateikommentar des Eintrags (wenn gewünscht)

Gültige Werte für <type> sind:

  0 - Gerät
  1 - Verzeichnis
 -1 - Datei
  2 - Verzeichnis in der Farbe von Zuweisungen
 -2 - Datei in der Farbe von Geräten
  3 - Verzeichnis in Fettdruck (wie Links)
 -3 - Datei in Fettdruck (wie Links)
  4 - Verzeichnis in der Farbe von Zuweisungen und Fettdruck
 -4 - Datei in der Farbe von Geräten und Fettdruck

Nach diesem Befehl wird die Anzeige des Dateilisters nicht automatisch
aufgefrischt. Dies geschieht erst nach Ausführung eines "lister refresh".
Beispiel:

 > lister add 121132636 "Meine Datei!" 12839 -1 540093905 prwed Mein Kommentar


Befehl: lister addstem

Syntax: lister addstem <handle> <stem>

Dieser Befehl fügt Dateien zu einen Dateilister mittels einer
Stammvariablen hinzu. Dieser Befehl ist leistungsfähiger als "lister add"
und sollte daher bevorzugt genutzt werden. Die Felder der Stammvariable
sind denen sehr ähnlich, die vom "lister query <handle> entry" Stammbefehl
zurückgegeben werden (Tatsächlich können Sie die Resultate von "query"
direkt an "addstem" übergeben, um einen identischen Eintrag in einem
anderen Lister zu erzeugen).

Die benutzten Felder sind:

 name       - Name des Eintrags
```

```
 size       - Dateigröße
 type       - Art des Eintrags (identisch mit "lister add")
 protstring - Schutzbits im ASCII-Format (z.B. "rwed")
 protect    - Schutzwert (Zahl, die benutzt wird, wenn "protstring" nicht
              angegeben wird)
 comment    - Dateikommentar
 datestring - Erzeugungsdatum und -zeit (ASCII-String)
 date       - Anzahl der Sekunden seit 1.1.1978 (wird benutzt, wenn
              "datestring" nicht angegeben wird)
 filetype   - ASCII-String für die Dateitypanzeige
 selected   - 0 oder 1
 version    - Versionsnummer
 revision   - Revisionsnummer
 verdate    - Datumsstring der Version
 userdata   - Benutzerdaten (Wert, kein String)
 display    - Benutzerdefinierter Anzeigestring
 menu       - Benutzerdefiniertes Popup-Menü
 base       - Basis-ID für das Popup-Menü
```

Nicht alle dieser Felder werden auch benötigt. Als absolutes Minimum sollten Sie die Felder "name" oder "display" spezifizieren.

Das Feld "display" erlaubt es Ihnen, einen komplett benutzerdefinierten String zu wählen, der für den Eintrag angezeigt wird. Es wird keine der anderen Informationen angezeigt, wenn dieser String angegeben wird. Die maximale Länge beträgt 256 Zeichen.

Das Feld "userdata" erlaubt es Ihnen, Ihren eigenen Wert für eine Benutzer-ID (oder einen anderen Wert) zu spezifizieren, der mit diesem Eintrag assoziiert wird. Der Hauptnutzen ergibt sich im Zusammenhang mit dem benutzerdefinierten Popup-Menü und den benutzerdefinierten Handlern.

Das Feld "menu" erlaubt es Ihnen, eine Stammvariable zu spezifizieren, die benutzerdefinierte Objekte für das Popup-Menü enthält, das dargestellt wird, wenn der Benutzer die rechte Maustaste über dem Eintrag drückt. Das Format entspricht dem für die Popup-Menüs bei AppIcons:

```
  stem.COUNT - Anzahl der Einträge
  stem.BASE - Basis-ID
  stem.0 - Eintrag 1
  stem.1 - Eintrag 2
  usw.
```

Wenn "count" auf 0 gesetzt wird, so ist das Popup-Menü für die rechte Maustaste für diesen Eintrag abgeschaltet. Ist dieses Feld nicht angegeben, so wird das Standard-Popup-Menü angezeigt. Wenn Sie "---" als Eintrag spezifizieren, so erscheint an dieser Stelle ein Menüseparator.
Wenn Sie eine Nachricht erhalten, daß der Benutzer einen dieser Menüeinträge angewählt hat, so enthält die Nachricht die ID dieses Eintrags. Dies ist der Wert entsprechend der Position des Eintrags in der Stammvariable (z.B. 0 für Eintrag 1, 1 für Eintrag 2, usw.). Wenn Sie eine Basis-ID ("base") definiert haben, so wird diese zu der eigentlichen ID addiert.

Lesen Sie dazu auch den Abschnitt über
benutzerdefinierte Handler
später in
diesem Kapitel, um weitere Informationen über Messageports zu erhalten.

Befehl: lister copy

Syntax: lister copy <handle> <destination>

Dieser Befehl kopiert den Inhalt des einen Listers in einen anderen. Im Gegensatz zu den meisten anderen Befehlen wird bei diesem die Anzeige des Zieldateilisters umgehend aufgefrischt.

Beispiel:

> lister copy 121132636 121963868

Befehl: lister clear

Syntax: lister clear <handle>

Dieser Befehl löscht den Inhalt des spezifizierten Listers. Die Anzeige wird erst nach Ausführen eines "lister refresh" erneuert.

In früheren Versionen von Opus wurde so auch der Name des benutzerdefinierten Handlers gelöscht. Dies geschieht nun nicht mehr!
Befehl: lister clear

Syntax: lister clear <handle> <item> <value>

Dieser Befehl löscht eine bestimmte Information aus dem spezifizierten Lister. <handle> ist das Handle das fraglichen Listers. <item> kann eines der folgenden Schlüsselworte sein.

abort

Dies löscht das "Abbruch"-Flag im spezifizierten Dateilister.

> lister clear 121132636 abort

flags <flags>

Löscht die Sortier- und Anzeigeflags des spezifizierten Listers. Die Anzeige wird nicht erneuert vor dem Ausführen eines "lister refresh". Schauen Sie bei
lister query
für die Liste der verfügbaren
Schlüsselworte nach.

Beispiel:

> lister clear 121132636 flags reverse

progress

Schaltet den Fortgangsindikator im spezifizierten Dateilister ab.

```
  > lister clear 121132636 progress
```

Befehl: lister close

Syntax: lister close [all|<handle>]

Dieser Befehl schließt den spezifizierten Dateilister, oder, wenn das Schlüsselwort "all" angegeben wird, alle Dateilister. Alle Aktionen, die gerade in diesen Dateilistern stattfinden, werden abgebrochen. Der Wert <handle> ist das Handle, das zurückgegeben wurde, als dieser Dateilister mit dem Befehl "lister new" geöffnet wurde.

Beispiel:

```
 > lister close 121132636
```

Befehl: lister empty

Syntax: lister empty <handle>

Dieser Befehl zeigt im spezifizierten Lister einen leeren Puffer an (ähnlich "lister clear", der den Inhalt des aktuellen Puffers löscht). Wenn keine leeren Puffer verfügbar sind (und kein neuer erzeugt werden kann), wird der existierende Puffer gelöscht. Ist ein benutzerdefinierte Handler hier zugehörig, wird dieser eine "inactive" Nachricht erhalten.

Beachten sie bitte, daß die Nachricht "inactive" von früheren Versionen von Opus 5 nicht an den benutzerdefinierten Handler gesendet wurden, wenn dieser Befehl ausgeführt wurde.
Befehl: lister getstring

Syntax: lister getstring <handle> <text> [secure] [<length>] [<default>]
         [<buttons>]

Dieser Befehl ist identisch mit dem Befehl "dopus getstring", außer daß hier noch ein Listerhandle als zusätzlicher Parameter übergeben wird und der Requester über dem Dateilister zentriert wird.

<text> ist ein String der im Requester als Text angezeigt wird und sollte, wenn Leerzeichen in ihm enthalten sind, in Anführungszeichen eingeschlossen sein. Das Schlüsselwort "secure" sorgt dafür, daß jeder eingegebene Buchstabe nur durch einen Stern (*) dargestellt wird, so daß sich hiermit auch Passwortabfragen realisieren lassen. <length> ist die maximal akzeptierte Eingabelänge und ist, wenn nicht spezifiziert, auf 80 Zeichen voreingestellt. <default> ist der Vorgabewert, der im Requester als Vorgabetext erscheint. <buttons> sind die Knöpfe, die Sie in dem Requester zur Verfügung haben wollen; die Knöpfe sollten dabei durch vertikale Striche ("|") abgetrennt sein. Wird der Parameter "buttons" weggelassen, so hat der Requester nur einen Knopf names "OK".

Beispiel:

```
 > lister getstring 121132636 '"Bitte Text eingeben" 40 "" OK|Abbruch'
```

Dieses Beispiel würde einen Requester mit dem Text "Bitte Text eingeben" anzeigen, eine maximale Eingabelänge von 40 Zeichen akzeptieren, keinen Vorgabetext zeigen und zwei Knöpfe, die mit "OK" und "Abbruch" beschriftet sind, anbieten.

Der Eingabestring wird, falls eingegeben, in RESULT zurückgeliefert. Die Grundzahl des gewählten Knopfes wird in der speziellen Variable DOPUSRC zurückgegeben. Im obigen Beispiel würde, wenn der Benutzer auf "OK" klickte, DOPUSRC den Wert 1 enthalten. Hätte er auf "Abbruch" geklickt, so würde 0 zurückgegeben. Nur dieser Befehl und der Befehl "dopus getstring" nutzen momentan die Variable DOPUSRC. Dies kann sich in der Zukunft mit weiteren Entwicklungen natürlich noch ändern.

Bitte beachten Sie, daß in früheren Version von Opus 5 die Variable RESULT nicht gelöscht wurde, wenn ein leerer String eingegeben wurde. Achten Sie darauf, ob von diese Änderung Ihre eigenen Skripte betroffen sind oder nicht.
Befehl: lister new

Syntax: lister new [<x/y/w/h>] [toolbar <toolbar>] [<path>]

Dieser Befehl erzeugt einen neuen Dateilister. Sie können optional die Position und die Größe des Dateilisters angeben. Voreingestellt ist -1/-1, so daß sich der Dateilister an der Position des Mauszeigers öffnet. Mit dem Schlüsselwort "toolbar" kann auch eine benutzerdefinierte Werkzeugleiste spezifiziert werden. Werkzeugleistendateien werden im Verzeichnis "DOpus5:Buttons" erwartet, wenn kein vollständiger Pfad dafür angegeben wird. Außerdem können Sie noch einen Startpfad definieren, der eingelesen werden soll, wenn der Dateilister geöffnet wird.

Beispiel:

```
 > lister new
 > lister new 100/50/400/300
 > lister new ram:
 > lister new 80/30/200/200 DH0:Work
 > lister new toolbar Custom_Toolbar Work:
                                          ---> 121132636
```

Wenn ein Dateilister erfolgreich geöffnet werden konnte, wird sein Handle in der Variable RESULT zurückgegeben. Sie müssen dieses Handle merken, wenn Sie später mit dem dazugehörigen Dateilister noch irgendetwas anfangen wollen. Im obigen Beispiel wurde das Handle "121132636" zurückgegeben. Dieses wird auch in weiteren Beispielen benutzt.

Befehl: lister query

Syntax: lister query <handle> <item>

Dieser Befehl gibt eine bestimmte Information eines spezifizierten Dateilisters zurück. <handle> ist das Handle des fraglichen Listers. Alle Informationen, die abgefragt werden, werden in der Variable RESULT zurückgegeben, es sei denn, ein Fehler tritt auf.
<item> kann eines der folgenden Schlüsselworte sein:

```
 all

Dies gibt die Handles aller Dateilister zurück, die momentan nicht "BUSY"
sind (dies sind alle, die gerade keine Operation ausführen). Hierfür wird
kein Handle in der Befehlsübergabe benötigt.

Beispiel:

 > lister query all
                    ---> 121132636 121963868


 abort

Dies gibt einen boolschen Wert zurück, der den Status des Abbruchflags des
Listers angibt. Dieser Befehl hat nur Gültigkeit, wenn der Dateilister
einen Fortgangsindikator (wie z.B. beim Kopieren) geöffnet hat, da dies der
einzige Moment ist, in dem der Benutzer eine Funktion abbrechen kann.
Es wird "1" zurückgeliefert, wenn "Abbruch" gewählt wurde und "0", wenn
nicht.

Beispiel:

 > lister query 121132636 abort
                                  ---> 0

Eine Abfrage des Abbruchflags setzte dieses in Opus 4 zurück. In Opus 5
ist dies nicht mehr der Fall. Wollen Sie dieses Flag zurücksetzen, müssen
Sie dazu "lister clear" anwenden.
 busy

Liefert einen boolschen Wert (0 oder 1) zurück, der anzeigt, ob der
spezifizierte Dateilister "BUSY" ist. Ist er "BUSY", wird eine "1"
zurückgegeben, ist er es nicht, eine "0".

Beispiel:

 > lister query 121132636 busy
                                ---> 1


 dest

Dies gibt die Handles aller momentan geöffneten Zieldateilister zurück. Da
Sie sich hier nicht direkt auf einen speziellen Lister beziehen, müssen Sie
kein Handle in der Befehlsübergabe angeben. Hierbei können Sie außerdem
Stammvariablen nutzen.

Beispiel:

 > lister query dest
                     ---> 121963868

Dieser Befehl unterstützt die Schlüsselworte "stem" und "var".
(Weitere Informationen dazu finden Sie unter "lister query entry").
```

```
 dirs <separator>
```

Gibt die Namen aller Verzeichnisse im spezifizierten Dateilister zurück.

Beispiel:

```
 > lister query 121132636 dirs ,
                                  ---> "Clipboards","ENV","T"
```

Dieser Befehl unterstützt die Schlüsselworte "stem" und "var".
(Weitere Informationen dazu finden Sie unter "lister query entry").

```
 display
```

Dies gibt einen String zurück, der alle Datenfelder angibt, die momentan vom Dateilister angezeigt werden. Der String besteht aus denselben Schlüsselwörtern, wie bei "sort". Die Reihenfolge ist die, in der sie im Dateilister angezeigt werden (sofern sie angezeigt werden).

Beispiel:

```
 > lister query 121132636 display
                                    ---> name size date protect comment
```

```
 entries <separator>
```

Gibt die Namen aller Einträge (Dateien und Verzeichnisse) des Dateilisters zurück

Beispiel:

```
 > lister query 121132636 entries
            ---> "Clipboards" "ENV" "T" "abc" "Disk.info" "readme" "zzz.zzz"
```

Dieser Befehl unterstützt die Schlüsselworte "stem" und "var".
(Weitere Informationen dazu finden Sie unter "lister query entry").

Beispiel:

```
 > lister query 121132636 entries stem files
```

Dies würde die folgenden Variablen zurückliefern:

```
  files.count = 7
  files.0 = Clipboards
  files.1 = ENV
  files.2 = T
  files.3 = abc
  usw.
```

```
 entry <name>
```

Gibt Informationen über den spezifizierten Eintrag zurück. <name> ist der

```
aktuelle Name des Eintrages, über den Sie Informationen wollen. Sie können
für <name> auch #xxx angeben (xxx ist eine Zahl), um die Ordnungszahl des
gewünschten Eintrages zu spezifizieren. Dieser Befehl kann Informationen
auf zwei Arten zurückliefern. Standardmäßig wird ein Informationsstring in
der Variable RESULT oder einer Variable Ihrer Wahl zurückgeliefert. Die in
diesem Fall zurückgelieferte Information ist:

 <name> <size> <type> <selection> <seconds> <protect> <comment>

 <name>: Der volle Name des Eintrages
 <size>: Die Größe dieses Eintrages
 <type>: Der Typ des Eintrag (<0 für eine Datei, >0 für ein Verzeichnis)
 <selection>: Der Anwahlstatus des Eintrags (1 wenn angewählt, 0 wenn nicht)
 <seconds>: Das Datum des Eintrags in Sekunden seit dem 01.01.1978
 <protect>: Die Schutzbits (im ASCII-Format)
 <comment>: Der Dateikommentar (wenn vorhanden)

Beispiel:

 > lister query 121132636 entry ENV
                                     ---> ENV -1 2 0 543401724 ----rwed

Standardmäßig wird das Ergebnis in der Variable RESULT zurückgegeben.
Wollen Sie eine andere Variable dafür nutzen, so können Sie dies mit dem
Schlüsselwort "var", gefolgt von einem Variablennamen, tun.

Beispiel:

 > lister query 121132636 entry ENV var my_variable

Die zweite und elegantere Methode gibt Informationen über den Eintrag in
einer Stammvariablen zurück. Um diese zweite Methode zu nutzen, müssen Sie
das Schlüsselwort "stem", gefolgt von dem gewünschten Namen der
Stammvariable angeben, angeben.

Beispiel:

 > lister query 121132636 entry ENV stem fileinfo

Die spezifizierte Stammvariable hätte mehrere Felder, wobei jedes Feld
Informationen über den abgefragten Eintrag enthält. Die dabei benutzten
Felder sind:

 name       - Dateiname des Eintrags
 size       - Dateigröße
 type       - Art des Eintrags (<0 = Datei, >0 = Verzeichnis)
 selected   - 0 oder 1
 date       - Anzahl der Sekunden seit 1.1.1978
 protect    - Schutzbits (Langwert)
 datestring - Erzeugungsdatum und -zeit (ASCII-Format)
 protstring - Schutzbits im ASCII-Format (z.B. "rwed")
 comment    - Dateikommentar (falls vorhanden)
 filetype   - Dateityp (falls vorhanden)
 version    - Versionsnummer
 revision   - Revisionsnummer
 verdate    - Datumsstring der Version (numerisch TT.MM.JJ)
 datenum    - Dateidatum im numerischen Format TT.MM.JJ
```

```
 time       - Dateizeit im 24-Stunden-Format HH:MM:SS
```

Zahlreiche andere "query"-Befehle in dieser Sektion unterstützen die Schlüsselworte "var" und "stem".

Beachten Sie bitte etwas bei der Nutzung folgender Befehle: lister query files, dirs, entries, selfiles, seldirs, selentries. In früheren Versionen von Opus 5 wurde die Variable RESULT nicht geleert, wenn keine Werte zurückgegeben werden konnten. Dies Problem wurde beseitigt. Wenn Sie diese Befehl mit dem Schlüsselwort "stem" benutzen, wird das Feld "count" nun in einem solchen Fall auch auf Null gesetzt.

```
 files <separator>
```

Gibt die Namen aller Dateien im spezifizierten Dateilister zurück. Die Namen werden als ein langer String zurückgegeben, durch Leerschritte getrennt. Sie können statt der Leerschritte aber auch einen anderen Separator wählen, wenn Sie das Schlüsselwort dafür benutzen.

Beispiel:

```
 > lister query 121132636 files
                                  ---> "abc" "Disk.info" "readme" "zzz.zzz"
```

Dieser Befehl unterstützt die Schlüsselworte "stem" und "var".
(Weitere Informationen dazu finden Sie unter "lister query entry").

```
 firstsel
```

Gibt den Namen des ersten angewählten Eintrags des Dateilisters zurück. Dieser Eintrag wird nicht deselektiert, so daß dieser Befehl solange den gleichen Namen zurückgibt, bis Sie selbst den Eintrag deselektieren.

Beispiel:

```
 > lister query 121132636 firstsel
                                     ---> "ENV"
```

```
 flags
```

Dies gibt einen String zurück, der die aktiven Anzeigeflags des Dateilisters enthält. Diese Flags sind:

```
 reverse   - Sortieren in umgekehrter Reihenfolge
 noicons   - Piktogramm filtern
 hidden    - Verborgene Dateien anzeigen
```

Beispiel:

```
 > lister query 121132636 flags
                                  ---> noicons
```

```
 hide
```

Dies liefert den Filter für das Verbergen von Einträgen für diesen Lister

zurück.

Beispiel:

```
 > lister query 121132636 hide
                                ---> #?.o
```

 handler

Dies gibt den Namen des aktuellen, benutzerdefinierten Handlerports zurück.

Beispiel:

```
 > lister query 121132636 handler
                                  ---> ArcDir121132636
```

 label

Dies gibt den Textstring zurück, der unterhalb der Piktogramms erscheint, wenn dieser Dateilister verborgen ist. Standardmäßig ist dies der Name des des Verzeichnisses, das der Dateilister gerade anzeigt. Dieser Name kann allerdings auch verändert werden durch Aufruf des Befehls "lister set label".

Beispiel:

```
 > lister query 121132636 label
                                ---> Ram Disk
```

 lock <type>

Dieser Befehl gibt den Sperrstatus des Dateilisters zurück, wobei <type> entweder "state" oder "format" ist. Sehen Sie dazu den Befehl

```
lister set lock
.
```

 mode

Dies gibt den aktuellen Darstellungsmodus des Dateilisters zurück, ebenso wie das Schlüsselwort "showall", wenn der Dateilister in einem der Piktogrammodi ist und auch Dateien ohne Piktogramme anzeigt. Die Modi des Dateilisters sind:

```
 name        - Textmodus
 icon        - Piktogrammodus
 icon action - Piktogramm Plus-Modus
```

Beispiel:

```
 > lister query 121132636 mode
                               ---> icon action showall
```

numdirs

Gibt die Anzahl der Verzeichnisse im Dateilister zurück.

Beispiel:

```
> lister query 121132636 numdirs
                                  ---> 3
```

numentries

Gibt die Anzahl aller Einträge (Dateien und Verzeichnisse) des Dateilisters zurück.

Beispiel:

```
> lister query 121132636 numentries
                                    ---> 7
```

numfiles

Gibt die Anzahl der Dateien im Dateilister zurück.

Beispiel:

```
> lister query 121132636 numfiles
                                   ---> 4
```

numselentries

Gibt die Anzahl der angewählten Einträge (Dateien und Verzeichnisse) des Dateilisters zurück.

numseldirs

Gibt die Anzahl der angewählten Verzeichnisse des Dateilisters zurück.

numselfiles

Gibt die Anzahl der angewählten Dateien des Dateilisters zurück.

path

Liefert einen String zurück, der den momentanen Pfad, der im Lister angezeigt wird, enthält.

Beispiel:

```
> lister query 121132636 path
                                ---> ram:
```

```
 position

Liefert die aktuelle Position und die Größe des Dateilisters zurück.
Außerdem wird das Schlüsselwort "locked" zurückgegeben, wenn die Position
des Dateilisters gesperrt ist.

Beispiel:

 > lister query 121132636 position
                                  ---> 80/30/200/200 locked


 selfiles <separator>

Gibt die Namen aller angewählten Dateien des Dateilisters zurück.
Dieser Befehl unterstützt die Schlüsselworte "stem" und "var".
(Weitere Informationen dazu finden Sie unter "lister query entry").


 seldirs <separator>

Gibt die Namen aller angewählten Verzeichnisse des Dateilisters zurück.
Dieser Befehl unterstützt die Schlüsselworte "stem" und "var".
(Weitere Informationen dazu finden Sie unter "lister query entry").


 selentries <separator>

Gibt die Namen aller angewählten Einträge (d.h. Dateien und Verzeichnisse)
des Dateilisters zurück. Dieser Befehl unterstützt die Schlüsselworte
"stem" und "var". (Weitere Informationen dazu finden Sie unter "lister
query entry").


 separate

Dies gibt ein Schlüsselwort zurück, das Ihnen die Eintragsordnung des
Dateilisters mitteilt. Gültige Eintragsordnungen sind:

 mix        - Dateien und Verzeichnisse gemischt
 dirsfirst  - Verzeichnisse zuerst
 filesfirst - Dateien zuerst

Beispiel:

 > lister query 121132636 separate
                                  ---> dirsfirst


 show

Dies liefert den Filter für die Anzeige von Einträgen zurück.


 source
```

Dies gibt die Handles aller momentan geöffneten Quelldateilister zurück. Da Sie sich hier nicht direkt auf einen speziellen Lister beziehen, müssen Sie kein Handle im Befehl angeben.

Beispiel:

```
 > lister query source
                        ---> 121132636 128765412
```

Dieser Befehl unterstützt die Schlüsselworte "stem" und "var".
(Weitere Informationen dazu finden Sie unter "lister query entry").

Beispiel:

```
 > lister query source stem sources
```

Dies würde zurückgeben:

```
  sources.count = 2
  sources.0 = 121132636
  sources.1 = 128765412
```

sort

Dies gibt ein Schlüsselwort zurück, das die Sortiermethode des Dateilisters angibt. Die möglichen Schlüsselworte sind:

```
 name      - Dateiname
 size      - Dateigröße
 protect   - Schutzbits
 date      - Datum
 comment   - Kommentar
 filetype  - Dateityp
 version   - Dateiversion
```

Beispiel:

```
 > lister query 121132636 sort
                                  ---> name
```

toolbar

Dies gibt zurück, welche Werkzeugleiste momentan vom spezifizierten Dateilister benutzt wird.

Beispiel:

```
 > lister query 121132636 toolbar
                                     ---> DOpus5:Buttons/toolbar
```

visible

Liefert einen boolschen Wert, der anzeigt, ob der spezifizierte Dateilister

momentan sichtbar ist.

Beispiel:

```
> lister query 121132636 visible
                                  ---> 1
```

Befehl: lister read

Syntax: lister read <handle> <path> [force]

Dieser Befehl liest den angegebenen Pfad in den spezifizierten Lister. Standardmäßig wird ein neuer Puffer zum Lesen des Verzeichnisses verwendet; wird jedoch das Schlüsselwort "force" verwendet, wird der aktuelle Puffer gelöscht und das Verzeichnis wird in diesen eingelesen. Der alte Pfad wird in RESULT zurückgegeben.

Beispiel:

```
 > lister read 121132636 'dh0:test'
                                       ---> RamDisk:
```

Befehl: lister refresh

Syntax: lister refresh (all|<handle>) [full] [date]

Dieser Befehl erneuert die Anzeige des spezifizierten Dateilisters, oder aller Dateilister, wenn das Schlüsselwort "all" anstelle eines Listerhandles angegeben wird. Anders als in Opus 4 erneuert keiner der bisher genannten Befehle die Anzeige des spezifizierten Dateilisters. Wenn Sie also wollen, daß Veränderungen sichtbar werden, dann müssen Sie diesen Befehl ausführen (z.B. nach dem Hinzufügen von Dateien, Ändern der Sortiermethode usw.), um die Anzeige zu erneuern. Das optionale Schlüsselwort "full" sorgt dafür, daß die Titelzeile und die Statusanzeige ebenfalls erneuert werden.

Beispiel:

```
 > lister refresh 121132636 full
```

Wird das Schlüsselwort "date" mitangegeben, so wird der Dateilister das Datum des Verzeichnisses erneuern, was dafür sorgt, daß bei der nächsten Aktivierung des Dateilisters das Verzeichnis nicht erneut eingelesen wird. Bei Angabe dieses Schlüsselwortes wird die eigentliche Eintragsanzeige des Dateilisters nicht aufgefrischt.

Beispiel:

```
 > lister refresh 121132636 date
```

Befehl: lister remove

Syntax: lister remove <handle> <name>

Dieser Befehl entfernt einen Eintrag aus dem spezifizierten Lister. <name> ist entweder der Name des Eintrags oder #xxx (xxx ist eine Zahl) zur Festlegung der Nummer des Eintrags. Die Anzeige wird erst nach Ausführen eines "lister refresh" erneuert.

Beispiel:

```
 > lister remove 121132636 #5
```

Befehl: lister request

Syntax: lister request <handle> <text> <buttons>

Dieser Befehl ist identisch mit dem Befehl "dopus request", außer daß hier noch ein Listerhandle als zusätzlicher Parameter übergeben wird und der Requester über dem Dateilister zentriert wird.

<text> ist der angezeigte Textstring und <buttons> sind die Knöpfe, die Sie im Requester erscheinen lassen wollen. Knöpfe werden voneinander auch hier durch einen vertikalen Strich abgetrennt.

Beispiel:

```
 > lister request 121132636 '"Bitte wählen Sie" "Option1|Option2|Option3"'
```

Dies würde einen Requester mit dem Text "Bitte wählen Sie" ausgeben und Ihnen drei Knöpfe zur Auswahl bieten, die mit "Option1", "Option2" und "Option3" beschriftet sind.

Der Wert des angewählten Knopfes wird in RC zurückgegeben. Der letzte Knopf (in diesem Fall "Option3") ist immer als "Abbruch"-Knopf gedacht und bekommt daher automatisch den Wert 0 zugewiesen. Die bei diesem Beispiel zurückgegebenen Werte sind also 1, 2, oder 0.

Befehl: lister set

Syntax: lister set <handle> <item> <value>

Dies übergibt eine Information an den Dateilister. <handle> ist das Handle des fraglichen Listers. <item> kann eines der folgenden Schlüsselworte sein:

```
 busy <status> [wait]
```

Setzt den "BUSY"-Status des Listers. "0" oder "off" stellt "BUSY" aus, "1" oder "on" stellt es an. Sie können dabei auch das Schlüsselwort "wait" angeben, wodurch der Befehl synchron ausgeführt werden kann (d.h. mehrere Dateilister werden synchron geschaltet).

Beispiel:

```
 > lister set 121132636 busy on wait
 > lister set 128765412 busy 0
```

```
 case
```

Dieser Befehl schaltet die Unterscheidung von Groß- und Kleinschreinbung an oder aus. Da das Amigadateisystem diese Unterscheidung eigentlich nicht benötigt, ist die Standardeinstellung hier 0. Für einige der benutzerdefinierten Handler kann es aber durchaus sinnvoll sein, diese Unterscheidung zu aktivieren

Beispiel:

```
 > lister set 121132636 case on
 > lister set 121132636 case off
```

```
 dest [lock]
```

Setzt den Listerstatus auf Ziel. Geben Sie zusätzlich "lock" an, wird der Lister als Ziel gesperrt.

Beispiel:

```
 > lister set 121132636 dest
```

```
 display <items>
```

Setzt die Information über die darzustellenden Daten des Listers. Die Anzeige wird erst nach Ausführen eines "lister refresh" erneuert. Die Schlüsselworte sind dieselben wie bei "lister query".

Beispiel:

```
 > lister set 121132636 display name date size protect
```

```
 field [<number> <string>]
```

Hiermit können Sie Ihre eigenen Textstrings als Beschriftung für die Sortierknöpfe nutzen. Sie können hiermit nicht den Sinn dieser Spalten im Dateilister ändern - dies ändert lediglich die Überschrift.

Mit <number> (0-9) geben Sie an, welcher String ersetzt werden soll.

Bei 0 beginnend sind dies:

```
 name, size, access, date, comment, type, owner, group, net, version
```

Wenn Sie einen Leerstring übergeben, so wird hier wieder der Standardwert eingetragen. Die Anzeige wird erst nach Ausführen eines "lister refresh <handle> full" erneuert.

Sie können mit "lister set <handle> field off" alle Sortierknöpfe auschalten oder diese mit "on" wieder einschalten. Beachten Sie bitte, daß wenn die Sortierknöpfe in der Konfiguration nicht eingeschaltet sind, diese auch nicht über ein ARexx-Skript eingeschaltet werden können.

Beispiel:

```
 > lister set 121132636 field 0 Dateiname 4 Notizen
```

```
 flags <flags>
```

Setzt die Sortier- und Anzeigeflags des Listers. Die Anzeige wird erst nach Ausführen eines "lister refresh" erneuert. Mögliche Schlüsselworte sind:

```
 reverse   - Sortieren in umgekehrter Reihenfolge
 noicons   - Piktogramm filtern
 hidden    - Verborgene Dateien anzeigen
```

Beispiel:

```
 > lister set 121132636 flags reverse noicons
```

```
 header <string>
```

Dies arbeitet ähnlich "lister set title", außer daß der Text in der Statuszeile (D:x/y V:x/y) ändert. Der alte String wird in RESULT zurückgegeben. Übergeben Sie hier einen Leerstring, so werden die Standardwerte wieder hergestellt. Wollen Sie hier tatsächlich nichts anzeigen lassen, so übergeben Sie ein "-"

```
 handler <port name> [quotes] [fullpath]
```

Setzt den Portnamen des Handlers für diesen Lister. Dies ist der Name des Messageports, an den Nachrichten von Opus gesendet werden. Geben Sie das Schlüsselwort "quotes" an, so werden alle Dateinamen, die in Nachrichten an den Port gesendet werden, in Anführungszeichen eingeschlossen. Sie sollten dies nutzen, damit Sie auch Dateinamen übergeben können, die Leerzeichen enthalten. Wenn Sie das Schlüsselwort "fullpath" benutzen, so enthalten Nachrichten immer den vollen Pfadnamen einer Datei, unabhängig davon, ob diese von einem Opus-Dateilister kamen oder nicht (üblicherweise erhalten Sie, wenn Sie die Datei von einem Dateilister erhalten, nur den Dateinamen und das Listerhandle, mit dem Sie den Pfad auch herausfinden können).

Beispiel:

```
 > lister set 121132636 handler 'lhadir_handler' quotes
```

Lesen Sie dazu auch den Abschnitt über
benutzerdefinierte Handler
in diesem
Kapitel für weitere Informationen.

```
 hide <pattern>
```

Setzt den Filter für das Verbergen von Einträgen. Der Filter wird sofort angewendent, aber die Anzeige wird erst nach Ausführen eines "lister refresh" erneuert.

Beispiel:

```
 > lister set 121132636 hide '#?.info'
```

 label

Dieser Befehl wird benutzt, um den Namen zu übergeben, der unter einem als Piktogramm verborgenen Dateilister angezeigt werden soll. Um eine solche benutzerdefinierte Beschriftung zu entfernen, benutzen Sie einfach diesen Befehl ohne Übergabe eines Namens.

Beispiel:

```
 > lister set 121132636 label Mein Lister
 > lister set 121132636 label
```

 lock <type>

Die Parameter für <type> können momentan sein:

 state [on|off]

  Dieser Parameter erlaubt Ihnen, einen Dateilister in seinem aktuellen Zustand zu sperren, so daß der Benutzer dies nicht ändern kann, bevor Sie ihn nicht wieder freigeben.

 format [on|off]

  Dieser Parameter sperrt das aktuelle Anzeigeformat. Dies verursacht, daß das Fenster zum Ändern des Anzeigeformates nicht mehr vom Benutzer aufgerufen werden kann.

Sie können diese Parameter alle in einer Zeile übergeben.

Beispiel:

```
 > lister set 121132636 lock state on format on
```

 mode

Dieser Befehl stellt den Darstellungsmodus des Dateilisters ein. Die Schlüsselworte sind:

```
 name        - Textmodus
 icon        - Piktogrammodus
 icon action - Piktogramm Plus-Modus
```

Wenn Sie zudem "showall" angeben, so werden auch Dateien ohne eigene Piktogramme in den Piktogrammodi mit solchen versehen.

Beispiel:

```
 > lister set 121132636 mode name
 > lister set 121132636 mode icon action showall
```

```
 namelength
```

Dieser Befehl stellt die maximale Dateinamenlänge in Dateilistern ein. Minimallänge ist 30 Zeichen (dies ist auch der Standardwert). Dieser Befehl ist nur nützlich für Leute, die ihre eigenen Handler schreiben wollen. Beachten Sie bitte, daß die meisten der internen Befehle von Opus momentan noch keine Dateinamen mit mehr als 30 Zeichen Länge unterstützen.

Beispiel:

```
 > lister set 121132636 namelength 256
```

```
 newprogress [name] [file] [info] [bar] [abort]
```

Dies schaltet im spezifizierten Dateilister den Fortgangsindikator an. Dieser Befehl funktioniert ähnlich dem alten "lister set progress", erlaubt Ihnen aber weitere Kontrollmöglichkeiten über die dargestellten Informationen.

```
 name  - reserviert Platz für die Anzeige des Dateinamens
 file  - reserviert Platz für die Fortgangsanzeige
 info  - reserviert Platz für die Anzeige der Informationszeile
 bar   - reserviert Platz für die Anzeige des Fortgangsbalkens
 abort - fügt einen Abbruchknopf hinzu
```

Fortgangsfenster, die beides, einen Balken und den Dateifortgang, anzeigen, haben die Balken- und Dateianzeige vertauscht. Anstelle eines Balkens, der prozentual die Anzahl kopierter Dateien und einer Prozentanzeige, die den Fortgang der jeweiligen Datei anzeigt, zeigt der Balken den einzelnen Dateifortgang an und eine Anzeige ’xxx von yyy’ gibt Information über die Anzahl der kopierten Dateien.

Beispiel:

```
 > lister set 121132636 newprogress name file info bar abort
```

```
 newprogress name <filename>
```

Wenn das Fortgangsfenster mit dem Parameter "name" geöffnet wurde, können Sie hiermit den aktuellen Dateinamen übergeben.

Beispiel:

```
 > lister set 121132636 newprogress name ’MeineDatei.TXT’
```

```
 newprogress file <total> <count>
```

Wenn der Fortgangsindikator mit dem Parameter "file", aber ohne den Parameter "bar" geöffnet wurde, so stellen Sie hiermit die Gesamtanzahl der Dateien und die Nummer der aktuellen Datei ein. Dies wird in der oberen rechten Ecke des Fensters als ’xx%’ angezeigt.

Wenn der Fortgangsindikator mit den Parametern "file" und "bar" geöffnet wurde, so stellen Sie hiermit die Gesamtzahl an Bytes und den aktuellen Bytezähler ein. Dies wird dann in der Balkenanzeige des Fensters dargestellt.

Beispiel:

```
 > lister set 121132636 newprogress file 12 4
```

```
 newprogress info <text>
```

<text> ist der Textstring, der zwischen dem Dateinamen und der Balkenanzeige des Fortgangsindikator dargestellt werden soll.

Beispiel:

```
 > lister set 121132636 newprogress info "Von 'T' nach 'RAM:'"
```

```
 newprogress bar <total> <count>
```

Wenn der Fortgangsindikator mit dem Parameter "bar", aber ohne den Parameter "file" geöffnet wurde, so stellen Sie hiermit die Gesamtzahl an Bytes und den aktuellen Bytezähler ein. Dies wird dann in der Balkenanzeige des Fensters dargestellt.

Wenn der Fortgangsindikator mit den Parametern "file" und "bar" geöffnet wurde, so stellen Sie hiermit die Gesamtanzahl an Dateien und die Nummer der aktuellen Datei ein. Dies wird dann als 'xxx von yyy'~in der oberen rechten Ecke des Fensters angezeigt.

Beispiel:

```
 > lister set 121132636 newprogress bar 1024 100
```

```
 newprogress title <text>
```

<text> ist der Textstring, der in der Titelzeile des Fortgangsindikators angezeigt werden soll.

Beispiel:

```
 > lister set 121132636 newprogress title 'Kopiere...'
```

Sie können die alten Befehle von "lister set progress" auch auf einen mit "newprogress" geöffneten Fortgangsindikator anwenden, jedoch können Sie damit nur den Dateinamen und den Balkenzähler ändern. Benutzen Sie "lister clear progress", um alte oder neue Fortgangsindikatoren zu entfernen.

```
 off
```

Setzt den Listerstatus auf "AUS" (weder Ziel noch Quelle).

Beispiel:

```
 > lister set 121132636 off
```

```
 path <path string>
```

Setzt den aktuellen Pfad des Listers. Bedenken Sie, daß dies nicht dazu führt, daß das angegebene Verzeichnis eingelesen wird; es wird lediglich der Anzeigestring geändert. Um das Verzeichnis auch einzulesen benutzen Sie bitte den Befehl "lister read".

Beispiel:

```
 > lister set 121132636 path 'dh0:work'
```

```
 position <x/y/w/h>
```

Dies stellt die Position und Größe des Dateilisters ein, wenn dessen Position nicht gesperrt ist. Wenn der Lister sichtbar ist, wird dies augenblicklich ausgeführt.

Beispiel:

```
 > lister set 121132636 position 20/20/400/300
```

```
 progress <total> <text>
```

Beachten Sie bitte, daß diese Befehle ersetzt wurden durch "lister set newprogress". Benutzen Sie bitte in neuen Skripten stattdessen diese Befehle.
Dies schaltet einen Fortgangsindikator im Dateilister an.

<total> spezifiziert die Anzahl der möglichen Einzelschritte und kontrolliert damit die Anzeige des Fortgangsbalkens. Ein Wert von "-1" läßt keinen Balken erscheinen. <text> ist ein Textstring, der im Kopf des Fortgangsindikators angezeigt wird.

Beispiel:

```
 > lister set 121132636 progress 38 'Archiviere Dateien ...'
```

```
 progress count <count>
```

Dies erneuert den Balken im Fortgangsindikator (dieser muß dazu schon eingeschaltet sein). <count> ist der Fortgang des Zählers, der durch den Balken angezeigt werden soll. <count> muß bei jedem Schritt größer werden, eine Verkleinerung der Position ist nicht zugelassen.

Beispiel:

```
 > lister set 121132636 progress count 4
```

```
 progress name <name>
```

Dies erneuert die Anzeige eines Dateinamens im Fortgangsindikator. Der

```
Dateiname wird oberhalb des Balkens angezeigt.

Beispiel:

 > lister set 121132636 progress name 'Datei.TXT'


 separate <method>

Setzt die Eintragsordnung für diesen Dateilister. Die Liste der Einträge
wird intern sofort neu sortiert, aber die Anzeige wird erst erneuert, wenn
Sie dies mit dem Befehl "lister refresh" anordnen. Mögliche Schlüsselworte
sind:

 mix        - Dateien und Verzeichnisse gemischt
 dirsfirst  - Verzeichnisse zuerst
 filesfirst - Dateien zuerst

Beispiel:

 > lister set 121132636 separate mix


 show <pattern>

Setzt den Filter für die Anzeige von Einträgen. Der Filter wird sofort
aktiv, aber die Anzeige wird erst nach Ausführen eines "lister refresh"
erneuert.

Beispiel:

 > lister set 121132636 show '#?.c'


 sort <method>

Setzt die Sortiermethode des Listers fest. Die Liste der Einträge wird
intern sofort neu sortiert, aber die Anzeige wird erst erneuert, wenn Sie
den Befehl "lister refresh" geben. Die möglichen Schlüsselworte sind:

 name      - Dateiname
 size      - Dateigröße
 protect   - Schutzbits
 date      - Datum
 comment   - Kommentar
 filetype  - Dateityp
 version   - Dateiversion

Beispiel:

 > lister set 121132636 sort date
 > lister set 121132636 sort filetype


 source [lock]

Setzt den Listerstatus auf Quelle. Geben Sie zusätzlich "lock" an, wird der
```

Lister als Quelle gesperrt.

Beispiel:

```
 > lister set 121132636 source lock
```

```
 title <string>
```

Setzt den Titel des Dateilisters (angezeigt in der Titelzeile des Listers). Die Titelzeile wird erst erneuert nach Ausführen eines "lister refresh full" (Lesen Sie dazu später mehr). Der alte Titel des Listers wird in RESULT zurückgegeben.

Beispiel:

```
 > lister set 121132636 title 'hello'
                                      ---> Work
 > lister set 121132636 title
                                      ---> hello
```

```
 toolbar <filename>
```

Dieser Befehl ändert die Werkzeugleiste, die im spezifizierten Dateilister benutzt wird.

Beispiel:

```
 > lister set 121132636 toolbar RAM:New_Toolbar
```

```
 visible <state>
```

Macht einen Lister sichtbar oder unsichtbar. Standardmäßig sind Lister beim Erzeugen sichtbar. Setzen Sie seinen Status allerdings auf "0" oder "off", wird der Lister nicht mehr angezeigt, bis Sie ihn wieder sichtbar machen.

Beispiel:

```
 > lister set 121132636 visible off
 > lister set 121132636 visible 1
```

```
Befehl: lister select
```

```
Syntax: lister select <handle> <name> <state>
```

Dieser Befehl ändert den Anwahlstatus eines Eintrags im spezifizierten Lister. <name> ist entweder der Name des Eintrags oder #xxx (xxx ist eine Zahl) zur Festlegung der Nummer des Eintrags. <state> ist der gewünschte Anwahlstatus ("0" oder "off" für deselektiert, "1" oder "on" für angewählt). Wird <state> nicht angegeben, wird der Anwahlstatus der angegebenen Datei umgeschaltet. Die Anzeige wird erst nach Ausführen eines "lister refresh" erneuert. Der vorherige Anwahlstatus wird in RESULT zurückgegeben.

Beispiel:

```
 > lister select 121132636 ENV on
                                  ---> off
```

Befehl: lister wait

Syntax: lister wait <handle> [quick]

Dieser Befehl veranlaßt das ARexx-Skript auf die Beendigung der momentan im spezifizierten Listers ausgeführten Operation zu warten, was immer auch die dort ausgeführte Operation ist. Aufgrund des Multitaskings von Opus 5, wird mit der Ausführung der ARexx-Befehle (wie "lister read" oder "lister new") sofort fortgefahren, selbst wenn der Lister die ihm zugewiesene Ausgabe noch nicht ausgeführt hat. Mit diesem Befehl veranlassen Sie ARexx zu warten, bis der Status des Listers nicht mehr "BUSY" ist. Ist der spezifizierte Lister bei Ausführung dieses Befehls nicht "BUSY", wartet das Programm bis zu zwei Sekunden, ob dieser auf "BUSY" schaltet, und fährt anderenfalls mit der Abarbeitung des Programms fort. Wird das Schlüsselwort "quick" angegeben, so wird der Befehl sofort weitermachen, wenn der Lister nicht "BUSY" ist, anstatt zwei Sekunden zu warten. Es wäre übrigens ziemlich nutzlos, ein "lister set busy 1" auszuführen und danach ein "lister wait".

Beispiel:

```
 > lister read 121132636 'c:'
 > lister wait 121132636
```

Befehl: lister iconify

Syntax: lister iconify (all|<handle>) <state>

Hiermit schalten Sie alle oder den spezifizierten Dateilister in den verborgenen Zustand als Piktogramm, wenn Sie für <state> on, 1 oder auch diesen Parameter weglassen. Um die Dateilister wieder zu öffnen muß <state> off oder 0 sein.

Beispiel:

```
 > lister iconify 121132636
 > lister iconify all off
```

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen. Der mit einem (*) gekennzeichnete Unterpunkt ist der, in dem Sie sich aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des Anzeigeprogramms.

Kapitelinhalt
Index

1.6 Der Basisbefehl 'command'

16.2.3 Der Basisbefehl "command"

Der dritte Basisbefehl von Opus 5 ist der Befehl "command". Dieser erlaubt es Ihnen, die internen Befehle von Opus 5 aus dem ARexx-Skript heraus aufzurufen. Die internen Befehle werden dabei exakt so ausgeführt, als wären sie durch einen Knopf oder ein Menü aufgerufen worden.

```
command [wait] [source <handle>] [dest <handle>] [original] command
[arguments]
```

Wenn Sie das Schlüsselwort "wait" angeben, so wird der Befehl synchron gestartet, anderenfalls kehrt er sofort zurück. Üblicherweise arbeiten die meisten Befehle mit dem Quell- und einem Zieldateilister. Darüberhinaus können Sie mit "source" und "dest" auch alternative Dateilister angeben.

Das Schlüsselwort "original" erlaubt Ihnen die Ausführung eines "echten" internen Befehls, wenn dieser in der Befehlsliste durch einen externen Befehl ersetzt wurde (externe Module, die Befehle zu Opus hinzufügen, haben höhere Priorität als die interne Liste). Das bedeutet, daß Sie ein Modul haben können, das einige Opus-Befehle ersetzt und unter bestimmten Umständen einige Dinge auf besondere Art erledigt, und in anderen Fällen rufen Sie einfach die originale, interne Funktion auf.

Der Parameter "command" ist der Name des auszuführenden Befehls und "arguments" sind, wie üblich, die optionalen Argumente für diesen Befehl.

Beispiele:

```
> command all
> command wait copy
> command read s:startup-sequence
> command source 121132636 makedir name=MeinVerzeichnis noicon
> command original wait delete ram:#?
```

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen. Der mit einem (*) gekennzeichnete Unterpunkt ist der, in dem Sie sich aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des Anzeigeprogramms.

Der Basisbefehl 'dopus'

Der Basisbefehl 'lister'
*
Der Basisbefehl 'command'
Hauptinhalt
Kapitelinhalt
Index

1.7 ARexx-Fehlercodes

16.3 ARexx-Fehlercodes

Listerhandles sind die aktuellen Speicheradressen der Dateilisterstruktur. Opus 5 weist jedes ungültige Handle mit einem Rückgabewert (RC) von 10 zurück. Alle Befehle, die Daten zurückgeben, tun dies in der Variable RESULT (mit Ausnahme von "dopus getstring" und "lister getstring") oder in einer von Ihnen spezifizierten Stammvariablen. Tritt ein Fehler dabei auf, wird der Fehlercode in der Variable RC zurückgegeben. Ein Wert von 0 in RC bedeutet, daß kein Fehler aufgetreten ist.

Die aktuell gültigen Fehlercodes sind:

1 RXERR_FILE_REJECTED

Die Datei, die Sie hinzufügen wollten, wurde von den Filtern des Dateilisters zurückgewiesen.

Beachten Sie, daß dies kein Fehler, sondern lediglich eine Warnung ist. Die Datei wird trotzdem hinzugefügt, wird aber nicht sichtbar, bevor nicht die Filter verändert werden.

5 RXERR_INVALID_QUERY
RXERR_INVALID_SET

Die Anweisung, die Sie als Schlüsselwort bei "lister query" oder "lister set" übergeben wollten, ist nicht gültig.

6 RXERR_INVALID_NAME
RXERR_INVALID_KEYWORD

Der spezifizierte Dateiname oder das Schlüsselwort waren nicht gültig.

8 RXERR_INVALID_TRAP

Der Trap, den Sie entfernen wollten, exisitiert nicht.

10 RXERR_INVALID_HANDLE

Das spezifizierte Listerhandle ist nicht gültig.

12 RXERR_NO_TOOLBAR

Der Dateilister hat keine gültige Werkzeugleiste.

15 RXERR_NO_MEMORY

Nicht genug Speicher für die von Ihnen verlangte Operation.

```
20      RXERR_NO_LISTER

Ein Dateilister konnte nicht geöffnet werden (üblicherweise aufgrund von
Speicherplatzmangel)

Sie können von hier direkt alle Abschnitte dieses Kapitels anwählen.
Der mit einem (*) gekennzeichnete Abschnitt ist der, in dem Sie sich
aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des
Anzeigeprogramms.


                ARexx

                ARexx-Befehle
                *
                ARexx-Fehlercodes

                Benutzerdefinierte Handler

                ARexx-Module
                   Hauptinhalt
                Kapitelinhalt
                 Index
```

1.8 Benutzerdefinierte Handler

```
                16.4  Benutzerdefinierte Handler

Das System benutzerdefinierter Handler erlaubt es Ihnen, den Namen eines
externen, öffentlichen Messageports zu definieren. Diesem Port werden immer
dann Nachrichten zugesendet, wenn bestimmte Dinge geschehen, an denen Sie
interessiert sind.  Nachrichten, die dorthin versendet werden, sind immer
korrekt formatierte ARexx-Nachrichten. Ein beispielhaftes Codefragment für
den Empfang einer Nachricht ist:

     call waitpkt(myportname)      /* Warte auf einkommende Nachrichten */

     packet=getpkt(myportname)     /* Empfange Wartenachricht */
     arg0=getarg(packet,0)         /* Empfange Argument 0 */
     arg1=getarg(packet,1)         /* Empfange Argument 1 */
     arg2=getarg(packet,2)         /* Empfange Argument 2, etc... */

     call reply(packet,0)          /* Antwort auf empfangene Nachricht */

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen.


                Benutzerdefinierte Handler für Dateilister

                Abgefangene Funktionen

                Popup-Menüereignisse bei Dateien

                Benutzerdefinierte Handler für AppIcons
```

```
                Sie können weiterhin direkt alle Abschnitte dieses Kapitels ↩
                   anwählen.
Der mit einem (*) gekennzeichnete Abschnitt ist der, in dem Sie sich
aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des
Anzeigeprogramms.


                ARexx

                ARexx-Befehle

                ARexx-Fehlercodes
                *
                Benutzerdefinierte Handler

                ARexx-Module
                   Hauptinhalt
                Kapitelinhalt
                 Index
```

1.9 Benutzerdefinierte Handler für Dateilister

```
                16.4.1  Benutzerdefinierte Handler für Dateilister
```

Ein benutzerdefinierter Handler wird einem Lister zugewiesen durch Aufruf von "lister set <handle> handler" für diesen Dateilister, wobei Sie den Namen Ihres Messageports übergeben. Immer, wenn etwas Interessantes mit Ihrem Dateilister geschieht, wird dem Handler eine ARexx-Nachricht geschickt. Der Handler kann als Rexx- oder als C-Programm implementiert werden (wobei in letzterem Fall die Nachricht selbst interpretiert werden muß). Anders als bei Opus 4 sorgt eine Nachricht, die an einen Handler verschickt wird, nicht dafür, daß Opus "hängt", bis auf diese Nachricht geantwortet wird (obwohl Sie natürlich trotzdem so schnell wie möglich auf eine Nachricht antworten sollten).

Bedenken Sie, daß benutzerdefiniertee Handler nur spezifisch für den im Lister sichtbaren Puffer zum Zeitpunkt der Namensfestlegung sind. Derselbe Handlerport kann auch für mehrere Puffer und sogar für mehrere Lister gesetzt werden. Beachten Sie bitte auch, daß die Groß- und Kleinschreibung bei den Namen der Messageports relevant ist.
Die Rexx-Nachricht identifiziert die Art des Ereignisses, den Lister, in dem dieses Ereignis stattfand und andere angemessene Daten.

Die Ereignisse, von denen Sie benachrichtigt werden können, sind momentan:

doubleclick (Doppelklick)

Wenn ein Doppelklick als Ereignis auftritt bedeutet dies, daß ein Eintrag des Listers vom Benutzer mit einem Doppelklick aktiviert wurde.
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "doubleclick"    (Ein String, der den Ereignistyp angibt)
   Arg1 - <handle>         (Listerhandle)
   Arg2 - <name>           (Name des Eintrags)
```

```
   Arg3 - undefiniert
   Arg4 - undefiniert
   Arg5 - <userdata>       (Wenn <userdata> mit dem Befehl "lister addstem"
                            definiert wurde)
   Arg6 - <qualifiers>     (String, der das Drücken folgender Tasten abgibt:
                            shift, alt, control (Amiga-Taste))
```

drop (Ablegen)

Die ist ein "Nehmen & Ablegen"-Ereignis und tritt auf, wenn einer oder mehrere Einträge in einem Lister abgelegt wurden.
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "drop"           (Ereignistyp)
   Arg1 - <handle>         (Ziellisterhandle)
   Arg2 - <names>          (Dateinamen)
   Arg3 - <handle>         (Quellisterhandle)
   Arg4 - undefiniert
   Arg5 - undefiniert
   Arg6 - <qualifiers>     (shift, alt, control (Amiga-Taste))
```

Bei mehreren übergebenen Dateinamen werden diese durch Leerschritte voneinander getrennt und werden, wenn das Schlüsselwort "quotes" beim Befehl "lister set handler" angegeben wurde, in Anführungszeichen eingeschlossen. Stammen die Dateien aus einem anderen Lister von Opus 5, dann wird in Arg3 das Handle dieses Listers übergeben. Ist dies der Fall und das Schlüsselwort "fullpath" wurde bei "lister set handler" nicht angegeben, so werden nur die Dateinamen (und nicht deren Pfade) in Arg2 übergeben. Der Quellpfad läßt sich dann aber leicht mit "lister query" herausfinden. Ist Arg3 gleich Null, dann werden die abgelegten Objekte wahrscheinlich von der Workbench stammen und die Dateinamen in Arg2 enthalten den vollen Pfad.

dropfrom (Nehmen von)

Dies entspricht genau dem Ereignis "drop", außer daß es das Nehmen von einem Lister anzeigt, anstatt das Ablegen auf einem.
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "dropfrom"       (Ereignistyp)
   Arg1 - <handle>         (Ziellisterhandle)
   Arg2 - <names>          (Dateinamen)
   Arg3 - <handle>         (Quellisterhandle)
   Arg4 - undefiniert
   Arg5 - undefiniert
   Arg6 - <qualifiers>     (shift, alt, control (Amiga-Taste))
```

Beachten Sie bitte, daß AppIcons auf "dropfrom"-Ereignisse empfangen können, jedoch weichen die Argumente dabei ein wenig ab. Sie können daran erkannt werden, daß in Arg4 das Wort "icon" immer präsent ist. Lesen Sie weiter unten mehr dazu.

parent (Mutterverzeichnis)

Dieses Ereignis wird empfangen, wenn der Menüpunkt "Mutterverzeichnis" aus dem Popup-Menü mit der Verzeichnisliste angewählt wird, oder, wenn der Benutzer auf den verborgenen Mutterverzeichnisknopf drückt oder den Hotkey für das Mutterverzeichnis benutzt ("/").
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "parent"          (Ereignistyp)
   Arg1 - <handle>          (Quellisterhandle)
   Arg2 - <path>            (Pfad des Dateilisters)
   Arg3 - undefiniert
   Arg4 - undefiniert
   Arg5 - undefiniert
   Arg6 - <qualifiers>      (Nur die Shift-Tasten)
```

root (Hauptverzeichnis)

Dieses Ereignis wird empfangen, wenn der Menüpunkt "Hauptverzeichnis" aus dem Popup-Menü mit der Verzeichnisliste angewählt wird, oder, wenn der Benutzer den Hotkey für das Mutterverzeichnis benutzt (":").
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "root"            (Ereignistyp)
   Arg1 - <handle>          (Quellisterhandle)
   Arg2 - <path>            (Pfad des Dateilisters)
   Arg3 - undefiniert
   Arg4 - undefiniert
   Arg5 - undefiniert
   Arg6 - <qualifiers>      (Nur die Shift-Tasten)
```

path (Pfad)

Diese Nachricht erhalten Sie, wenn der Benutzer einen neuen Pfad in das Pfadeingabefeld des Dateilisters eingibt.
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "path"            (Ereignistyp)
   Arg1 - <handle>          (Listerhandle)
   Arg2 - <path>            (Pfad des Dateilisters)
```

reread (Verzeichnis erneut einlesen)

Dieses Ereignis wird an Ihren Handler gesendet, wenn "Verzeichnis erneut einlesen" aus dem Popup-Menü mit der Verzeichnisliste angewählt wird.
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "reread"          (Ereignistyp)
   Arg1 - <handle>          (Listerhandle)
   Arg2 - <path>            (Neuer Pfad für den Dateilister)
```

active (aktiv)

Dieses Ereignis zeigt an, daß ein Puffer mit benutzerdefiniertem Handler sichtbar wurde.

Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - "active"          (Ereignistyp)
   Arg1 - <handle>          (Listerhandle)
   Arg2 - <title>           (Name des Puffers)
   Arg3 - undefiniert
   Arg4 - <path>            (Pfad des Dateilisters
```

Arg2 enthält den Titel des aktiv gewordenen Puffers, wenn dieser mit lister set title" eingestellt wurde. Wurde kein Titel benutzerdefiniert, wird der Pfadstring des Puffers stattdessen übergeben (d.h. in einem solchen Fall sind Arg2 und Arg4 identisch).

inactive (inaktiv)

Dieses Ereignis zeigt an, daß der Puffer dieses benutzerdefinierten Handlers nicht länger aktiv (sichtbar im Lister) ist. Die Argumente der Nachricht stimmen mit denen von "active" überein, abgesehen vom anderen Ereignistyp in Arg0. Diese Nachricht tritt auf, wenn der Puffer des Listers verändert wird (entweder durch den Benutzer oder durch Rexx) oder wenn der Lister geschlossen wird. Beachten Sie, daß Sie eventuell eine Nachricht "active" für einen anderen Puffer mit benutzerdefiniertem Handler oder sogar für denselben Puffer, direkt nach Erhalt der Nachricht "inactive", erhalten.

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen. Der mit einem (*) gekennzeichnete Unterpunkt ist der, in dem Sie sich aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des Anzeigeprogramms.

1.10 Abgefangene Funktionen

16.4.2 Abgefangene Funktionen

Nachrichten für abgefangene Befehle werden wie andere Nachrichten auch an den Lister gesendet.
Die übergebenen Argumente der Nachricht sind:

```
   Arg0 - <command>     (Name des Befehls oder "abort")
   Arg1 - <handle>      (Quellisterhandle, falls verfügbar)
   Arg2 - <files>       (Angewählte Dateien, falls verfügbar)
   Arg3 - <handle>      (Ziellisterhandle, falls verfügbar)
```

```
   Arg4 - <path>         (Quellpfad; dies ist nützlich, wenn kein Lister
                          damit verbunden ist)
   Arg5 - <args>         (Vom Benutzer mit der Funktion übergebene Argumente)
   Arg7 - <path>         (Zielpfad; erlaubt Ihnen die Unterstützung des
                          "Zielwahl"-Requesters)

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen.
Der mit einem (*) gekennzeichnete Unterpunkt ist der, in dem Sie sich
aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des
Anzeigeprogramms.


                Benutzerdefinierte Handler für Dateilister
                *
                Abgefangene Funktionen

                Popup-Menüereignisse bei Dateien

                Benutzerdefinierte Handler für AppIcons
                   Hauptinhalt
                Kapitelinhalt
                 Index
```

1.11 Popup-Menüereignisse bei Dateien

```
                16.4.3  Popup-Menüereignisse bei Dateien

Wenn Sie Datei zu einem Dateilister mit "lister addstem" hinzugefügt haben
und Ihre eigenen Popup-Menüs für diese Dateien spezifiziert haben, so
werden Sie Nachrichten erhalten, wenn diese Menüs vom Benutzer angewählt
werden.
Die übergebenen Argumente der Nachricht sind:

   Arg0 - "menu"         (Ereignistyp Menü)
   Arg1 - <handle>       (Listerhandle)
   Arg2 - <name>         (Name des Eintrags)
   Arg3 - <id>           (ID des Menüeintrags + Basis-ID, falls angegeben)
   Arg4 - "file"         (String, der dies als Dateimenüereignis identifiziert)
   Arg5 - <userdata>     (Wenn <userdata> mit dem Befehl "lister addstem"
                          definiert wurde)

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen.
Der mit einem (*) gekennzeichnete Unterpunkt ist der, in dem Sie sich
aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des
Anzeigeprogramms.


                Benutzerdefinierte Handler für Dateilister

                Abgefangene Funktionen
                *
                Popup-Menüereignisse bei Dateien

                Benutzerdefinierte Handler für AppIcons
                   Hauptinhalt
```

Kapitelinhalt
Index

1.12 Benutzerdefinierte Handler für AppIcons

16.4.4 Benutzerdefinierte Handler für AppIcons

AppIcons, die mit dem Befehl "addappicon" hinzugefügt wurden, sorgen auch dafür, daß Nachrichten gesendet werden. Alle Nachrichten von AppIcons haben dieselben Argumente:

```
   Arg0 - <event>       (String, der das Ereignis identifiziert (s.u.))
   Arg1 - <id>          (Die bei "addappicon" spezifizierte ID)
   Arg2 - <data>        (Dateinamen/Menü-ID/andere Informationen)
   Arg3 - <handle>      (Quellisterhandle, falls anwendbar)
   Arg4 - "icon"        (String, der dies als Piktogrammereignis
                         identifiziert)
```

Diese Ereignisse können bei AppIcons (siehe Arg0) auftreten:

doubleclick

Dies zeigt an, daß das Piktogramm doppelgeklickt wurde oder daß "Öffnen" aus dem Menü gewählt wurde.

dropfrom

Dies ist ein "Nehmen & Ablegen"-Ereignis, das anzeigt, daß einer oder mehrere Einträge auf diesem Piktogramm von einem Dateilister oder einem anderen Ort abgelegt wurden. Die Namen dieser Einträge sind in Arg2 verfügbar.

snapshot

Dieses Ereignis tritt auf, wenn der Menüpunkt "Fixieren" gewählt wurde. Die aktuelle Position des Piktogramms ist in Arg2 verfügbar (als x,y-String). Sie sind selber für die Speicherung dieser Position verantwortlich.

unsnapshot

Dieses Ereignis tritt auf, wenn der Menüpunkt "Position freigeben" gewählt wurde.

removed

Dieses Ereignis warnt davor, daß Opus beendet wurde und der Handlercode nun aufräumen und auch beenden sollte.

info

Dieses Ereignis tritt auf, wenn der Menüpunkt "Information" gewählt wird.

close

Dieses Ereignis tritt auf, wenn "Schließen" aus dem Popup-Menü gewählt

wurde.

menu

Dieses Ereignis zeigt an, daß einer der benutzerdefinierten Menüpunkte des Popup-Menüs angewählt wurde. Die Nummer des Menüeintrags wird in Arg2 übergeben.

menuhelp

Dieses Ereignis zeigt an, daß die Help-Taste gedrückt wurde, während der Mauszeiger sich über einem der benutzerdefinierten Menüpunkte des Popup-Menüs befand. Die Nummer des Menüeintrags wird in Arg2 übergeben.

Aufgrund der Natur von Opus 5 mit seinem internen Multitasking, kann man sich nicht hundertprozentig auf die Informationen, die benutzerdefinierte Handler erhalten, verlassen. Zum Beispiel kann es geschehen, daß Sie die Nachricht "active" erhalten, obwohl der Puffer, der dies auslöste, direkt wieder inaktiv wurde. Sie sollten daher überprüfen, ob der Port frei von Nachrichten ist, bevor Sie neue bearbeiten und Sie sollten den Befehl "lister query" benutzen, um sicherzustellen, daß die Dinge so sind, wie Sie es erwarten. Bedenken Sie auch, daß Lister jederzeit durch den Benutzer geschlossen werden können (es sei denn, Sie haben "BUSY" gesetzt).
Um zu überprüfen, ob ein Lister noch geöffnet ist, benutzen Sie "lister query path" (oder auch einen anderen Befehl der Gruppe "lister query"). Existiert der fragliche Lister nicht mehr, wird in der Variable RC der Fehlercode 10 (RXERR_INVALID_HANDLE) zurüchgegeben.

Seien Sie sich bewußt, daß diese Möglichkeiten bestehen. Generell werden keine Probleme auftreten, solange nicht der Benutzer (und das dürften meistens Sie selber sein) während der Ausführung solcher Programme darin "herumspielt".

Sie können von hier direkt alle Unterpunkte dieses Abschnitts anwählen. Der mit einem (*) gekennzeichnete Unterpunkt ist der, in dem Sie sich aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des Anzeigeprogramms.

1.13 ARexx-Module

16.5 ARexx-Module

ARexx-Module sind ARexx-Skripte, die im Verzeichnis "DOpus5:Modules" installiert sind. Um korrekt zu arbeiten, müssen sie die Dateiendung ".dopus" tragen.

Jedes ARexx-Modul kann neue, interne Befehle zu Opus hinzufügen.

Wenn beides, Opus 5 und ARexx, gestartet wurde, wird zuerst das Verzeichnis DOpus5:Modules" nach ARexx-Modulen gescannt, woraufhin dann bei jedem gefundenen Modul die "init"-Funktion aufgerufen wird. Jedes ARexx-Modul MUß eine Init-Funktion besitzen, ansonsten wird es nicht funktionieren. Diese Funktion ist dafür zuständig, daß die zusätzlichen Befehle in Opus 5 eingebunden werden.

Die Skripte werden mit vier oder mehr Parametern aufgerufen. Die ersten vier werden immer übergeben: Der Portname von Opus 5, der Funktionsname und die Handles von Quell- und Zieldateilister. Danach folgen weitere, durch den Benutzer spezifizierte Argumente.

Sie können soviele Befehle hinzufügen, wie Sie wünschen. Um Befehle hinzuzufügen, benutzen Sie den Befehl "dopus command". Die Syntax zum Aufruf ist:

```
dopus command <name> program <scriptname> [desc <description>]
[template <template>] [source] [dest]

name        - Der Name des Befehls.
scriptname  - Dateiname des Skriptes (dies muß der Name des Skriptes sein,
              so wie er im Verzeichnis "DOpus5:Modules" steht, jedoch ohne
              die Endung ".dopus5").
description - Optionale Befehlsbeschreibung (für die abrufbare Befehlsliste
             in den Editoren).
template    - Optionale Befehlsschablone im Stil von ReadArgs (das Prüfen
              der übergebenen Parameter müssen Sie selbst übernehmen).
source      - Zeigt an, daß der Befehl einen Quelldateilister braucht.
dest        - Zeigt an, daß der Befehl einen Zieldateilister braucht.
```

Das Feld "program" ist unabdingbar und Opus führt den Skriptnamen, den Sie hier angeben, bei jedem Aufruf der Funktion aus. Die Felder "description" und "template" sind optional. Wenn Sie die Schlüsselworte "source" oder "dest" (oder beide) angeben, so werden die passenden Listerhandle als Argumente an das Skript übergeben. Sind diese Schlüsselworte nicht in Gebrauch, so werden diese Argumente dennoch übergeben, sind dann aber gleich Null. Wenn Sie benutzerdefinierte Argumente an das Ende der Befehlszeile anhängen, sind Sie für deren Abfrage selbst verantwortlich.

Hier ist das komplette Beispiel eines ARexx-Moduls:

```
/* Beispielhaftes Directory Opus 5 ARexx-Modul */

parse arg portname function source dest arguments
address value portname
options results

/* Initialisieren */

if function='init' then do
```

```
     dopus command "Test1" program "test-command" desc "'Test Befehl 1'" template  ↩
        "TEST/S"
     dopus command "Test2" program "test-command" desc "'Test Befehl 2'" source
     exit
     end


/* Test function 1 */

if function='Test1' then do
     dopus request "'Test Befehl 1 empfangen!'" "Ok"
     exit
     end


/* Test function 2 */

if function='Test2' then do
     str="'Test Befehl 2 empfangen - Quellhandle " || source
     dopus request str "Ok"
     exit
     end
```

Sie können von hier direkt alle Abschnitte dieses Kapitels anwählen.
Der mit einem (*) gekennzeichnete Abschnitt ist der, in dem Sie sich
aktuell befinden. Benutzen Sie zum Blättern bitte die Knöpfe des
Anzeigeprogramms.