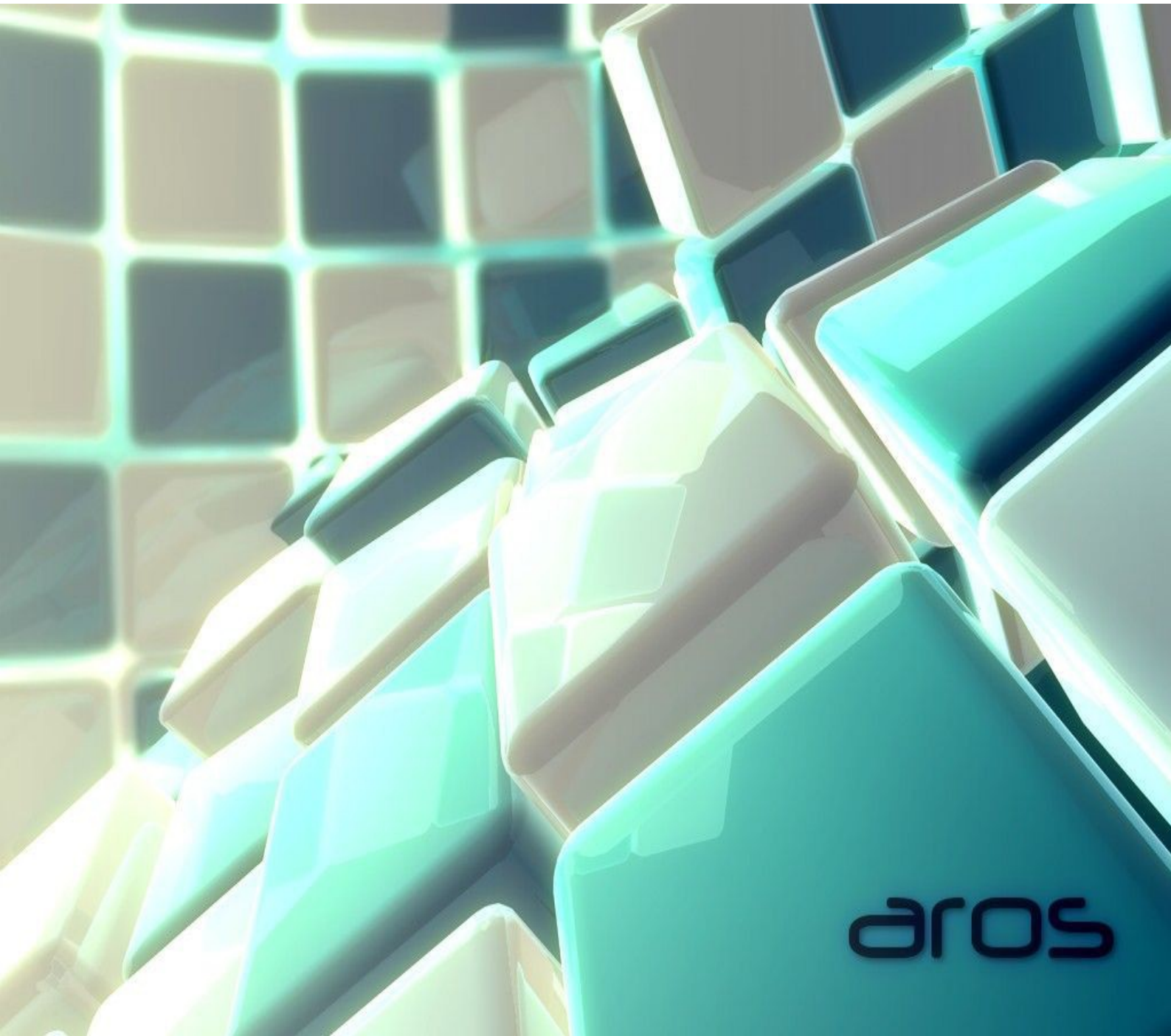




AROS Shell Manual



aros

AROS PUBLIC LICENSE (APL)

You may obtain a copy of the License at
<http://www.aros.org/license.html>

DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

COPYRIGHT AND TRADEMARKS.

Copyright © 1995-2007, The AROS Development Team. All rights reserved.
Amiga® is a trademark of Amiga Inc. All other trademarks belong to their respective owners.
The AROS Kitty mascot is © 2003 by Eric Schwartz
The image used in the cover is © 2005 by Damir Sijakovic.

This manual was produced using a variety of resources and documentation available in www.aros.org and other AROS resource sites by João Ralha, Peter Hutchison and Paolo Besser.

Original documentation written by Matthias Rustler.

Poseidon commands and utilities documentation by Chris Hodges.

AROSTCP documentation based on the AmiTCP documentation, copyright © 1993 AmiTCP/IP Group and © 1982, 1986, 1990 The Regents of the University of California.

Other commands documentation is copyright © their respective owners.

Everything included in this manual has been collected for the only purpose to provide a complete, exhaustive resource for all CLI commands and utilities included in AROS nightly builds.

TABLE OF CONTENTS

WELCOME.....	7
DOCUMENTATION CONVENTIONS.....	7
OTHER USEFUL OR REFERENCE DOCUMENTATION.....	8
 INTRODUCTION.....	 9
FEATURES.....	9
 CHAPTER 1: AROS SHELL BASICS.....	 11
AROS SHELL.....	11
PATH.....	11
COMMAND TEMPLATE.....	13
PATTERNS.....	14
REDIRECTION.....	14
PIPE.....	14
SPECIAL DEVICES.....	15
RUNNING IN BACKGROUND.....	15
BASIC FILE OPERATION WITH SHELL:.....	15
<i>Creating a directory.....</i>	<i>15</i>
<i>Deleting a file.....</i>	<i>16</i>
WORKING WITH SMART FILE SYSTEM (SFS).....	16
<i>SFS case-sensitive filenames.....</i>	<i>16</i>
<i>Undelete files.....</i>	<i>16</i>
<i>Minding your disk space.....</i>	<i>17</i>
<i>The 'Help' utility.....</i>	<i>17</i>
 SCRIPT PROGRAMMING.....	 18
AROS-DOS SCRIPTING.....	18
WORKING WITH A SCRIPT FILE.....	18
<i>Editing.....</i>	<i>18</i>
<i>Executing from shell.....</i>	<i>19</i>
<i>Executing from wanderer.....</i>	<i>19</i>
<i>Command type execution*.....</i>	<i>19</i>
CHANGING SYSTEM SCRIPT FILES.....	19
 AROS COMMAND REFERENCE.....	 21
<i>About file locations.....</i>	<i>22</i>
<i>AddAudioModes.....</i>	<i>23</i>
<i>AddBuffers.....</i>	<i>23</i>
<i>AddDatatypes.....</i>	<i>24</i>
<i>AddUSBClasses.....</i>	<i>24</i>
<i>AddUSBHardware.....</i>	<i>24</i>
<i>Alias.....</i>	<i>25</i>
<i>Arp.....</i>	<i>26</i>
<i>Ask.....</i>	<i>27</i>
<i>Assign.....</i>	<i>27</i>
<i>Avail.....</i>	<i>28</i>

<i>Beep</i>	28
<i>BindDrivers</i>	29
<i>Break</i>	29
<i>CD</i>	30
<i>ChangeTaskPri</i>	30
<i>Clip</i>	31
<i>Conclip</i>	32
<i>Copy</i>	32
<i>CopyToPAR</i>	35
<i>Cut</i>	36
<i>Date</i>	37
<i>Debug</i>	37
<i>Decoration</i>	37
<i>Delay (now deprecated)</i>	38
<i>Delete</i>	38
<i>DevList</i>	39
<i>Dhclient</i>	39
<i>Dir</i>	40
<i>Diskchange</i>	41
<i>Echo</i>	41
<i>Eject</i>	42
<i>Else</i>	42
<i>Endcli / EndShell</i>	43
<i>Endif</i>	43
<i>Endskip</i>	43
<i>Eval</i>	44
<i>Execute</i>	45
<i>Failat</i>	45
<i>Fault</i>	46
<i>Filenote</i>	46
<i>Format</i>	47
<i>Get</i>	48
<i>Getenv</i>	48
<i>GfxControl</i>	49
<i>Grep</i>	49
<i>Guru</i>	50
<i>Hostname</i>	51
<i>Iconx</i>	51
<i>Identify</i>	52
<i>If</i>	52
<i>Ifconfig</i>	53
<i>Info</i>	56
<i>Iprefs</i>	57
<i>Install</i>	57
<i>Install-grub2-i386-pc</i>	57
<i>Install-i386-pc (now deprecated)</i>	58
<i>Join</i>	59
<i>Lab</i>	59
<i>Lha</i>	60
<i>LibList</i>	60
<i>List</i>	61
<i>Load</i>	62
<i>Loadresource</i>	62
<i>Lock</i>	63
<i>Makedir</i>	64
<i>Makelink</i>	64

<i>Mount.....</i>	<i>65</i>
<i>Newshell.....</i>	<i>65</i>
<i>Open.....</i>	<i>66</i>
<i>Partition.....</i>	<i>67</i>
<i>Path.....</i>	<i>68</i>
<i>PathPart.....</i>	<i>69</i>
<i>Ping.....</i>	<i>70</i>
<i>Pipe.....</i>	<i>71</i>
<i>Play.....</i>	<i>72</i>
<i>Prompt.....</i>	<i>72</i>
<i>Protect.....</i>	<i>73</i>
<i>PsdDevLister.....</i>	<i>74</i>
<i>PsdErrorLog</i>	<i>74</i>
<i>PsdStackLoader.....</i>	<i>75</i>
<i>Quit.....</i>	<i>75</i>
<i>Quitaros.....</i>	<i>75</i>
<i>Reboot.....</i>	<i>75</i>
<i>Relabel.....</i>	<i>76</i>
<i>Rename.....</i>	<i>76</i>
<i>RequestChoice.....</i>	<i>77</i>
<i>RequestFile.....</i>	<i>78</i>
<i>RequestString.....</i>	<i>78</i>
<i>Resident.....</i>	<i>79</i>
<i>ResList.....</i>	<i>79</i>
<i>Resolve.....</i>	<i>80</i>
<i>Route.....</i>	<i>80</i>
<i>Run.....</i>	<i>82</i>
<i>Search.....</i>	<i>82</i>
<i>Set.....</i>	<i>83</i>
<i>Setcache.....</i>	<i>84</i>
<i>SetClock.....</i>	<i>84</i>
<i>SetDefaultFont.....</i>	<i>85</i>
<i>SetDate.....</i>	<i>85</i>
<i>Setenv.....</i>	<i>86</i>
<i>SetKeyboard.....</i>	<i>86</i>
<i>SFSformat.....</i>	<i>87</i>
<i>SFSquery.....</i>	<i>87</i>
<i>Shell.....</i>	<i>88</i>
<i>Shutdown.....</i>	<i>89</i>
<i>Skip.....</i>	<i>89</i>
<i>Sort.....</i>	<i>90</i>
<i>SortCopy.....</i>	<i>90</i>
<i>Stack.....</i>	<i>91</i>
<i>Status.....</i>	<i>92</i>
<i>Type.....</i>	<i>92</i>
<i>UnACE.....</i>	<i>93</i>
<i>Unalias.....</i>	<i>93</i>
<i>UnARJ.....</i>	<i>94</i>
<i>UnDMS.....</i>	<i>94</i>
<i>UnLZX.....</i>	<i>94</i>
<i>Unpack.....</i>	<i>95</i>
<i>Unset.....</i>	<i>95</i>
<i>Unsetenv.....</i>	<i>96</i>
<i>UnShar.....</i>	<i>96</i>
<i>UnZip.....</i>	<i>96</i>
<i>Version.....</i>	<i>97</i>

<i>Wait.....</i>	<i>98</i>
<i>WaitX.....</i>	<i>98</i>
<i>Which.....</i>	<i>99</i>
<i>Why.....</i>	<i>99</i>

POSEIDON USB DEVICES UTILITIES.....101

<i>DRadioTool.....</i>	<i>101</i>
<i>PencamTool</i>	<i>101</i>
<i>PowManTool</i>	<i>103</i>
<i>RocketTool</i>	<i>104</i>

APPENDIX I – PRINTER ESCAPE SEQUENCES (OR COMMANDS)....105

APPENDIX II – AROS-DOS ERROR MESSAGES.....106

HOW TO USE THIS MANUAL

Welcome

This manual is meant to get people used to AROS-DOS. It is for everybody interested in AROS, as it tries to provide information on AROS in different levels of expertise. We'll try to cover everything in depth, but in such a way that you don't need to learn what you don't want to learn.

It tries to cover multiple platform usage, with the exception of AFA (AROS for Amiga) although many issues covered here should apply.

First chapters on this manual are for you to get the hang on the usage of the AROS shell; they cover most basic and essential aspects of daily usage of the Shell and AROS-DOS commands.

Later chapters are intended mostly for script programming and consult purpose.

If you're familiar with Amiga and AmigaDOS itself, most of the subject covered here should not be novelty to you, but there's a lot of software specific subjects that are dealt here that might be of use to you.

Documentation Conventions

The following conventions are used through out this manual:

KEYWORDS	Keywords are displayed in all uppercase letters, however, the arguments are not case sensitive (unless stated otherwise).
<n>	Angle brackets enclose variable information that the user should address.
Courier	Text appearing in Courier font (black colour) represents information displayed on AROS screen
Courier	Text appearing in Courier font (blue colour) represents information displayed on host OS screen
Key1+Key2	Key combinations displayed with a + (plus) sign connecting them



Did you know...

This image regards some more curious aspect of AROS itself or related subject.



Warning:

This kind of image is intended to get your attention regarding some procedure or behaviour which might require caution from your part.



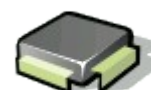
NOTE:

This image regards something that you should know in order to perform an operation on the best possible manner.



Example/Tutorial

This underlines an example of an integrated or complex operation performed under AROS.



Hardware related:

It pinpoints some important points regarding hardware support, behaviour or configuration.

indicate that pressing the keys simultaneously for a giving action or procedure.

Notice the side of this page it has some image with different purposes. The purposes are illustrated and described on that page.

Also on the side of the page might be the Icon program that the excerpt near it refers to, basically it acts as quick reference.

Other useful or reference documentation

You should also read the following manuals:

AROS User Manual

AROS Installation Manual

Also it's recommended that you read other more actively maintained documentation, which is kept in <http://www.aros.org/>.

INTRODUCTION

AROS-DOS features a CLI (Command Line Interface) which shares the same advantages and usage of the AmigaDOS one.

And like AmigaDOS it combines all the best aspects of Unix and DOS Command Line Interfaces, with some exclusive features.

Most of the commands are understandable at first reading, as well as the options passed as arguments.

"And still it provides conventions for handling arguments, and especially options, that dramatically increase ease-of-use. It provides a complete scripting environment that is so well integrated that scripts are indistinguishable from built-in commands - all the supporting functionality for built-in commands become instantly and fully available for a script, and a script can be used everywhere and in every way that a built-in command can be. Writing scripts for AmigaDOS (AROS-DOS) is a joy, and the results are usable, readable, and maintainable."

Furthermore, AmigaDOS provides an elegant means of managing program configuration. Each "shortcut" icon in the GUI can invoke a program, optionally with a document file. Nothing new there. But each icon can be easily configured to create a custom program "environment" using simple name/value pairs similar to Java property files and Windows INI files. There are no centralized registries or configuration databases, yet it is immensely usable." (Williams Robert D.)

So as you can see, AROS-DOS may play an important part in your work. And might just offer the necessary commands to complete your work – whether is simple or complex file management, network support, script programming... you might find that AROS-DOS does the job in an handsome manner.

Features

AROS Shell features the following aspects:

- Implicit change directory (Cd)
- Command execution history
- Completion help
- Simple file matching (Case unsensitive)
- Versatile pattern matching
- Softlinking and Hardlinking of files
- Multiple sessions (can have different background processes)
- Redirection and piping of command input / output
- Most commands work across different file-systems
- Powerful script programming



Did you know that AmigaOS was the first pre-emptive multi-tasking operating system for the "Personal Computer" concept?

Launched in 1985 the Amiga 1000 sported the AmigaOS which unlike it's 16-bit counterparts (IBM pc, Apple Mac and Atari st) was a true pre-emptive multi-tasking operating system taking advantage of the Custom Chip architecture.

The latest AmigaOS offering during the Commodore Amiga era was the 3.1 version in 1993 and offered a more stable and flexible OS that endured for many years and got different additions. This is regarded as the main fact that kept Amiga community active even after the Commodore demise.

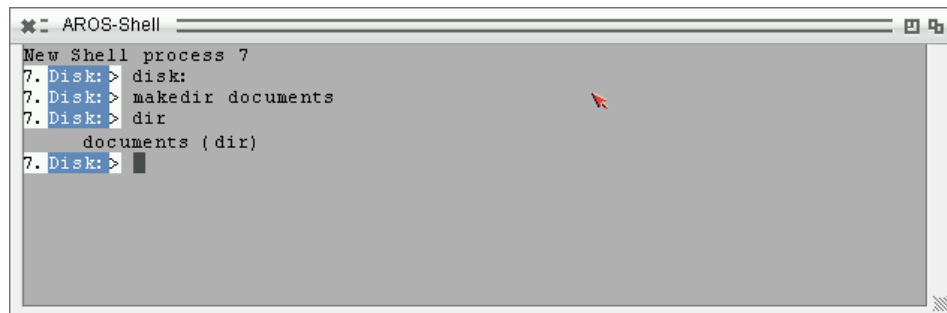


Warning:

READ ABOUT THE AROS
PUBLIC LICENSE (APL)
AND DISCLAIMER OF
WARRANTY ON PAGE 2.

AROS is alpha quality software. This means that it is currently mostly fun to play with and cool to develop for. If you came here because you thought AROS was a finished, complete and fully usable operating system, you will most likely be disappointed. AROS isn't there yet, but we're slowly moving in the right direction.

CHAPTER 1: AROS SHELL BASICS



```
AROS-Shell
New Shell process 7
7. Disk:> disk:
7. Disk:> mkdir documents
7. Disk:> dir
documents (dir)
7. Disk:>
```

AROS Shell

AROS has a command line interface, the 'Shell'. You can start it with menu Wanderer>Shell. A window with the path of the current AROS window will be opened. The prompt normally shows the number of the Shell and the current path.

The Shell has a command history. You can access commands with cursor up and down keys.

There is a completion feature. If you type the first letters of a command or a file name and press the tabulator key, the Shell searches for a matching name. If there are more possibilities a window will be opened where you can select an entry.

AROS commands and file names are case insensitive*. You can even use mixed case.

AROS searches commands in the current directory and in the search path. You can view and change the search path with the path command. Some important commands

- * CD -- changes directory (not really needed in AROS shell)
- * DIR -- shows content of directory
- * COPY -- copies files and directories
- * DELETE -- deletes files and directories
- * INFO -- shows available drives
- * MAKEDIR -- creates directory
- * RENAME -- renames files and directories
- * TYPE -- show content of text file

Path

A path is complete description of the location of a particular file on a disk device. When a program demands the name of a file for loading, it will specify the file's path, including the



(*) Warning:

Although commands will work in a case insensitive manner, that might not be the case for some file-systems other than FFS and FAT.

When working with Unix File-systems be aware that they are case sensitive and must be dealt appropriately.

Also SFS has a case-sensitive option possible to set on formatting the partition, if by some reason you require that also apply case sensitivity to your commands.

volume or device name and all the drawers that lead to the file.

Absolute paths start with the drive name and a colon (:), directories are separated with a slash (/). The drive name can be a device name (dh0:), a volume name (workbench:) or a logical drive (see assign command)

Example: `dh0:dir1/dir2/file.dat`

Do bear in mind that the device name and the volume name are different ways of identifying a given disk. For most purposes using either the name or the device id will grant access to the same path (either from shell or a file-requestor).

For example if you have a CD-ROM in device cd0: (probably the first CD-ROM) with the volume name "My-CD", you can reference it as either "cd0:" or "My-CD:". Although referencing as it's volume name with no CD present in CD drive will cause AROS to pop a requestor asking the user to insert the "My-CD" in an available drive.

If you need the current path as argument of a command, you can just write "".

Example: `copy from ram:x to ""`

A colon alone means base directory of the current path. When the path starts with a colon then it is relative to the base directory of the given path.

A slash at the beginning of a path means: go one level up. Two slashes mean go two levels up and so on.

When a path contains spaces it must be written within double quote characters.

Example: `type "name with spaces"`

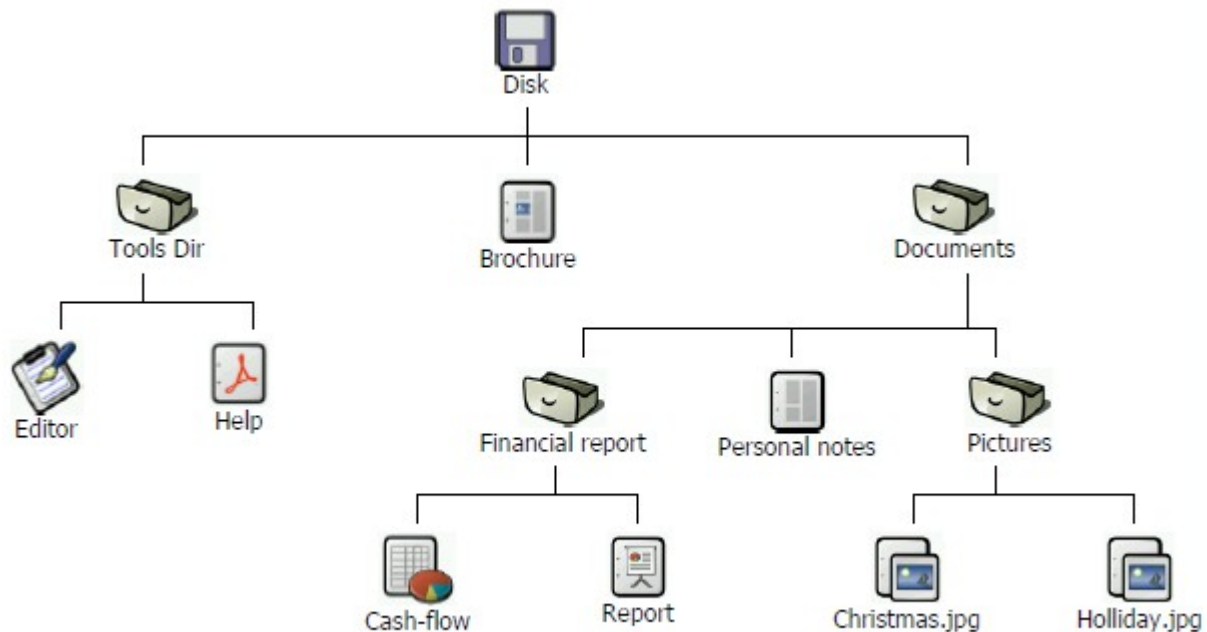
The following rules apply for AROS naming of files and drawers:

- Names can be up to 31 characters long in OFS/FFS devices and 107 in FFS / FAT devices...
- Colons (:) and slashes (/) are not allowed within a name. These characters are reserved for path statements.
- Upper and lower case differences (such as capitalization) are preserved and displayed by AROS. The system does not distinguish the difference, since it treats them in case-insensitive manner (upper case and lower case are considered the same at internal level of the OS).
- Duplicate file names are not allowed within the same drawer. If you save a file with the same name as an existing file in a drawer, it overwrites the original file in that drawer.
- The use of spaces must be handled with caution, since confusion might arise. Also for shell handling of space containing names, it must apply commas(",) at the beginning and end of the full path.

The icons that Wanderer uses to represent the files in a volume or a drawer are stored in

special .info files, with the name of the .info file matching the name of the file it represents. For example, the icon for Calculator, a simple calculator tools, is found in the file "Calculator.info".

So a basic drawer/file structure might be something like this:



Command template

A question mark after the command shows the parameter template of the command. The command is then in a mode where it waits for you to type in the parameters.

Example: `copy ?`

`FROM/M, TO/A, ALL/S, QUIET/S, BUF=BUFFER/K/N, CLONE/S, DATES/S, NOPRO/S, COM/S, NOREQ/S`

The keywords can have options:

`/A` -- argument must be given

`/K` -- keyword must be written when using this argument

`/S` -- switch; just write the keyword to access the switch

`/N` -- numerical argument

`/M` -- more than one argument is possible

`/F` -- rest of command line

`=` -- abbreviation; you can optionally use the abbreviation

When calling a command '=' can be used for distinct assignment between keyword and value:

Example: `copy from=a.dat to=b.dat`

Patterns

Some commands allow patterns for file name parameters:

? -- one arbitrary character
#? -- zero or more arbitrary characters
#x -- zero or more x
~ -- negation
| -- or
() -- group
[] -- range

Examples:

```
dir #?.info
dir #?~(.info)
dir a(b|c)d
dir [a-c]e
```

Redirection

> redirects output to file or device
>> redirects output and appends to file
< redirects input from file or device

Example: `dir >ram:a`

Pipe

If you want to forward the output of a command to another command you can use the pipe. You have to connect the commands with a | character. There must be at least one space before and after the |:

Example: `dir | othercommand`

But what if the second command wants to read the input from a file? The solution is to use the fake device 'in':

Example: `dir | more in:`

Another way of nesting commands is using backticks (`). You can forward the output of a command to another one just by including it into backticks.

Example: `cut ``date`` word 2`

Special devices

* ram: This is a area of the AROS's internal memory that is set up as a file storage device like a disk. Files and directories can be copied into RAM: for temporary storage.

But after a reset its content is lost.

* nil: if you don't want the messages of a command to be displayed in the window you can use the 'nil:' device. Example: delete #? >nil:

Running in background

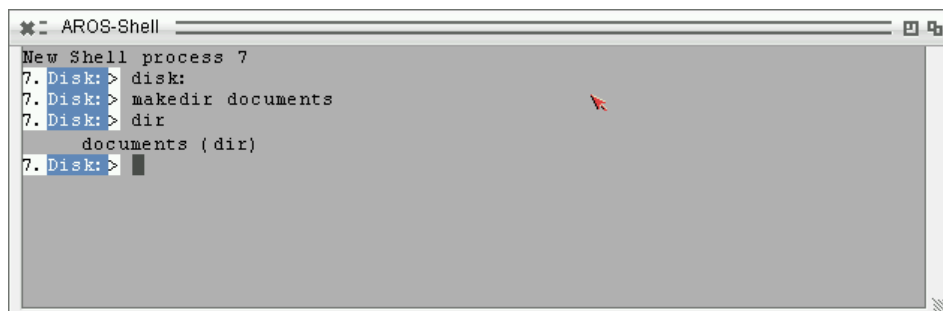
Normally, a command blocks the Shell until it is finished. You can run commands in the background with the run command.

Example: run delete #?

Info files

Files with the suffix '.info' play an important role in Wanderer. They contain the icon picture and some additional information. When you work with shell commands you have to take the Info files into account.

Basic file operation with Shell:



```
AROS-Shell
New Shell process 7
7.Disk:> disk:
7.Disk:> mkdir documents
7.Disk:> dir
      documents (dir)
7.Disk:>
```

Creating a directory

At the shell prompt window type the volume name followed by a colon (:) sign to access the disk.

For example: Disk:

Or you can use: cd Disk:

Then type the command mkdir <argument>, where the argument is the name of the drawer you wish to create.

For example: mkdir documents

You may check the current path contents with the dir command. Logically it will present your created directory.

Deleting a file

Type the Volume name followed by a colon sign to get to the disk intended.

To delete a file or drawer you must use the command "Delete <path/object>" – where path is the full path to the file or drawer intended to delete.

For example:

Disk:

```
Delete documents/myfile
```

Working with Smart File System (SFS)

In AROS you're given the choice to set up partitions with SFS filesystem. This is journaling file-system developed initially for Amiga machines. It's been ported to AROS by the AROS Development Team.

Although the SFS handles filenames big as 107 chars in size, some programs may not work with this kind of filename lengths (Directory Opus for one...). AROS-DOS commands all comply to the 107 char filename SFS limit, and AROS core apps should have no problem with that. But as recommendation it would better to use 31 char filenames whenever possible.

SFS case-sensitive filenames

SFS can be optionally set to case sensitive evaluation of file-names in its structure. The File-name case sensivity is set in SFSformat, and to set it you have to use the switch CASESENSITIVE. So by default it's not used.

But would you choose this option you'll have to use it accordingly.

So filenames like "Test.file" isn't the same as "TEST.file". They both can exist in the same directory and will only be accessed if typed (or passed) accordingly it's UPPER or lower case chars.

When copying to other file-systems devices, this can a problem, because if you have two files with only this kind of case-sensitiveness difference, the result may be the first file being overwritten by the second or some kind of error output.

Undelete files

If your SFS partition wasn't formatted with the NORECYCLED switch, then you'll have a special directory where all files the deleted files are automatically moved named .recycled.

Do bear in mind that SFS by default only keeps the last 360 files deleted in .recycled directory (any size as long as there was enough space on the drive when deleted).

To gain access to the undelete dir open a shell and type the name of the partition where the file(s) was placed, followed by a colon (:), and press enter.

Now type `.recycled` to access the dir.

You can now restore any file that happens to be there...

Example:

```
1.SYS:> dh1:
1.SYS:> .recycled
1.SYS:.recycled> dir
  myfile.text
```

To restore this file just copy (or move) it back to where it belongs.

```
1.SYS:.recycled> copy myfile.text dh1:documents/
```

If the SHOWRECYCLED switch was used you may use wanderer or another file-manager like Directory Opus.

Minding your disk space

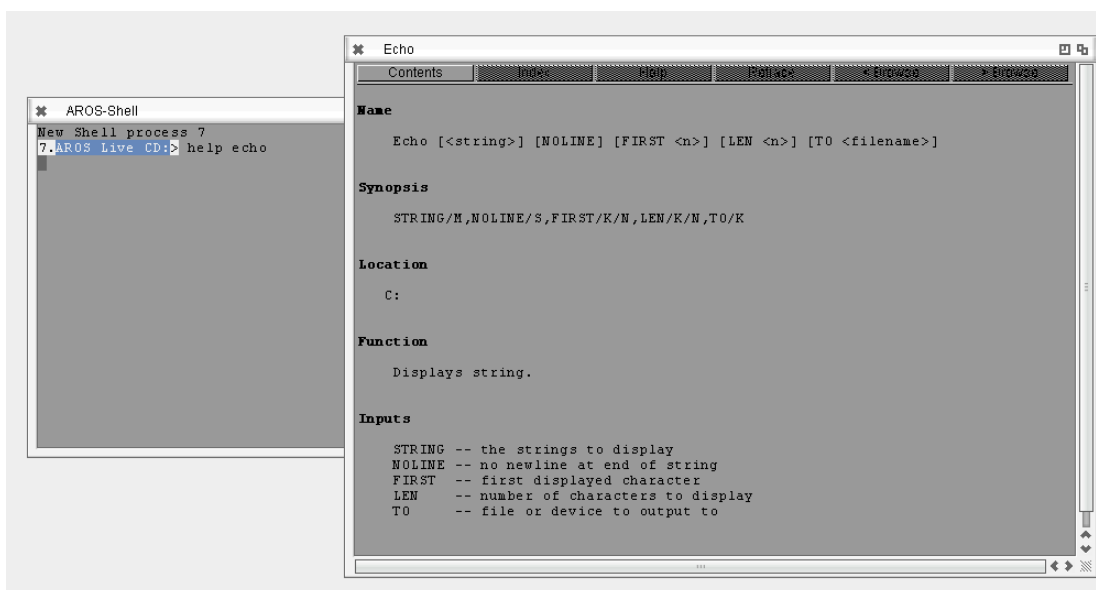
Every deleted file it will go into the .recycled dir and as for command output results deleted. Of course the files in .recycled will still take space on the drive – although you might have info otherwise.

So you'll have to delete files in .recycled regularly (specially big files like emulation disk images and ISO's) so that you always have free space as displayed by the relevant commands.

Do bear in mind that if you're going to use delete a lot of times (specially with big files), bear in mind that the device will perform a more slow manner – because it has to put data back and forth to. For that kind of situation there's the NORECYCLED switch of SFSformat.

The 'Help' utility

Recent nightly builds provide a new utility called 'help', which shows manual for most AROS commands in amiga-guide format. Just type help <command> in a shell to have its synopsis and template shown on the screen.



Script Programming

AROS-DOS Scripting

AROS-DOS Scripting languages allows you to accomplish simple tasks with a minimum amount of code. You can write a program that prints "Hello, world" to a window shell using just:

```
Echo "Hello World"
```

While it generally takes three to ten times as much code in a language like C, C++, or Java, and requires you to have some level of expertise on the given language for that.

Of course AROS-DOS scripting language is interpreted rather than compiled, so speed is limited to the disk access of the script files and the command invoked.

Scripting languages are also likely to have features that make them dynamic. For example, this might mean being able to generate code at run time and execute the new code within the program that generated it. Also, it's easy to make variables spring into existence when needed.

They may also provide some form of data interchange between applications or even control them to some degree.

Working with a script file

The script file is simply a text file that includes a number of commands that are executed – something similar to the batch files on the MS-DOS.

This covers a basic tutorial of what you could do.

Editing

The first thing you have to do is load a text editor such as EDITOR in tools. You can save time by calling it from shell and typing:

```
Editor RAM:myScriptfile
```

The program will now load.

In the EDITOR window type:

```
Echo "Hello WORLD! This script file lists all the assigns  
performed."  
Assign  
ENDCLI
```

Now select SAVE from the pull-down menus. The file should now be save to ram.

Executing from shell

Taking the example before – open the shell (If not already) and type the following into the NewShell window so your script file can be executed.

```
Execute RAM:myScriptfile
```

It will present all assigns present in the system and return you to the shell prompt.

Executing from wanderer

For running the script file from the wanderer app, you need to add a project icon file to it. Taking on the ongoing example, you can do this in the following manner (type in shell window):

```
Copy env:sys/def_project.info as RAM:myScriptfile.info
```

Now, you'll go to the Wanderer screen and open the Ram Disk window. Click once on the scriptfile icon in RAM and select INFORMATION from the pull-down menus. A information window will present the info on the icon selected. Click on default Tool and type in C:ICONX in text gadget. Now exit by saving and double-click on the icon.

You will find that it will now be executed without going through the Shell.

Command type execution*

Type in a Shell window: Protect RAM:myScriptfile +s

This will add what is called a protection flag – this specific flag sets the bit S (which means script). Now when you enter Ram:myScriptfile from the Shell you will get a list of all the assigns that you have.

Don't forget to call this script file it will have to be in the path listing or else you'll have to use always the full path.

Also if you wish to execute a script file from a script, you still have to use the command execute.



(*) NOTE:

The s:user-startup file may not exist on first boot, but it can be created.

The s:startup-sequence file will only execute this file if it exists.

Changing system script files

Like we said before, the script file is simply a text file that includes a number of commands that are executed – and some of these text files are actually system script files.

Three of the most important files, which reside in Sys:s/ directory, are:

- **startup-sequence** (the first file of AROS executed)
- **user-startup** (a file specially for user editing and customizing)

- **shell-startup** (the startup file executed whenever a shell window is opened)

It must be noted that while the Sys:s/ was intended initially for AROS-DOS script files, it may also house rexx script files. By default the REXX: dir is assigned to S: drawer.

If you wish to set a different drawer just edit the following lines in the S:startup-sequence:

```
If EXISTS C:RexxMast
    Assign REXX: S:
    Run <NIL: >NIL: C:RexxMast
EndIf
```

Just change the S: to whenever you wish to have your rexx script files (example: Sys:rexx/ - Dir must be created first if it does not exist!).

Unless you're an experienced person on the use of AROS script language (or AmigaDOS), we recommend you to leave the files as they are. Since you can loose the ability to boot AROS or have some significant part of it unusable.

AROS command Reference

For a better understanding of the various command purposes it's helpful if we divide them by some sort of domain application. The Group of commands are then divided like this:

Disk devices Oriented commands

AddBuffers	Info	Relabel
Assign	Install	SetCache
DiskChange	install-i386-pc	SFSdefragmentGUI
Format	Lock	SFSformat
HDTTool	Partition	SFSquery

Files / Directories oriented commands

CD	List	Search
Copy	MakeDir	SetDate
Delete	MakeLink	Sort
Dir	Open	Type
Filenote	Path	Unpack
Identify	Protect	Version
Join	Rename	

Shell / CLI oriented commands

Alias	NewShell	Status
ConClip	Prompt	Unalias
Echo	Resident	Which
EndCli	Run	Why
Execute	Shell	
IconX	Stack	

Script programming oriented commands

Ask	Fault	RequestFile
Beep	Get	Set
Echo	Getenv	Setenv
Else	If	Skip
EndIf	Lab	Unset
EndSkip	Quit	Unsetenv
FailAt	RequestChoice	Wait

AROS system oriented commands

AddAudioModes	Decoration	SetClock
AddDatatypes	IPrefs	SetDefaultFont
Avail	LoadResource	SetKeyboard
ChangeTaskPri	Mount	SetLocale
Date	Reboot	

Miscellaneous oriented commands

Eval	QuitAROS	RexxMast
------	----------	----------

As for a simple description of these categories:

- The disk device oriented commands cover the basics of creation and data manipulation and information on the AROS supported storage volume.
- The file and drawer type commands cover from the ordinary file management to a more complex type commands which include file/dir soft linking and file hard linking.
- The shell commands are the ones that deal with shell works, display and process / command execution or termination
- The script programming commands are the ones that control AROS-DOS script execution flow, variable handling and evaluation, user input, as well as script termination conditions...
- The system commands are subsets of the Core system, which may be handled manually by the user using the shell. Loading and setting devices, resources as well providing basic system specific information.

It's common for third-party commands to be also included in system directories such as c: (Sys:c/). The digital documentation which may accompany these should be copied into the help: (Sys:Locale/Catalogs) – This manual covers only the official AROS shell commands.

There's also a set of commands present in the C: which is more developer oriented, so they're not covered in this documentation. The commands include:

Developer oriented commands		
CheckMem	DevList	ResList
CRList	LeakWatch	TaskList
DamageList	LibList	Stacksnoop

In this chapter we present the full list of user available shell commands, alphabetically organized. The list includes the command description, usage and synopsis, as well as input detail, examples on use, notes and associated commands.

About file locations

This manual now covers also many CLI utilities which aren't strictly to be considered as "AROS commands", but are part of components commonly distributed with AROS itself, like AROSTCP or Poseidon USB Stack, or are ports from 68K Amiga™ or Unix world, like compression/decompression software and other stuff. They are included in nightly builds and distributions as well, however their location may change between *nightlies* and distributions. In this manual, we will follow nightly build file location, but Icaros Desktop, AspireOS, AROS Vision and Broadway users may find them elsewhere, according to distributors' tastes.

You can notice them because they're not originally supposed to be included in SYS:C, even if your AROS distribution of choice has moved them there.

AddAudioModes

Usage: ADDAUDIOMODES <name> [QUIET] [REFRESH] [REMOVE] [DBLSCAN]

Synopsis	FILES/M, QUIET/S, REFRESH/S, REMOVE/S, DBLSCAN/S
Location	Sys:C
Function	Builds and removes audio modes which AHI can understand from a list included in Devs:AudioModes. Manually using this command is very unusual, since the AHI audio system does it automatically the first time it gets started. It may be useful, however, in installation scripts.
Inputs	FILES – specifies which descriptors must be added to the current audio mode list QUIET – suppresses error and output messages REFRESH – scans Devs:AudioModes and adds all descriptors found to the current list DBLSCAN – if specified, it will open and immediately after close a native, double-scan screen. On some systems using a graphic card, it will enable sample frequencies greater than 28 kHz with native audio. An appropriate monitor driver in Devs:Monitors is needed to make it work.
Examples	<pre>1.sys> AddAudioModes EMU10KX</pre> <p>Adds the EMU10KX audio mode to the current modes</p>

AddBuffers

Usage: AddBuffers (drive) [(N)]

Synopsis	DRIVE/A, BUFFERS/N
Location	Sys:C
Function	Add buffers to the list of available buffers for a specific drive. Adding buffers speeds disk access but has the drawback of using up system memory (512 bytes per buffer). Specifying a negative number subtracts buffers from the drive. If only the DRIVE argument is specified, the number of buffers for that drive are displayed without changing the buffer allocation.
Inputs	DRIVE -- the drive to alter the buffer allocation of BUFFERS -- the number of buffers to add (or subtract in case of a negative number) to a drive.

AddDatatypes

Usage: `AddDataTypes files/M QUIET/S REFRESH/S`

Synopsis `FILES/M, QUIET/S, REFRESH/S, LIST/S`

Location `Sys:C`

Function AddDatatypes allows you to activate a set of specific datatypes. This might be necessary if new datatypes were installed on your system or were not activated on startup.

Inputs `FILES` - The name of the file(s) of the corresponding datatype.
 `QUIET` - Won't output any messages
 `REFRESH` – Reread existing datatypes
 `LSIT` – This will display a list of current datatypes loaded in memory

Result Standard DOS error codes.

Example: `AddDataTypes gif.datatype REFRESH`

AddUSBClasses

Usage: `AddUSBclasses [QUIET] [REMOVE]`

Synopsis `QUIET/S,REMOVE/S`

Location `Sys:C`

Function Adds or removes USB device classes for Poseidon USB stack.

Inputs `QUIET` – do not show error and output messages
 `REMOVE` – removes a USB class from the current list

See also `AddUSBHardware`

AddUSBHardware

Usage: `AddUSBHardware <device> [UNIT <unit>] [QUIET] [REMOVE] [ALL]`

Synopsis `DEVICE,UNIT/N,QUIET/S,REMOVE/S,ALL/S`

Location `Sys:C`

Function Adds or removes USB device hardware from Poseidon USB stack.

After reading the hardware driver, a class scan is done automatically.

Inputs **DEVICE** – required argument, containing the absolute path and the USB device driver to add or remove
UNIT – optional unit number, if multiple units are supported.
 Default unit is 0
QUIET – do not show error and output messages
REMOVE – removes a USB device driver from the current list.
 To remove hardware, the exact same path and name used for adding it must be used with REMOVE as well
ALL – tries to add all units of the given device. If REMOVE is specified, it removes all entries, effectively putting the stack offline.

See also AddUSBClasses

Alias

Usage: `Alias`

Synopsis `NAME,STRING/F`

Location `Sys:C`

Function Alias allows you to create an alternate name for other DOS commands.
 If Alias is used with no parameters, it will display the current list of Aliases defined within the current shell.

 Using a pair of square brackets within an alias allows you to provide the 'new' dos command with parameters.

Inputs `NAME` - The name of the alias to set.
 `STRING` - The value of the alias `NAME`.

Result Standard DOS error codes.

Example: `Alias DF "Type [] number"`

 By typing "DF S:Shell-Startup" in the shell, you are actually executing the command "Type S:Shell-Startup number". This will display the contents of the S:Shell-Startup file in the shell with line numbers on the left hand side.

See also Unalias

Arp

Usage: `arp <command> <values> [parameters]`

Synopsis	<code>arp hostname</code> <code>arp -a [netname hostname]</code> <code>arp -d hostname</code> <code>arp -s hostname address [temp] [pub]</code> <code>arp -f filename</code>
Location	<code>Sys:system/network/AROSTCP/C</code>
Function	Arp displays and modifies the Internet to hardware address translation tables used by the Address Resolution Protocol. The hardware address is a hexadecimal string with each octet separated by a colon, for instance 0:12:ff:a. The length of the address must be correct for the specified interface.
Inputs	<p>None</p> <p>If no options are specified (first form above), arp displays the current ARP entry for hostname. The hostname must either appear in the hostname database (SEE hosts), or be a DARPA Internet address expressed in Internet standard "dot notation". Hostname can also be resolved by nameserver.</p> <p>-a</p> <p>Display all current ARP entries by reading the address mapping table of the specified (sub)network. 'Hostname' is used to as default network specifier.</p> <p>-d</p> <p>If an ARP entry exists for the host called hostname, delete it. [This requires super-user privileges.]</p> <p>-s</p> <p>Create an ARP entry for the host called hostname with the hardware station address address. The hardware station address is given as hexadecimal bytes separated by colons. If an ARP entry already exists for hostname, the existing entry is updated with the new information. The entry is permanent unless the word temp is given in the command. If the word pub is specified, the entry is published, which means that this system will act as an ARP server responding to requests for hostname even though the host address is not its own.</p> <p>-f</p> <p>Read file filename and set multiple entries in the ARP tables. Entries in the file should be of the form:</p> <pre>hostname address [temp] [pub]</pre> <p>Argument meanings are the same as for the -s option.</p>

Ask

Usage: Ask <prompt>

Synopsis	PROMPT/A
Location	Sys:C
Function	Prompts the user for an input. Possible inputs are y for yes and n or Return for no. Selecting y sets the return code to 5.
Inputs	PROMPT -- the string is displayed in the window
See also	RequestChoice

Assign

Usage: Assign [(name):] [{(target)}] [LIST] [EXISTS] [DISMOUNT] [DEFER]

Synopsis	NAME, TARGET/M, LIST/S, EXISTS/S, DISMOUNT/S, DEFER/S, PATH/S, ADD/S, REMOVE/S, VOLS/S, DIRS/S, DEVICES/S
Location	Sys:C
Function	<p>ASSIGN creates a reference to a file or directory. The reference is a logical device name which makes it very convenient to specify assigned objects using the reference instead of their paths.</p> <p>If the NAME and TARGET arguments are given, ASSIGN assigns the given logical name to the specified target. If the NAME given is already assigned to a file or directory the new target replaces the previous target. A colon must be included after the NAME argument.</p> <p>If only the NAME argument is given, any assigns to that NAME are removed. If no arguments whatsoever are given, all logical assigns are listed.</p>
Inputs	<p>NAME -- the name that should be assigned to a file or dir</p> <p>TARGET -- one or more files or directories to assign the NAME to</p> <p>LIST -- list all assigns made</p> <p>EXISTS -- if NAME is not assigned, set the condition flag to WARN</p> <p>DISMOUNT -- remove the volume or device NAME from the dos list</p> <p>DEFER -- make an ASSIGN to a path or directory that not need to exist at the time of assignment. The first time the</p>

NAME is referenced the NAME is bound to the object

PATH -- path to assign with a non-binding assign. This means that the assign is re-evaluated each time a reference to NAME is done. Like for DEFER, the path doesn't have to exist when the ASSIGN command is executed

ADD -- don't replace an assign but add another object for a NAME (multi-assigns)

REMOVE -- remove an ASSIGN

VOLS -- show assigned volumes if in LIST mode

DIRS -- show assigned directories if in LIST mode

DEVICES -- show assigned devices if in LIST mode

Avail

Usage: Avail [CHIP | FAST | TOTAL | FLUSH] [H | HUMAN]

Synopsis CHIP/S, FAST/S, TOTAL/S, FLUSH/S, H=HUMAN/S

Location Sys:C

Function Give a summary of the memory usage and availability in the system. To free up unused memory that still may be allocated (libraries, devices, fonts and such present in memory but which are currently not in use), use the FLUSH option.

Inputs

CHIP -- show only "chip" memory

FAST -- show only "fast" memory

TOTAL -- show information on memory regardless of type

FLUSH -- remove unnecessary things residing in memory

HUMAN --display more human-readable values (gigabytes as "G", megabytes as "M", kilobytes as "K")

Notes "Chip" and "fast" memory are associated with the Amiga computer and may not be applicable on your hardware platform.

Beep

Usage: Beep

Synopsis N/A

Location Sys:C

Function BEEP produces a beep via Intuition DisplayBeep(NULL).

BindDrivers

Usage: BindDrivers [DEVICES|DRIVERS|DIR]

Synopsis DEVICES/S,DRIVERS/S,DIR/K/A

Location Sys:C

Function For all device drivers with a .info file in SYS:Expansion, load the device driver via Exec/InitResident() if its PRODUCT=tooltype matches a device that is in the system, and not yet configured.

Inputs DEVICES -- List all devices, and their bindings
 DRIVERS -- List all drivers, and their supported products
 DIR <directory> -- Directory to search, instead of SYS:Expansion/

Example C:BindDrivers

Break

Usage: Break <process> [ALL|C|D|E|F]

Synopsis PROCESS/N,PORT,ALL/S,C/S,D/S,E/S,F/S

Location Sys:C

Function BREAK sends one or more signals to a CLI process.
The argument |PROCESS| specifies the numeric ID of the CLI process that you wish to send the signal to. The STATUS command will list all currently running CLI processes along with their ID.
You can also specify a public port name and send signal's to the port's task.

You can send all signals at once via option ALL or any combination of the flags CTRL-C, CTRL-D, CTRL-E and CTRL-F by their respective options. When only the CLI process ID is specified the CTRL-C signal will be sent.

The effect of using the BREAK command is the same as selecting the console window of a process and pressing the relevant key combination.

The normal meaning of the keys is:

CTRL-C	-	Halt a process
CTRL-D	-	Halt a shell script
CTRL-E	-	Close a process' window
CTRL-F	-	Make active the process' window

Example 1.SYS:> BREAK 1

Send the CTRL-C signal to the process numbered 1.

1.SYS:> BREAK 4 E

Send the CTRL-E signal to the process numbered 4.

Notes Not all programs respond to these signals, however most should respond to CTRL-C.

CD

Usage: CD [DIR]

Synopsis DIR

Location Sys:C

Function Without argument it shows the name of the current directory.
With argument it changes the current directory.

Inputs DIR -- path to change to current directory

Notes AROS Shell does not require CD to actually change its path. The user can just type the path name and it will perform the same way.

ChangeTaskPri

Usage: ChangeTaskPri <priority> [PROCESS <process number>]

Synopsis PRI=PRIORITY/A/N,PROCESS/K/N

Location Sys:C

Function The ChangeTaskPri command is used to change the current run priority of a Task. As AROS is a multitasking operating system, you can determine which tasks receive more CPU time by changing their priorities.

The value of |priority| can be from -128 to 127, however values greater than 4 are not recommended as they can interfere with vital system processes. Higher values will give tasks a higher CPU priority.

You can use the Status command to examine the list of Tasks that are running and their process numbers.

Example 1.SYS:> ChangeTaskPri 1 Process 1

Set the priority of the current process to 1.

```
1.SYS:> ChangeTaskPri 1
```

Also sets the priority of the current process to 1.

See also Status

Clip

Usage: CLIP [COUNT] [UNIT <unit>] [GET [WAIT]] [SET [TEXT]]

Synopsis U=UNIT/N/K,W=WAIT/S,G=GET/S,P=PUT=S=SET/S,C=COUNT/S,
TEXT

Location Sys:C

Function Handle the clipboard's units (read or write text) from the Shell. It
can be used to store or retrieve text from the clipboard, or count
how many clips (units) are used.

Inputs GET -- retrieves text from a specified unit number (if supplied)
SET [TEXT] -- stores [TEXT] in a specified unit number (if supplied)
COUNT -- counts and displays the number of filled clipboard units
UNIT -- specifies the clipboard unit to use with GET and PUT
actions
WAIT -- used with the GET action, tells the command to wait for a
specified unit to be filled with data, then it will do the GET
action.

Example 1.sys> CLIP PUT Hello UNIT 2

Stores the string 'Hello' in clipboard unit 2

```
1.sys> CLIP UNIT 2
```

Displays the contents of clipboard unit 2

```
1.sys> CLIP COUNT
```

Counts how many clipboard actually contain data

```
1.sys> CLIP SET
```

Deletes the content of clipboard unit 0.

Notes If no GET, PUT or COUNT argument is specified, text will be
retrieved. GET will be the default action, while default unit will be 0
(if not expressed).

Conclip

Usage: CONCLIP [[UNIT | CLIPUNIT] <unitnumber>] [OFF]

Synopsis CLIPUNIT=UNIT/N, ON/S, OFF/S

Location Sys:C

Function Enable clipboard cut/copy/paste functionality in console windows and string gadgets. This enables the use of a system global clipboard.

Inputs CLIPUNIT=UNIT/N, ON/S, OFF/S

Example Conclip CLIPUNIT=1

This will set the global clipboard unit to 1. The available clipboards can be checked in clips: dir

Dir Clips:

Copy

Usage: Copy FROM TO [ALL] [QUIET) [BUF=n] [CLONE] [DATES] [NOPRO] [COM] [NOREQ]

Synopsis FROM/M, TO, ALL/S, QUIET/S, BUF=BUFFER/K/N, CLONE/S, DATES/S, NOPRO/S, COM=COMMENT/S, NOREQ/S,

Location Sys:C

Function Creates identical copies of one or more files.

Inputs FROM -- multiple input files
TO -- destination file or directory
ALL -- deep scan into sub directories
QUIET -- suppress all output and requesters
BUFFER -- buffer size for copy buffer in 512 byte blocks
(default 1024 (= 512K))
CLONE -- copy comment, protection bits and date as well
DATES -- copy dates
NOPRO -- do not copy protection bits
COMMENT -- copy file comment
NOREQ -- suppress requesters

PATTERN -- a pattern the filenames must match
DIRECT -- copy mode only: copy file without any tests or options

VERBOSE -- gives more output
ERRWARN -- do not proceed, when one file failed
MAKEDIR -- produce directories
MOVE -- delete source files after copying successful
DELETE -- do not copy, but delete the source files
HARDLINK -- make a hardlink to source instead of copying
SOFTLINK -- make a softlink to source instead of copying
FOLNK -- also makes links to directories
FODEL -- delete protected files also
FOOVR -- also overwrite protected files
DONT OVR -- do never overwrite destination
FORCE -- DO NOT USE. Call compatibility only.

Examples

'Copy DIRECT text PRT:'

This prints a file called text. - Copy manages a lot of such cases automatically, but maybe this option is needed sometimes.

Detailed info

FROM:

Source file(s). For directories, all contained files are source files. May have standard patterns.

TO:

Destination file or for multiple sources destination directory. Destination directories are created (including all needed parent directories).

ALL:

Scan directories recursively

QUIET:

Copy is completely silent here. Really no output is given, also no requests for missing disks or other problems!

BUF=BUFFER:

Specify the number of 512 byte buffers for copying. Default are 200 buffers [100KB memory]. One buffer is minimum size, but should never be used.

PAT=PATTERN:

PATTERN allows to specify a standard dos pattern, all file have to match. This is useful with ALL option.

CLONE:

The file comment, date and protection bits of the source files are copied to destination file or directory.

DATES:

The date information of source is copied to destination.

NOPRO:

The protection bits of sources are NOT copied. So the destination gets the default bits [rwd].

COM=COMMENT:

The file comment is copied to destination.

NOREQ:

No standard DOS requests are displayed, when an error occurs.

DIRECT:

Certain devices do not allow some of the used DOS packet request types. This option is a really easy copy command, which only opens source and destination directly without any tests and checks.

Options ALL, PAT, CLONE, DATES, NOPRO, COM, MAKEDIR, MOVE, DELETE, HARD, SOFT, FOLNK, FODEL, FOOVR, DONTTOVR and multiple input files cannot be specified together with DIRECT. This option needs one input and one output file.

When you want to delete a softlink, which does no longer point to a valid file, you need this option as well.

VERBOSE:

Copy gives additional output.

ERRWARN:

Copy knows and returns the 3 types of dos.library errors:

5 WARN - The processing of one file failed, Copy skips this file and proceeds the next.

10 ERROR - The creation of a directory or any other bad error happened. Copy quits after that.

20 FAIL - A really hard error happened (No memory, Examine failed, ...). Copy quits after that.

When option ERRWARN is used, the result 5 (WARN) gets result 10 (ERROR). So Copy aborts every time an error occurred.

MAKEDIR:

All names specified in FROM field are taken as directories, which must be created.

MOVE:

The files are not copied, but moved (or renamed). This means that after move operation the source does no longer exist.

DELETE:

This does not copy anything, but delete the source files!

HARD=HARDLINK:

Instead of copying the files, a hard link is created. This only works, when destination is on same device as source. When ALL option is specified, the directories are scanned recursively, else Copy produces links to the directories.

SOFT=SOFTLINK:

Instead of copying directories, a soft link is created. These links are useable between different devices also. Soft links are only created for directories. Files are skipped here. Option FORCELINK is therefore always set to true.

FOLNK=FORCELINK:

When linking of directories should be possible, this option is needed.

FODEL=FORCEDELETE:

When this option is enabled, files are deleted also, when they are delete protected.

FOOVR=FORCEOVERWRITE:

When this option is enabled, files are overwritten also, when they are protected.

DONTOVR=DONTOVERWRITE:

This option prevents overwriting of destination files.

Notes Softlinks are not official supported by OS and may be dangerous. I suggest not to use this option!
See section "About links" for possible problems.

See also Delete, Rename, MakeDir, MakeLink

CopyToPAR

Usage: CopyToPAR [<FILE>] [USB] [QUIET]

Synopsis FILE/A,USB/S,QUIET/S

Location Sys:C

Function Copies (or sends) a file to parallel.device or usbparallel.device.

Inputs FILE -- Either a file, a directory or a pattern to match.
USB -- Use usbparallel.device.
QUIET -- Suppresses any output to the shell.

Result Standard DOS return codes.

Cut

Usage: CUT <string> [CHAR <range> | WORD <range> [SEPARATOR <string>]]

Synopsis STRING/A,C=CHAR/K,W=WORD/K,S=SEPARATOR/K

Location Sys:C

Function Extracts some characters or full words from a string.

User can set a begin and an end position for both CHAR and WORD arguments: the former extracts characters, the latter a whole idiom. Words are separated by a selectable character called SEPARATOR. Default is space (" "), but a string can be used as well.

User may specify a position range with numbers, in the form "P1-P2". P1 is the first character/word to extract from the original string, "-" is the minus character while P2 is the last character/word to extract. If only a position is expressed, cut will extract just a single character/word, unless a hyphen-minus character is placed before or after the number: P- extracts everything placed in the string starting with the P position, while -P will extract everything from the beginning of the string stopping at the P position.

Examples

Example 1:

```
1.sys> Cut "A yellow flower" CHAR 8  
w  
extract one character.
```

Example 2:

```
1.sys> Cut "A yellow flower" CHAR 6-12  
low flo  
extract from character 6 to 12.
```

Example 3:

```
1.sys> Cut "A yellow flower" CHAR -8  
A yellow  
Extract from character 1 to 8 without specifying the beginning position.
```

Example 4:

```
1.sys> Cut "A yellow flower" CHAR 10-  
flower  
extract from character 10 of the string until the end.
```

Example 5:

```
1.sys> Cut "World of Commodore" WORD 2 SEPARATOR "o"  
mm  
extract the second word (using the user-defined separator "o").
```

Result Standard DOS return codes.

Date

Usage: `Date [<day>] [<date>] [<time>] [TO | VER <filename>]`

Synopsis `DAY,DATE,TIME,TO=VER/K`

Location `Sys:C`

Function Displays or sets the system date and/or time.

Inputs `DAY` -- sets date by name (Monday, Tuesday, ... , tomorrow, yesterday)
 `DATE` -- sets date in format DD-MMM-YY. For MMM either the number or the first 3 letters of the month in English.
 `TIME` -- sets time in format HH:MM:SS
 `TO` -- output is sent to file

Example `1.Sys:> date 2-feb-06`
 `1.Sys:> date 21:10`

Debug

Usage: `Debug`

Location `Sys:C`

Function Activates built-in AROS debugger (SAD)

Example `1.Sys:> debug`

Decoration

Usage: `Decoration`

Location `Sys:C`

Function Allows user definable skins for the intuition windows, menus and gadgets. It must be launched before Wanderer - usually in the S:startup-sequence

Notes See also `iprefs`

Delay (now deprecated)

Usage: Delay (n) [TICK | TICKS]

Synopsis TIME/N,TICK=TICKS/S

Location Sys:C

Function Wait a certain amount of ticks clock (1/50 of a second).

Inputs TICK=TICKS -- time unit in ticks to wait for.

Notes Delay has been removed from recent nightlies. Use wait/waitx instead.

Delete

Usage: Delete {(name | pattern)} [ALL] [QUIET] [FORCE]

Synopsis NAME/K, ALL, QUIET, FORCE

Location Sys:C

Function Deletes files and directories. You may delete several files and directories by listing them separately or by using wildcards. To abort a multiple delete, press CTRL-C. Delete will notify the user of which files it weren't able to delete.
Delete cannot delete directories which are not empty unless the ALL option is used. To suppress file and directory names from being printed while deleted use the QUIET option. If the 'd' protection bit is cleared for a file or directory, it may not be deleted unless the FORCE option is used.

Inputs FILE/M/A - files or directories to delete (may contain patterns)
ALL/S - recursively delete directories
QUIET/S - don't print which files/directories were deleted
FORCE/S - delete files/directories even if they are protected from deletion

Example 1.Sys:> Delete RAM:T/#? ALL FORCE

Deletes all directories and files recursively in the directory RAM:T even if they are protected from deletion.

DevList

Usage: DevList

Synopsis (N/A)

Location Sys:C

Function Gives a list of running devices: address, version, revision, opencnt, flags, name

Dhclient

Usage: dhclient [-p port] [-d] [-q] [-l] [-r] [-lf lease-file] [-pf pid-file] [-cf config-file] [-sf script-file] [-s server] [-g relay] [-n] [-nw] [-w] [interface]

Synopsis Unix-like command

Location Sys:system/network/AROSTCP/C

Function The Internet Software Consortium DHCP Client, dhclient, provides a means for configuring one or more network interfaces using the Dynamic Host Configuration Protocol, BOOTP protocol, or if these protocols fail, by statically assigning an address.

Inputs The names of the network interfaces that dhclient should attempt to configure may be specified on the command line. If no interface names are specified on the command line dhclient will normally identify all network interfaces, eliminating non-broadcast interfaces if possible, and attempt to configure each interface.

If the DHCP client should listen and transmit on a port other than the standard (port 68), the -p flag may be used. It should be followed by the udp port number that dhclient should use. This is mostly useful for debugging purposes. If a different port is specified for the client to listen on and transmit on, the client will also use a different destination port - one greater than the specified destination port.

The DHCP client normally transmits any protocol messages it sends before acquiring an IP address to, 255.255.255.255, the IP limited broadcast address. For debugging purposes, it may be useful to have the server transmit these messages to some other address. This can be specified with the -s flag, followed by the IP address or domain name of the destination.

For testing purposes, the giaddr field of all packets that the client sends can be set using the -g flag, followed by the IP address to

send. This is only useful for testing, and should not be expected to work in any consistent or useful way.

The DHCP client will normally run in the foreground until it has configured an interface, and then will revert to running in the background. To run force dhclient to always run as a foreground process, the -d flag should be specified. This is useful when running the client under a debugger, or when running it out of inittab on System V systems.

The -q flag prevents any messages other than errors from being printed to the standard error descriptor.

The client normally doesn't release the current lease as it is not required by the DHCP protocol. Some cable ISPs require their clients to notify the server if they wish to release an assigned IP address. The -r flag explicitly releases the current lease, and once the lease has been released, the client exits.

The -1 flag cause dhclient to try once to get a lease. If it fails, dhclient exits with exit code two.

The DHCP client normally exits if it isn't able to identify any network interfaces to configure. On laptop computers and other computers with hot-swappable I/O buses, it is possible that a broadcast interface may be added after system startup. The -w flag can be used to cause the client not to exit when it doesn't find any such interfaces. The DHCP client can be directed not to attempt to configure any interfaces using the -n flag. This is most likely to be useful in combination with the -w flag.

The client can also be instructed to become a daemon immediately, rather than waiting until it has acquired an IP address. This can be done by supplying the -nw flag.

Dir

Usage: Dir [(dir | pattern)] [OPT A | I | D | F] [ALL] [DIRS] [FILES] [INTER]

Synopsis DIR,OPT/K,ALL/S,DIRS/S,FILES/S,INTER/S

Location Sys:C

Function DIR displays the file or directory contained in the current or specified directory. Directories get listed first, then in alphabetical order, the files are listed in two columns. Pressing CTRL-C aborts the directory listing

Inputs	ALL -- Display all subdirectories and their files recursively. DIRS -- Display only directories. FILES -- Display only files. INTER -- Enter interactive mode.
Notes	Interactive listing mode stops after each name to display a question mark at which you can enter commands. These commands are: Return -- Go to the next file or directory. E/ENTER -- Enters a directory. DEL/DELETE -- Delete a file or an empty directory. C/COM -- Let the file or directory be the input of a DOS command (which specified after the C or COM or specified separately later). Q/QUIT -- Quit interactive mode. B/BACK -- Go back one directory level.

Diskchange

Usage: Diskchange DEVICE

Synopsis	DEVICE/A
Location	Sys:C
Function	Diskchange is intended only for removable media. For example if you have a internal floppy disk drive set to noclick AROS will not perform disk presence check. So you have to manually update any disk change made.
Inputs	DEVICE -- The disk device unit.
Example	Diskchange df0:

Echo

Usage: Echo [<string>] [NOLINE] [FIRST <n>] [LEN <n>] [TO <filename>]

Synopsis	STRING/M,NOLINE/S,FIRST/K/N,LEN/K/N,TO/K
Location	Sys:C
Function	Displays a string. The actual echo command is capable of displaying printer escape sequences to control text formatting in Shell, so *E is equivalent to ESC character.

Inputs STRING -- the strings to display
 NOLINE -- no newline at end of string
 FIRST -- first displayed character
 LEN -- number of characters to display
 TO -- file or device to output to

Example Echo " *E[0;0H*E[J* "

 This will clear the shell window (just like a pseudo cls command)

Eject

Usage: Eject <device>

Synopsis DEVICE/A

Location Sys:C

Function Ejects media from a device. This feature is not supported by all device types.

Inputs DEVICE -- Name of device to eject media from.

See also Load

Else

Usage: Else

Synopsis (none)

Location Sys:C

Function Separate the 'true' and 'false' blocks of an If statement. The block following an Else command is executed if the condition in the previous If statement was false.

Example If EXISTS Sys:Devs
 Copy random.device Sys:Devs/
 Else
 Echo "Cannot find Sys:Devs"
 EndIf

See also If, EndIf

Endcli / EndShell

Usage: Endcli or EndShell

Synopsis (none)

Location Sys:C

Function Terminates the current Command line interface session (shell process).

Endif

Usage: Endif

Synopsis (none)

Location Sys:C

Function Ends an If block. If the condition of the If command is false, execution will skip to the corresponding EndIf command, in case there isn't an Else command present.

Example

```
If EXISTS Sys:Devs
    Copy random.device Sys:Devs/
Else
    Echo "Cannot find Sys:Devs"
EndIf
```

See also If, Else

Endskip

Usage: Endskip

Synopsis (none)

Location Sys:C

Function Ends a skip command.

See also Skip

Eval

Usage: EVAL <value1> <op> <value2> [<TO>] [<Lformat>]

Synopsis VALUE1/A,OP,VALUE2/M,TO/K,LFORMAT/K

Location Sys:C

Function Evaluate an integer expression and print the result. The result is written to standard output if not the TO switch are used which instead prints the result to a file.
Using the switch LFORMAT, it is possible to direct how to write the result. Numbers prefixed by 0x or #x are interpreted as hexadecimal and those prefixed by # or 0 are interpreted as Octal. Alphabetical characters are indicated by a leading single quotation mark ('), and are evaluated as their ASCII equivalent.

Inputs VALUE1, OP, VALUE2 - The expression to evaluate. The following operators are supported:

Operator	Symbols
-----	-----
addition	+
subtraction	-
multiplication	*
division	/
modulo	mod, M, m, %
bitwise and	&
bitwise or	
bitwise not	~
left shift	lsh, L, l
right shift	rsh, R, r
negation	-
exclusive or	xor, X, x
bitwise equivalence	eqv, E, e

TO -- File to write the result to

LFORMAT -- printf-like specification of what to write.
The possible switches are:

%xd -- hexadecimal output, width digit d
%od -- octal output, width digit d
%n -- decimal output
%c -- character output (the ANSI-character
corresponding to the result value)

By specifying *n in the LFORMAT string, a newline is output.

Execute

Usage: `Execute <script> [{<arguments>}]`

Synopsis FILE/A
Location Sys:C
Function Executes a script with DOS commands.
Inputs FILE -- file to execute
See also Iconx

Failat

Usage: `FailAt <limit>`

Synopsis RCLIM/N
Location Sys:C
Function FailAt sets the return code limit of the current shell script. If any command returns with a failure code of this value or higher the script shall abort.
Common failure codes are:
 0 - No error
 5 - Warning
 10 - Error
 20 - Failure

The normal value for the return code limit is 10.

Example If we have a script with the commands

```
Copy RAM:SomeFile DF0:
Echo "Done!"
```

and the file RAM:SomeFile does not exist, the Copy command will return with:

```
Copy: object not found
Copy: returned with error code 20
```

and the script will abort. However if you include the command

```
FailAt 21
```

then the script will complete since the return code from Copy is less than the return code limit.

Fault

Usage: `Fault <error number>`

Synopsis `NUMBERS/N/M`

Location `Sys:C`

Function Fault prints the message corresponding with the error number supplied. Any number of error numbers can be given at once, but they must be separated by spaces.

Example `1.SYS:> Fault 205`
`Fault 205: object not found`

This tells you that the error code 205 means that a disk object could not be found.

Filenote

Usage: `Filenote <file> <comment> [ALL] [QUIET]`

Synopsis `FILE/A,COMMENT,ALL/S,QUIET/S`

Location `Sys:C`

Function Add a comment to a file or directory.

Filenote allows a recursive scan of all directories adding comments to each file/directory it finds that matches the file pattern specified.

Example `FILE` - Always has to be specified. Can be either a filename with a full path or a file pattern that is to be matched.

`COMMENT` - The ASCII string that is to be added as a comment to the file(s)/dir(s) specified.

To provide a comment that has embedded quotation marks, precede each quote with an asterisk.

I.e.: Filenote FILE=RAM:test.txt COMMENT="*hello*"

ALL - Boolean switch. If specified, Filenote scans the directories that match the pattern specified, recursively.

QUIET - Boolean switch. If specified, no diagnostic text will be displayed to standard output.

Result Standard DOS return codes.

Example Filenote ram: hello all

Recurse through each directory in RAM: adding "hello" as a filenote to each file/directory.

Notes Output from AROS' Filenote is more neat and structured than the standard Filenote command.

Does not yet support multi-assigns.

Format

Usage: Format DRIVE=<Drive> [NAME=<name>] [OFS | FFS] [INTL | NOINTL] [FORCE] [QUIET]

Synopsis DRIVE/K/A, NAME/K/A, OFS/S, FFS/S, INTL=INTERNATIONAL/S, NOINTL=NOINTERNATIONAL/S, FORCE/S, QUIET/S

Location Sys:C

Function Format will initialise a disk to be useable by the AROS.

Inputs DRIVE -- The device disk unit designation
NAME -- Sets the volume name.
OFS -- Formats the drive with the Old File system (not recommended for HD's).
FFS -- Formats the drive with the Fast File system.
INTL -- Enables the FFS support of international characters.
FORCE --
QUIET -- It will not display any output, ordinary or error messages.

Example Format DRIVE=DF0: NAME="MyDisk" FFS

This will format a disk present in the internal floppy drive with the name "Mydisk" and with Fast File System.

Notes This command only supports OFS and FFS file system.

See also SFSformat, Info

Get

Usage: `Get <Name>`

Synopsis NAME/A

Location Sys:C

Function Retrieves the information stored in the given local variable.

Inputs NAME - The name of the local variable.

Example `Get Result2`

 This will retrieve the secondary return code of the last command that was executed.

Result Standard DOS error codes.

See also Set, unset

Getenv

Usage: `Getenv <Name>`

Synopsis NAME/A

Location Sys:C

Function Retrieves the information stored in the given global variable.

Inputs NAME - The name of the local variable.

Example `1.sys> Getenv Extraspath`

`dh0:Extras`

 Retrieves the full path of Extras:, normally stored in the \$extraspah environment variable.

Result Standard DOS error codes.

See also Setenv, unsetenv

GfxControl

Usage: `GfxControl`

Synopsis	PREVENT_DIRECT_BITMAP_ACCESS=PDBA/S, ALLOW_DIRECT_BITMAP_ACCESS=ADBA/S,DUMP/S
Location	Sys:C
Function	Change some internal options of cybergraphics.library
Inputs	PREVENT_DIRECT_BITMAP_ACCESS -- Causes LockBitMapTagList() calls to always fail ALLOW_DIRECT_BITMAP_ACCESS -- Allow LockBitMapTagList() to go to gfx driver which may or may not support it. (default)
Result	DUMP -- Show current settings Standard DOS return codes.

Grep

Usage: `grep [FLAGS] <EXPRESSION> <FILE>`

Synopsis	Unix-like command
Location	Extras:Aminet/fish
Function	searches a file for a given pattern.
Inputs	Flags are single characters preceeded by '-': -c Only a count of matching lines is printed -f Print file name for matching lines switch, see below -n Each line is preceeded by its line number -v Only print non-matching lines

The file_list is a list of files (wildcards are acceptable on RSX modes).

The file name is normally printed if there is a file given.

The -f flag reverses this action (print name no file, not if more).

The regular_expression defines the pattern to search for. Upper- and lower-case are always ignored. Blank lines never match. The expression should be quoted to prevent file-name translation.

x An ordinary character (not mentioned below) matches that

character.

'\' The backslash quotes any character. "\\$" matches a dollar-sign.

'^' A circumflex at the beginning of an expression matches the beginning of a line.

'\$' A dollar-sign at the end of an expression matches the end of a line.

'.' A period matches any character except "new-line".

':a' A colon matches a class of characters described by the following

':d' character. ":a" matches any alphabetic, ":d" matches digits,

':n' ":n" matches alphanumerics, ": " matches spaces, tabs, and

': ' other control characters, such as new-line.

'*' An expression followed by an asterisk matches zero or more occurrences of that expression: "fo*" matches "f", "fo" "foo", etc.

'+' An expression followed by a plus sign matches one or more occurrences of that expression: "fo+" matches "fo", etc.

'-' An expression followed by a minus sign optionally matches the expression.

'[]' A string enclosed in square brackets matches any character in that string, but no others. If the first character in the string is a circumflex, the expression matches any character except "new-line" and the characters in the string. For example, "[xyz]" matches "xx" and "zyx", while "[^xyz]" matches "abc" but not "axb". A range of characters may be specified by two characters separated by "-". Note that, [a-z] matches alphabetics, while [z-a] never matches.

The concatenation of regular expressions is a regular expression.

Example `1.sys> list | grep Device`

Searches for the "Device" word in the output of List command. If exists, it show only the occurring line.

Guru

Usage: `Guru <Error code> [LASTALERT]`

Synopsis	GURU,L=LASTALERT/S
Location	Sys:C
Function	Retrieves the information about system error messages, or "Guru meditations".
Inputs	GURU – error code to interpretate
Example	<pre>1.sys> guru 33652234 Alert Code: 33652234 Type: Recoverable Subsystem: gadtools.library General: Cannot open resource Specified: Unknown</pre>

Hostname

Usage: `hostname [-s=SHORT]`

Synopsis	Unix-like command
Location	Sys:system/network/AROSTCP/C
Function	Hostname prints the current network name of the system, in the form hostname.domain.
Inputs	-s Trims off any domain information from the printed name.
Example	<pre>1.sys> hostname arosbox.arosnet</pre>

Iconx

Usage: (From Wanderer)

Synopsis	FILE/A
Location	Sys:C
Function	<p>Enables a script to be run from Wanderer.</p> <p>The script must have a Project .info file and the Default Tool set to "C:IconX" to run it.</p> <p>When double-clicked the script file is executed in a shell window opened for this sole purpose.</p>

Inputs	FILE - The script filename to execute.
	Tooltypes for script icon: WINDOW -- Specification of the shell window default: con:0/50//80/IconX/Auto STACK=n -- default: 40960 USERSHELL=YES NO -- default: YES WAIT=n -- Wait n seconds before closing window (default 2) DELAY=n -- Wait n/50 seconds before closing window
See also	Execute

Identify

Usage: `Identify <file|dir> [VERBOSE]`

Synopsis	FILE/M/A, VERBOSE/S
Location	Sys:C
Function	Identifies the file type or directory.
Inputs	FILE -- file to be recognized VERBOSE -- activates verbose output
Example	<pre>Identify s:startup-sequence</pre> <pre>S:startup-sequence Text/Ascii</pre> <p>It will identify the startup-sequence as a text file.</p>

If

Usage: `If <condition>`

Synopsis	NOT/S,WARN/S,ERROR/S,FAIL/S,,EQ/K,GT/K,GE/K,VAL/S,EXISTS/K
Location	Sys:C
Function	Carry out all the commands in a block if a given conditional is true. (A block is a run of command lines ended with an Else or EndIf command.) For every If command there must be a corresponding EndIf. If the condition is false, command execution will skip to the corresponding Else of EndIf command.
Inputs	NOT -- Negates the value of the condition

WARN -- True if the previous return code was greater than or equal to 5.
 ERROR -- True if the previous return code was greater than or equal to 10.
 FAIL -- True if the previous return code was greater than or equal to 20.
 EQ, GE, GT -- True if the first value is equal, greater than or equal respectively greater than the second.
 VAL -- Indicate that the comparison should treat the strings as numerical values.
 EXISTS <string> -- True if the file or directory <string> exists.

Example If 500 GT 200 VAL
 echo "500 is greater than 200"
 Else
 If EXISTS S:User-Startup
 echo "User-Startup script found in S:"
 Execute S:User-Startup
 EndIf
 EndIf

Notes ERROR and FAIL will only be appropriate if the fail level of the script is set via FailAt (the standard fail level is 10 and if any return code exceeds or equals this value, the script will be aborted).

See also Else, EndIf, FailAt

Ifconfig

Usage: ifconfig interface address_family [address [dest_address]] [params]
 ifconfig interface [address_family]

Synopsis Unix-like command

Location Sys:system/network/AROSTCP/C

Function Ifconfig is used to assign an address to a network interface and/or configure network interface parameters. ifconfig must be used at boot time to define the network address of each interface present on a machine. It can also be used at other times to redefine an interface's address or other operating parameters.

The command:

```
ifconfig interface/unit
```

with no optional command arguments supplied displays the current configuration for interface. If address_family is specified, ifconfig reports only the details specific to that address family.

Inputs

interface

A string of the interface name concatenated with unit number, for example ``eth0'`. The AmiTCP/IP network interfaces are defined in the ``AmiTCP:db/interface'` file. For example, a interface `sl` corresponds by default to ``Devs:networks/rhcslip.device'`.

address_family

Name of protocol on which naming scheme is based. An interface can receive transmissions in differing protocols, each of which may require separate naming schemes. Therefore, it is necessary to specify the `address_family`, which may affect interpretation of the remaining parameters on the command line. The only address family currently supported is `inet` (DARPA-Internet family).

Address

Either a host name present in the host name database, (SEE `hosts`), or a DARPA Internet address expressed in Internet standard "dot notation". The host number can be omitted on 10-Mbyte/second Ethernet interfaces (which use the hardware physical address), and on interfaces other than the first.

dest_address

Address of destination system. Consists of either a host name present in the host name database, `hosts(4)`, or a DARPA Internet address expressed in Internet standard "dot notation".

SWITCHES

The following operating parameters can be specified:

up

Mark an interface "up". Enables interface after an "ifconfig down." Occurs automatically when setting the address on an interface. Setting this flag has no effect if the hardware is "down".

Down

Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

arp

Enable the use of Address Resolution Protocol in mapping between network level addresses and link-level

addresses (default).

-arp

Disable the use of Address Resolution Protocol.

metric n

Set the routing metric of the interface to n, default 0. The routing metric is used by the routing protocol (see gated). Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.

Debug

Enable driver-dependent debugging code. This usually turns on extra console error logging.

-debug

Disable driver-dependent debugging code.

netmask mask

(Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks. mask includes the network part of the local address, and the subnet part which is taken from the host field of the address. mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the file AmiTCP:db/networks. `mask' contains 1's for each bit position in the 32-bit address that are to be used for the network and subnet parts, and 0's for the host part. mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

Broadcast

(Inet only) Specify the address that represents broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

Example

```
1.sys> ifconfig -a
```

```
lo0: flags=9<UP,LOOPBACK> mtu 1536
      inet 127.0.0.1 netmask 0xff000000
net0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX> mtu 1500
      address: 00:0c:29:10:ac:40
      inet 192.168.163.129 netmask 0xffffffff0 broadcast
192.168.163.255
```

Shows current status of all network cards

```
1.sys> ifconfig lo0 127.0.0.1
```

This command marks internal loopback device "UP", and attach an inet address 127.0.0.1 to it.

```
1.sys> ifconfig cslip0 inet 193.102.4.144
193.102.4.129
```

This command starts the CSLIP driver, attach an address 193.102.4.144 (our internet address) and a destination address 193.102.4.129 (the internet address of the host you are connecting) to it.

```
1.sys> ifconfig eth0 inet 193.124.100.64 netmask
255.255.255.192 -arp
```

This command loads an ethernet driver, marks it "up", disables ARP protocol for it, attaches an inet address 193.124.100.65 to it and sets its netmask to 255.255.255.192. A bitwise logical and of netmask and address for the interface forms a subnet address, in this case 193.124.100.64. All packets aimed to hosts with same subnet address (that is hosts 193.124.100.64 - 193.124.100.127) are routed to this interface.

Info

Usage: Info [DISKS] [VOLS] [GOODONLY] [BLOCKS] [DEVICES]

Synopsis DISKS/S, VOLS=VOLUMES/S, GOODONLY/S, BLOCKS/S, DEVICES/M

Location Sys:C

Function Show information on file system devices and volumes. When given no arguments, information on all devices and volumes found in the system is displayed. If information is wanted only for some specific devices, these names may be given as arguments..

Inputs DISKS -- show information on file system devices
VOLS -- show information on volumes
GOODONLY -- don't show any information on bad devices or volumes
BLOCKS -- show additional block size and usage information
DEVICES -- device names to show information about

Example Info

Unit	Size	Used	Free	Full	Errs	State	Type	Name
Foreign harddisk:	964.1M	776.7M	187.4M	81%	0	read/write	OFS	Workbench
RAM:	8.0M	7.1M	7.1M	12%	0	read/write	OFS	Ram Disk

See also [Format](#), [SFSformat](#)

Iprefs

Usage: `Iprefs`

Synopsis (N/A)

Location Sys:C

Function Initialises preferences files on startup.

Notes This is a command executed on startup by the `s:startup-sequence` script file. So no actual need of the command afterwards...

See also [Decoration](#)

Install

Usage: `Install <Drive> [NOBOOT] [CHECK] [FFS]`

Synopsis `DRIVE/A, NOBOOT/S, CHECK/S, FFS/S`

Location Sys:C

Function Saves a boot block to a floppy disk.
If the NOBOOT is appointed it will be able to be boot on computer startup (Amiga only)

Inputs `DRIVE` -- show information on file system devices
 `NOBOOT` -- should be set on PC Floppy drives
 `CHECK` -- Verify the existing boot block
 `FFS` -- For FFS formatted Floppy disks

Example `Install df0: NOBOOT FFS`

Notes This is a pretty useless command for PC-Drives, since most systems require grub to be present on disk for AROS to boot.

See also [Install-i386-pc](#), [Format](#)

Install-grub2-i386-pc

Usage: `Install-grub2-i386-pc <Device> <Unit> [PN=n] [GRUB ldr] [FORCELBA]`

Synopsis `DEVICE/A, UNIT/K/K/A, PN/K/N, GRUB/K/A, FORCELBA/S`

Location	Sys:C
Function	Installs the GRUB 2 bootloader to the boot block of the specified disk or partition, basically making a disk device or partition bootable.
Inputs	<p>DEVICE -- Device name (e.g. ata.device)</p> <p>UNIT -- Unit number</p> <p>PN -- Specifies a partition number. If specified, GRUB is installed to this partition's boot block. Otherwise, GRUB is installed to the disk's boot block.</p> <p>GRUB -- Path to GRUB directory.</p> <p>FORCELBA -- Force use of LBA mode.</p>
Example	<pre>Install-grub2-i386-pc DEVICE ata.device UNIT 0 GRUB DH0:boot/grub</pre>
Notes	This could seriously damage your data in the hard drive...
See also	Partition, Format

Install-i386-pc (now deprecated)

Usage: `Install-i386-pc <Device> <Unit> [PN=n] [GRUB ldr] [KERNEL knl] [FORCELBA]`

Synopsis	DEVICE/A, UNIT/K/K/A, PARTITIONNUMBER=PN/K/N, GRUB/K/A, FORCELBA/S
Location	Sys:C
Function	It basically made a disk device or partition bootable, by installing old GRUB version 1 onto it. It is now obsolete, due to the fact that current AROS builds and distributions use GRUB 2. It's kept in this manual for reference and historical reasons.
Inputs	<p>DEVICE -- device name (i.e.: ata.device)</p> <p>UNIT -- Unit number</p> <p>PN -- Partition number (advice: the first AROS ffs partition)</p> <p>GRUB -- Install the Grub boot loader on this device</p> <p>KERNEL - Install the AROS Kernel onto this device</p> <p>FORCELBA -- Force use of Logical Block Addressing</p>
Example	<pre>install-i386-pc device ata.device unit 0 PN 1 grub dh0:boot/grub kernel dh0:boot/aros-i386.gz</pre>
Notes	This could seriously damage your data in the hard drive...
See instead	Install-grub2-i386-pc

Join

Usage: Join [FILE] {(file | pattern)} AS|TO (filename)

Synopsis FILE/M/A,AS=TO/K/A

Location Sys:C

Function Join makes one big file of all listed files by putting them together in the order given. The destination file may not have the same name as any of input files. You must supply a destination file name. The original files remain unchanged. Any number of files can be Joined in one operation.

Inputs FILE -- files to join
TO=AS -- the name of the combined file

Example Join Text1.doc Text2.doc AS Text.doc

This will merge the two text files into one.

Result Standard DOS error codes.

Lab

Usage: Lab <Name>

Synopsis NAME/A

Location Sys:C

Function Declares a label in a script file. This label may be referred to in a Skip command.

Inputs NAME - The name of the label.

Example

```
If NOT EXISTS S\User-Startup
    Skip NoUserSeq
EndIf

FailAt 20
Execute S\User-Startup
Quit

Lab NoUserSeq
Echo "No User-Startup found"
```

If the file is not found it skips to the error message...

See also Skip

Lha

Usage: lha [-]{axelvudmcp}[qvnfodizg012][w=<dir>] archive [file...]

Synopsis Unix-like command

Location Extras/Aminet/C

Function Compresses, expands and manages files in LHA archives.

Inputs Commands:

- a Add (or replace) to archive
- x, e Extract from archive
- l, v List and Verbose list
- u Update newer files to archive
- d Delete from archive
- m Move to archives (means 'ad')
- c re-construct new archive
- p Print to standard-out from archive
- t Test file CRC in archive

Options:

- q Quiet
- v Verbose
- n Not execute
- f Force (overwrite at extract)
- t Files are text file
- o Use Lharc compatible method (a/u)
- w=<dir> Specify extraction path (x/e)
- d Delete files after (a/u/c)
- i Ignore directory path (x/e)
- z Files not compress (a/u)
- g [Generic] format (for compatibility)
- 0/1/2 Header level (a/u)

LibList

Usage: LibList

Synopsis (N/A)

Location Sys:C

Function Gives a list of loaded libraries: address, version, revision, opencnt, flags, name

List

Usage: List [(dir | pattern | filename)] [PAT (pattern)] [KEYS] [DATES] [NODATES] [TO (name)] [SUB (string)] [SINCE (date)] [UPTO (date)] [QUICK] [BLOCK] [NOHEAD] [FILES] [DIRS] [LFORMAT (string)] [ALL]

Synopsis DIR/M,P=PAT/K,DATES/S,NODATES/S,TO/K,SUB/K,SINCE/K,UPTO/K,QUICK/S

'
BLOCK/S,NOHEAD/S,FILES/S,DIRS/S,LFORMAT/K,ALL/S

Location Sys:C

Function Lists detailed information about the files and directories in the current directory or in the directory specified by DIR.

The information for each file or directory is presented on a separate line, containing the following information:

- name
- size (in bytes)
- protection bits
- date and time

Inputs DIR -- The directory to list. If left out, the current directory will be listed.
PAT -- Display only files matching 'string'
KEYS -- Display the block number of each file or directory
DATES -- Display the creation date of files and directories
NODATES -- Don't display dates
TO (name) -- Write the listing to a file instead of stdout
SUB (string) -- Display only files, a substring of which matches the substring 'string'
SINCE (date) -- Display only files newer than 'date'
UPTO (date) -- Display only files older than 'date'
QUICK -- Display only the names of files
BLOCK -- File sizes are in blocks of 512 bytes
NOHEAD -- Don't print any header information
FILES -- Display files only
DIRS -- Display directories only
LFORMAT -- Specify the list output in printf-style
ALL -- List the contents of directories recursively

The following attributes of the LFORMAT strings are available

- %A -- file attributes
- %B -- size of file in blocks rather than bytes
- %C -- file comment
- %D -- creation date
- %E -- file extension
- %F -- volume name
- %K -- file key block number

%L -- size of file in bytes
%M -- file name without extension
%N -- file name
%P -- file path
%S -- superseded by %N and %P; obsolete
%T -- creation time

Example List C:
Directory "C:" on Wednesday 12-Dec-99
AddBuffers 444 --p-rwed 02-Sep-99 11:51:31
Assign 3220 --p-rwed 02-Sep-99 11:51:31
Avail 728 --p-rwed 02-Sep-99 11:51:31
Copy 3652 --p-rwed 02-Sep-99 11:51:31
Delete 1972 --p-rwed 02-Sep-99 11:51:31
Execute 4432 --p-rwed 02-Sep-99 11:51:31
List 5108 --p-rwed 02-Sep-99 11:51:31
Installer 109956 ----rwed 02-Sep-99 11:51:31
Which 1068 --p-rwed 02-Sep-99 11:51:31
9 files - 274 blocks used

Result Standard DOS error codes.

See also Dir

Load

Usage: Load <device>

Synopsis DEVICE/A

Location Sys:C

Function Loads media into a device. This feature is not supported by all device types.

Inputs DEVICE -- Name of device to load media into.

Loadresource

Usage: Loadresource <name>

Synopsis NAME/M/A

Location Sys:C

Function This command will load a system resource file such as a library, device, font, catalog...
The advantage is that it speed's up the data access to the

resources loaded into memory.

Inputs **NAME** – The resource name

Example 1.SYS:> Loadresource Libs:asl.library

 This will load the asl requester library into memory.

Result Standard DOS error codes.

Lock

Usage: Lock <drive> [ON|OFF] [<passkey>]

Synopsis DRIVE/A,ON/S,OFF/S,PASSKEY

Location Sys:C

Function Lock will cause the specified device or partition to be made write-protected or write-enabled. This write protection is a soft write protection which is handled by the volume file system. Hence the protection will be reset (to writable) on the next system reboot.

It is possible to specify an optional passkey which can be used to password protect the locking. The same passkey that is used to lock the volume must be used to unlock the volume. The passkey may be any number of characters in length.

The volume given **MUST** be the device or root volume name, not an assign.

Inputs **DRIVE** – The drive to be locked/unlocked;
 ON,OFF – Sets the lock status either on or off;
 PASSKEY – A password passed as string to protect the lock or confirm the unlock.

Example 1.SYS:> Lock Work:

 This will lock the volume called Work: without a passkey.

```
1.SYS:> Lock Work:
1.SYS:> MakeDir Work:SomeDir
Can't create directory Work:Test
MakeDir: Disk is write-protected
```

 The volume Work: is locked, so it is impossible to create a directory.

```
1.SYS:> Lock Work: OFF
```

 This will unlock the volume work.

```
1.SYS:> Lock Work: MyPassword
```

This will lock Work: with the passkey "MyPassword"

Result Standard DOS error codes.

Makedir

Usage: `Makedir <Name> [ALL]`

Synopsis `NAME/M,ALL/S`

Location `Sys:C`

Function Create new empty directories with specified names.

Inputs `NAME` -- names of the directories that should be created
 `ALL` -- creates intermediate directories

Example `Makedir Ram:test`

This will create a directory called "Test" in Ram Disk

Notes MakeDir does not create an icon for a new directory.

Makelink

Usage: `Makelink <from> <to> [HARD] [FORCE]`

Synopsis `FROM/A, TO/A, HARD/S, FORCE/S`

Location `Sys:C`

Function Create a link to a file

Inputs `FROM` -- The name of the link
 `TO` -- The name of the file or directory to link to
 `HARD` -- If specified, the link will be a hard-link; default is
 to create a soft-link
 `FORCE` -- Allow a hard-link to point to a directory

Example `Makelink ls c:list`

Creates a "ls" file with a symbol link (symlink) to the "list" command in c:

Result Standard DOS error codes.

Notes Not all file systems support links.

Mount

Usage: `Mount <Device> <From>`

Synopsis `DEVICE/M, FROM/K`

Location `Sys:C`

Function Loads and mounts a device

Inputs `DEVICE -- The device type to be mounted`
 `FROM -- Specify a mount file containing device information`

Example `Mount DEVS:FAT0`

 Mounts a fat device defined on FAT0 file...

Result Standard DOS error code.

Newshell

Usage: `Newshell [<Window Opts>] [<script file>]`

Synopsis `WINDOW, FROM`

Location `Sys:C`

Function Create a new shell in a new console window. This window will become the active one. The new shell inherits most attributes of the parent shell like the current directory, stack size, prompt and so on. However, it is completely independent of the parent shell. The window belonging to the new shell may be specified by using the WINDOW keyword.

Inputs `WINDOW -- Specification of the shell window`

 `X -- number of pixels from the left edge of`
 `the screen`
 `Y -- number of pixels from the top edge of`
 `the screen`
 `WIDTH -- width of the shell window in pixels`
 `HEIGHT -- height of the shell window in pixels`
 `TITLE -- text to appear in the shell window's`

title bar
 AUTO -- the window automatically appears when the program needs input or output
 ALT -- the window appears in the specified size and position when the zoom gadget is clicked
 BACKDROP -- the window is a backdrop window
 CLOSE -- include a close gadget
 INACTIVE -- the window is not made active when opened
 NOBORDER -- the window is borderless, only the size, depth and zoom gadgets are available
 NOCLOSE -- the window has no close gadget
 NODEPTH -- the window has no depth gadget
 NODRAG -- the window cannot be drag; implies NOCLOSE
 NOSIZE -- the window has no size gadget
 SCREEN -- name of a public screen to open the window on
 SIMPLE -- if the window is enlarged the text expands to fill the available space
 SMART -- if the window is enlarged the text will not expand
 WAIT -- the window can only be closed by selecting the close gadget or entering CTRL-\.

FROM -- File to execute before resorting to normal shell operations. If nothing is specified S:Shell-Startup is used.

Example NewShell "CON:10/10/640/480/My own shell/CLOSE"

This will open a shell window with the name "My own shell".

See also Endcli

Open

Usage: Open <Name>

Synopsis NAME/A

Location Sys:C

Function This command is a context sensitive file/drawer loader / executer / display tool.
 It means that it will have appropriate action set for each file type.
 Such as:

- if you open a executable command it will be launched as if a run >nil: command was executed.
- If a text or image file is opened, it will launch Multiview.

- If you open a drawer it will present you a wanderer file browser type window and let's you navigate

Inputs NAME -- The file or drawer name

Example Open Ram:

It will open a wanderer window with the ram disk contents.

Partition

Usage: Partition [<Device> <Unit>] [SYSSIZE=<n>] [WORKSIZE=<N>] [MAXWORK] [WIPE] [FORCE] [QUIET]

Synopsis DEVICE, UNIT/N, SYSSIZE/K/N, WORKSIZE/K/N, MAXWORK/S, WIPE/S, FORCE/S, QUIET/S

Location Sys:C

Function Partition creates either one or two AROS partitions on a given drive. Existing partitions will be kept unless the WIPE option is specified (or a serious bug occurs, for which we take no responsibility). Partitions created by this command must be formatted before they can be used.

By default, a single SFS System partition is created using the largest amount of free space possible. A smaller size can be chosen using the SYSSIZE argument. To also create a Work partition, either WORKSIZE or MAXWORK must additionally be specified. The WORKSIZE argument allows the size of the Work partition to be specified, while setting the MAXWORK switch makes the Work partition as large as possible.

The filesystems used by the System and Work partitions may be specified using the SYSTYPE and WORKTYPE arguments respectively. The available options are "SFS" (Smart Filesystem, the default), and "FFSIntl" (the traditional so-called Fast Filesystem).

The DOS device names used for the System and Work partitions may be specified using the SYSNAME and WORKNAME arguments respectively. By default, these are DH0 and DH1.

If you wish to use only AROS on the drive you run this command on, you can specify the WIPE option, which destroys all existing partitions on the drive. Be very careful with this option: it deletes all other operating systems and data on the drive, and could be disastrous if the wrong drive is accidentally partitioned.

If the drive does not already contain an extended partition, one is created using the largest available region of free space. The AROS

partitions are then created as a logical partition within. This is in order to make the addition of further partitions easier.

Inputs	<p>DEVICE -- Device driver name (ata.device by default)</p> <p>UNIT -- The drive's unit number (0 by default, which is the primary master when using ata.device)</p> <p>SYSSIZE -- The System (boot) partition size in megabytes.</p> <p>SYSTYPE -- The file system to use for the system partition, either "SFS" (the default) or "FFSIntl".</p> <p>SYSNAME -- The name to use for the system partition (defaults to DH0).</p> <p>WORKSIZE -- The Work (secondary) partition size in megabytes. To use this option, SYSSIZE must also be specified.</p> <p>MAXWORK -- Make the Work partition as large as possible. To use this option, SYSSIZE must also be specified.</p> <p>WORKTYPE -- The file system to use for the work partition, either "SFS" (the default) or "FFSIntl".</p> <p>WORKNAME -- The name to use for the work partition (defaults to DH1).</p> <p>WIPE -- Destroy all other partitions on the drive, including those used by other operating systems (CAUTION!).</p> <p>FORCE -- Do not ask for confirmation before partitioning the drive.</p> <p>QUIET -- Do not print any output. This option can only be used when FORCE is also specified.</p> <p>RDB -- Create only RDB partitions, no MBR or EBR partitions will be created.</p>
Example	<pre>Partition ata.device 1 SYSSIZE 200 MAXWORK</pre>
Notes	<p>This is a very dangerous command – use it only if you're sure of what you're doing. Using HDToolBox instead of this command may sometimes be safer, as it shows where partitions will be created on the drive before changes are written to disk. However, HDToolBox can be unreliable.</p>

Path

Usage: Path [{<dir>}] [ADD] [SHOW] [RESET] [REMOVE] [QUIET] [HEAD]

Synopsis	PATH/M,ADD/S,SHOW/S,RESET/S,REMOVE/S,QUIET/S,HEAD/S
Location	Sys:C
Function	Changes the search path for commands. Without arguments it shows the path.
Inputs	<p>PATH -- path</p> <p>ADD -- adds path</p> <p>SHOW -- shows path</p> <p>RESET -- removes existing path and replaces it by new path</p>

REMOVE -- removes the given path
QUIET -- suppresses dialog when a path is not found
HEAD -- inserts path at beginning of path list

Example path dh0:work add

Adds the work dir in dh0 to the search path.

PathPart

Usage: PATHPART [DIR <path name>] [FILE <path name>] [ADD <device name | directory name | file name>]

Synopsis DIR/K,FILE/K,ADD/K/M

Location Sys:C

Function Extracts directory or file name from a path, or assembles a path. This command can break down directory and file names into their respective directory and file name components, and is also able to assemble or re-assemble the individual names into combined path names again. This can be very useful in scripts.

Inputs DIR <path name> – extracts the directory component from the <path name>
FILE <path name> – extracts the file component from the <path name>
ADD – builds a path from <device name>, <directory name> and <file name>

Examples 1.sys>PATHPART DIR Work:Pippo/Pluto

Work:Pippo

Obtains the directory name component of a path.

1.sys> PATHPART FILE Work:Pippo/Pluto

Pluto

Obtains the file name component of a path.

1.sys> PATHPART ADD Work: Pippo Pluto

Work:Pippo/Pluto

Builds a complete new path name from given components

1.sys> PATHPART ADD 'PATHPART DIR
Work:Pippo/Pluto' Foo

Work:Pippo/Foo

Removes the last part of the path name, then replace it with a new one

Ping

Usage: ping [-Rdfnqrv] [-c count] [-i interval] [-l preload] [-p pattern] [-s packetsize] [-l [hosts]] destination

Synopsis Unix-like command, pay attention to case sensing

Location Sys:system/network/AROSTCP/C

Function Ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ("pings") have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of "pad" bytes used to fill out the packet. It's basically useful to understand if a network connection between two hosts has been successful.

Inputs -d
 Set the SO_DEBUG option on the socket being used. Essentially, this socket option is not used by Linux kernel.

-f
 Flood ping. For every ECHO_REQUEST sent a period dot is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. If interval is not given, it sets interval to zero and outputs packets as fast as they come back or one hundred times per second, whichever is more.

-n
 Numeric output only. No attempt will be made to lookup symbolic names for host addresses.

-p pattern
 You may specify up to 16 "pad" bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, -p ff will cause the sent packet to be filled with all ones.

-R
 Record route. Includes the RECORD_ROUTE option in the ECHO_REQUEST packet and displays the route buffer on returned packets. Note that the IP header is only large enough for nine such routes. Many hosts ignore or discard this option.

-q

Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

-c count

Stop after sending count ECHO_REQUEST packets. With deadline option, ping waits for count ECHO_REPLY packets, until the timeout expires.

-i interval

Wait 'interval' seconds between sending each packet. The default is to wait for one second between each packet. This option is incompatible with the -f option.

-l preload

If preload is specified, ping sends that many packets as fast as possible before falling into its normal mode of behavior.

-p pattern

You may specify up to 16 "pad" bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, -p ff will cause the sent packet to be filled with all ones.

-s packetsize

Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

-L [hosts]

Use loose routing IP option. Includes IPOPT_LSRR option in the ECHO_REQUEST packet with all specified hosts in the route. Many hosts won't support loose routing, such a host can either ignore or return the loose routed ICMP packet in the middle of the route.

Pipe

Usage: Pipe <command>

Synopsis COMMAND/F

Location Sys:C

Function	<p>Uses the <code>_pchar</code> and <code>_mchar</code> environment variables to split the <code>COMMAND</code> into fragments.</p> <p>Where <code>_pchar</code> is seen, the commands on either side are connected with a PIPE: from the left side's <code>Output()</code> to the right side's <code>Input()</code>.</p> <p>Where <code>_mchar</code> is seen, the commands are executed in sequence, with no PIPE: between them, and <code>Input()</code> and <code>Output()</code> comes from the terminal.</p>
Inputs	<code>COMMAND</code> -- the command to execute
Example	<pre>1.sys> set _pchar " " 1.sys> set _mchar ";" 1.sys> echo Hello ; echo World Hello World 1.sys> Type S:Startup-Sequence Sort</pre>
Notes	<p>The "<code>_pchar</code>" and "<code>_mchar</code>" environment variables are used to determine where to split the command, and what action to perform.</p> <p>Note that <code>_pchar</code> and <code>_mchar</code> are limited to 2 characters - any additional characters will be silently ignored.</p>

Play

Usage: `Play` [`<FILE>`]

Synopsis	<code>FILE/S</code>
Location	<code>Sys:C</code>
Function	Play a sound file, using <code>datatypes.library</code> .
Inputs	<code>FILE</code> -- Filename to play.
Example	Sound should play to the default audio device

Prompt

Usage: `Prompt` [`<OPTS>`]

Synopsis	<code>OPTS/K</code>
----------	---------------------

Location	Sys:C
Function	Specify the prompt for the current shell.
Inputs	<p>PROMPT -- The prompt to set as a string. The following commands may be used in a printf kind of style.</p> <p>N -- cli number S -- name of the current directory R -- return code of the last operation</p> <p>If no prompt is specified "%N.%S> " is used as default.</p>
Example	<pre>Prompt "Oepir Risti.%N> "</pre> <pre>Oepir Risti.10></pre> <p>(if the CLI number was 10).</p>

Protect

Usage:	<code>Protect [<file>] [FLAGS] [ADD SUB] [ALL] [QUIET]</code>
Synopsis	FILE/A,FLAGS,ADD/S,SUB/S,ALL/S,QUIET/S
Location	Sys:C
Function	<p>Add or remove protection bits from a file or directory.</p> <p>Protect allows the use of pattern matching and recursive directory scans to protect many files/directories at any one time.</p>
Inputs	<p>FILE -- Either a file, a directory or a pattern to match.</p> <p>FLAGS -- One or more of the following flags:</p> <p>S - Script P - Pure A - Archive R - Read W - Write E - Execute D - Delete</p> <p>ADD -- Allows the bits to be set and hence allowable. SUB -- Allows the bits to be cleared and hence not allowable. ALL -- Allows a recursive scan of the volume/directory. QUIET -- Suppresses any output to the shell.</p>
Results	Standard DOS return codes.

Example `Protect ram: e add all`

 Recurses the ram: volume and attaches the executable bit.

PsdDevLister

Usage: `PsdDevLister`

Synopsis `SHOWROOT/S,QUICK/S,STRINGS/S`

Location `Sys:C`

Function Give a detailed list of all the USB devices currently in the system. It is appreciated that you include the output of this program for bug reporting.

Inputs `SHOWROOT/S` - Normally, the root hub(s) are excluded from the output, as they don't contain valuable information. Specify this switch, if you really want to see it.
`QUICK/S` - If given, omits some output, decreasing verbosity.
`STRINGS/S` - Tries to read out a list of string descriptors the device contains, might cause some devices to crash, hence this is disabled by default.

Example `1.> PsdDevLister`

PsdErrorLog

Usage: `PsdErrorLog`

Synopsis `NOFLUSH/S,DEBUG/S`

Location `Sys:C`

Function Prints out all information, warning and error messages accumulated so far in the Poseidon stack. These messages will automatically be flushed, so calling `PsdErrorLog` another time will only reveal the new messages since the last call.

Inputs `NOFLUSH/S` - If given, outputs the errors without discarding them.
`DEBUG/S` - Prints some additional debug information. If Poseidon ever seems to hang, include the output of `PsdErrorlog` with `DEBUG` enabled.

Example `1.> PsdErrorLog NOFLUSH >ram:Errors.log`

 Redirects the error log to file `Errors.log` in RAM:, without flushing

errors from memory.

PsdStackLoader

Usage: `PsdStackLoader`

Synopsis	(N/A)
Location	Sys:C
Function	Used internally to run Poseidon USB Stack. It's automatically created by Trident.

Quit

Usage: `Quit`

Synopsis	RC/N
Location	Sys:C
Function	Exit the script execution returning some sort pre-defined code
Inputs	RC -- the return code

Quitaros

Usage: `Quitaros`

Synopsis	(N/A)
Location	Sys:C
Function	This command is only of interest if you're using a full-screen host system - for ease of use. It quits the Host client...

Reboot

Usage: `Reboot [COLD]`

Synopsis	COLD/S
----------	--------

Location	Sys:C
Function	Reboots the machine. Any programs and data in memory will be lost and all disk activity will cease. Make sure no disk access is being carried out by your computer.
Inputs	COLD -- tells to perform cold (complete) reboot of the machine. Otherwise only AROS is restarted.

Relabel

Usage: `Relabel DRIVE=<Drive> NAME=<Name>`

Synopsis	DRIVE/A, NAME/A
Location	Sys:C
Function	Rename a volume
Inputs	DRIVE -- The volume to rename NAME -- The new name
Example	<code>Relabel Drive DF0: Name "MyDisk"</code> This will rename the disk in df0: as MyDisk.

Rename

Usage: `Rename [{FROM}] <name> [TO|AS] <name> [QUIET]`

Synopsis	FROM/A/M,TO=AS/A,QUIET/S
Location	Sys:C
Function	Renames a directory or file. Rename can also act like the UNIX mv command, which moves a file or files to another location on disk.
Inputs	FROM -- The name(s) of the file(s) to rename or move. There may be many files specified, this is used when moving files into a new directory. TO AS -- The name which we wish to call the file. QUIET -- Suppress any output from the command.
Result	Standard DOS error codes.

Example `Rename letter1.doc letter2.doc letters`

 Moves letter1.doc and letter2.doc to the directory letters.

```
Rename ram:a ram:b quiet
Rename from ram:a to ram:b quiet
Rename from=ram:a to=ram:b quiet
```

 All versions, renames file "a" to "b" and does not output any diagnostic information.

RequestChoice

Usage: Requestchoice TITLE=<Title> BODY=<Body> GADGETS=<Gadgets> [**<Pubscreen>**]

Synopsis TITLE/A,BODY/A,GADGETS/A/M,PUBSCREEN/K

Location Sys:C

Function Allows AmigaDOS scripts to have access to the EasyRequest() function for input.

Inputs TITLE - The text to display in the title bar of the requester.
 BODY - The text to display in the body of the requester.
 GADGETS - The text for each of the buttons.
 PUBSCREEN - The name of the public screen to open the requester on.

Result Standard Codes

Example RequestChoice "This is a title" "This is*Na body" Okay|Cancel

 This is self-explanatory, except for the "*N". This is the equivalent of using a '\n' in C to get a newline in the body of the requester. This requester will open on the Workbench screen.

```
RequestChoice Title="This is a title" Body="This is*Na
body" Gadgets=Okay|Cancel PubScreen=DOPUS.1
```

 This will do exactly the same as before except that it will open on the Directory Opus public screen.

Notes To place a newline into the body of the requester use *n or *N.

 To place a quotation mark in the body of the requester use *".

 The CLI template gives the GADGETS option as ALWAYS given; this is different from the original program. This way, we do not have to check to see if the gadgets have been given.

RequestFile

Usage: Requestfile [DRAWER=<Drawer>] [FILE=<File>] [PATTERN=<Pattern>]
[TITLE=<Title>] [POSITIVE=<Positive>] [NEGATIVE=<Negative>]
[ACCEPTPATTERN=<Acceptpattern>] [REJECTPATTERN=<Rejectpattern>] [SAVEMODE]
[MULTISELECT] [DRAWERONLY] [NOICONS] [PUBSCREEN=<Pubscreen>]
[INITIALVOLUMES]

Synopsis DRAWER,FILE/K,PATTERN/K,TITLE/K,POSITIVE/K,NEGATIVE/K,
ACCEPTPATTERN/K,REJECTPATTERN/K,SAVEMODE/S,MULTISELECT/S

'
DRAWERONLY/S,NOICONS/S,PUBSCREEN/K,INITIALVOLUMES/S
Location Sys:C

Function Creates file requester. The selected files will be displayed separated by spaces. If no file is selected the return code is 5 (warn).

Inputs DRAWER -- initial content of drawer field
FILE -- initial content of file field
PATTERN -- content of pattern field (e.g. #?.c)
TITLE -- title of the dialog box
POSITIVE -- string for the left button
NEGATIVE -- string for the right button
ACCEPTPATTERN -- only files which match the pattern are displayed
REJECTPATTERN -- files which match the pattern aren't displayed
SAVEMODE -- requester is displayed as save requester
MULTISELECT -- more than one file can be selected
DRAWERONLY -- only drawers are displayed
NOICONS -- no icon files (#?.info) are displayed
PUBSCREEN -- requester is opened on the given public screen
INITIALVOLUMES -- shows the volumes

Result Standard DOS error codes.

Example Requestfile DRAWER="Work:Documents" PATTERN="#?.doc"
TITLE="Select a document file"

RequestString

Usage: RequestString [STRING] [TEXT] [TITLE] [NOGADS] [WIDTH] [SAFE]
[PERSIST] [ENCRYPT] [COMPARE] [PUBSCREEN]

Synopsis STRING, TEXT/K, TITLE/K, NOGADS/S, WIDTH/N, SAFE/S,
PERSIST/S, ENCRYPT/S, COMPARE/K, PUBSCREEN/K

Location Sys:C

Function Shows a requester with a string gadget for user input.

Input	<p>STRING -- Initial content of string gadget.</p> <p>TEXT -- Label string.</p> <p>TITLE -- Title string of requester. This also adds dragbar, closegadget and a depthgadget.</p> <p>NOGADS -- Suppress gadgets when TITLE argument is given.</p> <p>WIDTH -- Minimal width as number of characters.</p> <p>SAFE -- Hide user input with "*".</p> <p>PERSIST -- Intuition is blocked until requester is quitted.</p> <p>ENCRYPT -- Encrypt result before returning. Requires that one of these environment variables is set: USER, USERNAME or LOGIN.</p> <p>COMPARE -- If the input string is not equal to the argument of COMPARE return WARN.</p> <p>PUBSCREEN -- Open requester on given pubscreen.</p>
Notes	<p>PERSIST doesn't always work.</p> <p>WIDTH is not implemented.</p>

Resident

Usage: Resident [<Name>] [<File>] [REMOVE] [ADD] [REPLACE] [PURE|FORCE] [SYSTEM]

Synopsis	NAME, FILE, REMOVE/S, ADD/S, REPLACE/S, PURE=FORCE/S, SYSTEM/S
Location	Sys:C
Function	Stores a command in memory (no longer requires reloading from disk each time it is executed). To ensure a command can be made resident check the pure flag – Use "list" command.
Inputs	<p>NAME – Command name</p> <p>FILE, - File command name</p> <p>REMOVE – Removes the command from memory</p> <p>ADD – Add to memory</p> <p>REPLACE – replace any equal command entry avail in memory</p> <p>PURE=FORCE – Force to make it pure resident</p> <p>SYSTEM – Shows the system's resident code segmens or add code to the system resident list. Leave well alone!</p>
See also	Which, List

ResList

Usage: ResList

Synopsis	(N/A)
----------	-------

Location	Sys:C
Function	Gives a list of loaded resources: address and names.

Resolve

Usage: `resolve [NET] IPADDR [PROTOCOLS] [PORT]`

Synopsis	NET/S,IPADDR,PROTOCOL/N/K,TCP/S,UDP/S,PORT/K/N
Location	Sys:system/network/AROSTCP/C
Function	Resolve resolves Internet address, network number, protocol number or port number. Host addresses are resolved by Domain name server, if your host is using one.
Inputs	<p>NET/S – The IPADDR is regarded as network address instead of host address.</p> <p>IPADDR – The Internet address in standard dot notation to resolve.</p> <p>PROTOCOL/N/K – The Internet protocol number.</p> <p>PORT/K/N – The port number in the range 0 -- 65535. A port number and protocol identifies an Internet service uniquely.</p> <p>TCP/S – Look up a service using TCP protocol (default).</p> <p>UDP/S – Look up a service using UDP protocol.</p>

Route

Usage: `route [-n] [-q] [-v] command [modifiers] destination gateway`

Synopsis	Unix-like command
Location	Sys:system/network/AROSTCP/C
Function	show / manipulate the IP routing table.
Inputs	Options supported by route:
	-n

Prevent attempts to print host and network names symbolically when reporting actions.

`-v`
(verbose) Print additional details.

`-q`
Suppress all output.

Commands accepted by route:

`add`
Add a route.

`delete`
Delete a specific route.

The destination is the destination host or network, gateway is the next-hop gateway to which packets should be addressed. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with destination. The optional modifiers `-net` and `-host` force the destination to be interpreted as a network or a host, respectively. Otherwise, if the destination has a "local address part" of `INADDR_ANY`, or if the destination is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

For example, `128.32` is interpreted as `-host 128.0.0.32`; `128.32.130` is interpreted as `-host 128.32.0.130`; `-net 128.32` is interpreted as `128.32.0.0`; and `-net 128.32.130` is interpreted as `128.32.130.0`.

To add a default route, give the destination as 'default'.

If the route is via an interface rather than via a gateway, the `-interface` modifier should be specified; the gateway given is the address of this host on the common network, indicating the interface to be used for transmission.

The optional `-netmask` qualifier is used to specify the netmask of the interface. One specifies an additional ensuing address parameter (to be interpreted as a network mask). The implicit network mask generated can be overridden by making sure this option follows the destination parameter.

All symbolic names specified for a destination or gateway are looked up first as a host name using `gethostbyname()`. If this lookup fails, `getnetbyname()` is then used to interpret the name as that of a network.

Diagnostics

`add [host | network] %s: gateway %s flags %x`
 The specified route is being added to the tables. The values printed are from the routing table entry supplied in the `IoctlSocket()` call. If the gateway address used was not the primary address of the gateway (the first one returned by `gethostbyname()`), the gateway address is printed numerically as well as symbolically.

`delete [host | network] %s: gateway %s flags %x`
 As above, but when deleting an entry.

Network is unreachable
 An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

not in table
 A delete operation was attempted for an entry which wasn't present in the tables.

routing table overflow
 An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

Run

Usage: `Run [QUIET] [EXECUTE] <Command>`

Synopsis `COMMAND/F, QUIET/S, EXECUTE/S`

Location `Sys:C`

Function Run a program, that is start a program as a background process. That means it doesn't take over the parent shell.

Inputs `COMMAND` -- the program to run together with its arguments
 `QUIET` -- avoids printing of the background CLI's number
 `EXECUTE` -- Executes a shell script instead

Notes To make it possible to close the current shell, redirect the output using:
 `Run >NIL: program arguments`

Search

Usage: `Search [FROM] {(name | pattern} [SEARCH] (string | pattern) [ALL] [NONUM] [QUIET] [QUICK] [FILE] [PATTERN] [LINES=Number]`

Synopsis	FROM/K, SEARCH/K, ALL/S, NONUM/S, QUIET/S, QUICK/S, FILE/S, PATTERN/S, LINES/N
Location	Sys:C
Function	Search looks through the files contained in the FROM directory for a specified string (SEARCH); in case the ALL switch is specified, the subdirectories of the FROM directory are also searched. The name of all files containing the SEARCH string is displayed together with the numbers of the lines where the string occurred. If CTRL-C is pressed, the search will be abandoned. CTRL-D will abandon searching the current file.
Inputs	NONUM -- no line numbers are printed QUIET -- don't display the name of the file being searched QUICK -- more compact output FILE -- look for a file with a specific name rather than a string in a file PATTERN -- use pattern matching when searching CASE -- use case sensitive pattern matching when searching LINES -- extra lines after a line match which should be shown
Notes	If the object is found, the condition flag is set to 0. Otherwise it's set to WARN.

Set

Usage: Set [<Name> <String>]

Synopsis	NAME,STRING/F
Location	Sys:C
Function	Set a local environment variable in the current shell. If any global variables have the same name the local variable will be used instead. This instance the variable is only accessible from within the shell it was defined. If no parameters are specified, the current list of local variables is displayed.
Inputs	NAME - The name of the local variable to set. STRING - The value of the local variable NAME.
Result	Standard DOS error codes.
Example	Set Jump 5

Sets a local variable called "Jump" to the value of "5".

See also Get, Unset

Setcache

Usage: Setcache <Device> [<Lines>] [<Readahead>] [NOCOPYBACK]

Synopsis DEVICE/A, LINES/N, READAHEAD/N, NOCOPYBACK/S

Location Sys:C

Function SetCache sets the read-ahead cache of a Smart File system device (usually a hard drive). It is not the same as the buffers you can add using the AddBuffers command.
The read-ahead cache is used to pre-fetch data which may be needed later on. Because most hard disks don't suffer a speed penalty when reading a bit more data this can increase performance drastically.

Inputs DEVICE - The device name you want to set the read caches.
 Don't use the colon (:) at the end.
 LINES - Controls the number of read-ahead buffers. Each buffer is a specific bytes in size, which you can control using the READAHEAD parameter. It is a good idea to set this to at least 8 buffers or more.
 READAHEAD - The number of bytes which the file system will read ahead. This controls the size of the buffers you specified with the LINES parameter.
 NOCOPYBACK - Turns off copyback mode. There shouldn't be any reason why you want to do this, short of testing purposes.

Example SetCache SFSDRIVE Lines=8 ReadAhead=8192

 Sets the read-ahead cache to 8 buffers of 8192 bytes each

Notes This command is specific for Smart Filesystem drives, and it won't work with the Fast File System (FFS) or another.

SetClock

Usage: SetClock {LOAD|SAVE|RESET}

Synopsis LOAD/S,SAVE/S,RESET/S

Location Sys:C

Function	<p>SetClock can be used to:</p> <ul style="list-style-type: none"> - Load the time from the battery backed-up clock; - Save the time to the battery backed-up clock; - Reset the battery backed up clock.
Example	<p>SetClock LOAD</p> <p>Will set the system time from the battery backed-up clock. In most systems this will be done automatically during system startup.</p> <p>SetClock SAVE</p> <p>Will set the time of the battery backed-up clock from the current system clock time.</p> <p>SetClock RESET</p> <p>Will reset the battery backed-up to a value of the 1st January 1978 00:00:00. This is mostly used if the battery backed-up clock has an error and will not respond to normal load and save commands.</p>
Notes	Date, Time Preference setting (AROS User Manual)

SetDefaultFont

Usage: SetDefaultFont <Fontname> <Fontsize> [SCREEN]

Synopsis FONTNAME/A, FONTSIZE/N/A, SCREEN/S

Location Sys:C

Function Set the default system/screen Font

Inputs

- FONTNAME -- the name of the font
- FONTSIZE -- the size of the font
- SCREEN -- if specified set the default screen font otherwise set the default system font.

Example SetDefaultFont ttcourier 12

Notes The default system font must be mono spaced (non-proportional)

SetDate

Usage: SetDate <Filename> [<WEEKDAY>] [<DATE>] [<TIME>] [ALL]

Synopsis FILE/A, WEEKDAY, DATE, TIME, ALL/S

Location	Sys:C
Function	Changes the file timestamp.
Inputs	FILE -- the name of the font WEEKDAY -- specifies the weekday DATE -- specifies the date TIME -- specifies the time ALL -- Changes all files and dirs timestamps in the path given
Example	<code>SetDate ram:testfile 12-12-2005</code>

Setenv

Usage: `Setenv [<Name>] [SAVE] [<String>]`

Synopsis	NAME,SAVE/S,STRING/F
Location	Sys:C
Function	<p>Sets a global variable from the current shell. These variables can be accessed from any program executing at any time.</p> <p>These variables are usually not saved in the ENVARC: directory, hence they can only be used by programs during the current execution of the operating system. When using SAVE argument, the variable is also saved in ENVARC: If no parameters are specified, the current list of global variables are displayed.</p>
Inputs	NAME - The name of the global variable to set. SAVE - Save the variable also in ENVARC: STRING - The value of the global variable NAME.
Result	Standard DOS error codes
Example	<code>Setenv DEFEDITOR Editor</code> <p>Any program that accesses the variable "DEFEDITOR" will be able to find out the name of the text-editor the user would like to use, by examining the contents of the variable.</p>
See also	Getenv, Unsetenv.

SetKeyboard

Usage: `SetKeyboard <Keymap>`

Synopsis	KEYMAP/A
----------	----------

Location	Sys:C
Function	Set the keymap for the current shell.
Inputs	KEYMAP -- the keymap to use with the current shell
Example	<code>SetKeyboard s</code> Makes the current shell use the Swedish keymap.
Notes	To make a certain keymap be the default for all shells, use the preferences input program so specify your default choice.

SFSformat

Usage: `SFSformat DRIVE <Drive> [NAME <name>] [CASESENSITIVE] [NORECYCLED|SHOWRECYCLED]`

Synopsis	DRIVE/A/K, NAME/A/K, CASESENSITIVE/S, NORECYCLED/S, SHOWRECYCLED/S
Location	Sys:C
Function	Format will initialise a disk to be useable by the AROS with the SFS file system.
Inputs	<p>DRIVE -- The device disk unit designation</p> <p>NAME -- Sets the volume name.</p> <p>CASESENSITIVE -- Enables different filename evaluation performed according upper or lower case chars used .</p> <p>NORECYCLED -- Does not create a .recycled dir in the device formatted.</p> <p>SHOWRECYCLED -- Enables the .recycled drawer to show up and be selected Wanderer or DirOpus etc. as a normal dir entry on the device formatted</p>
Example	<p><code>Format Drive DH1: Name "MyDisk" SHOWRECYCLED</code></p> <p>This will format the secondary disk partition (or a second hard disk) with the name "Mydisk" and with Smart File System.</p>
Notes	This command only supports SFS file system.

SFSquery

Usage: `SFSquery <Device>`

Synopsis	DEVICE/A/K
Location	Sys:C
Function	It basically provides information on a given formatted SFS device partition.
Inputs	<p>DRIVE -- The device disk unit designation</p> <p>NAME -- Sets the volume name.</p> <p>CASESENSITIVE -- Enables different filename evaluation performed according upper or lower case chars used .</p> <p>NORECYCLED -- Does not create trashcan drawer in the device formatted.</p> <p>SHOWRECYCLED -- Creates a trashcan drawer on the device formatted</p>
Example	<pre>10.OS4:> sfsquery work:</pre> <pre>SFSquery information for work: (SFS Version 1.2) Start/end-offset : 0x00000000:00102000 - 0x00000000:3baf6000 Device API : (standard) Bytes/block : 512 Total blocks : 1953696 Cache accesses : 1932316 Cache misses : 3973 (0%) Read-ahead cache : 8x 8192 bytes (Copyback) Flush timeout : act. 20s - inact. 0.5s Max Name Length : 107 DOS buffers : 1000 SFS settings : [RECYCLED]</pre>
Notes	This command only supports SFS file system.

Shell

Usage: Shell [COMMAND=<Command>] [FROM=<From>]

Synopsis	COMMAND/K/F, FROM
Location	Sys:C
Function	Start a shell (interactive or background).
Inputs	<p>COMMAND -- command line to execute</p> <p>FROM -- script to invoke before user interaction</p>
Example	<pre>shell FROM S:Startup-Sequence</pre> <p>Starts a shell and executes the startup script.</p>
Notes	The script file is not a script in execute sense (as you may not use any .key, .bra or .ket and similar things).
See also	Execute, NewShell

Shutdown

Usage: Shutdown

Synopsis	(N/A)
Location	Sys:C
Function	Shuts down the system and turn the computer off.
Notes	Currently, it works on AROS-M68K under Janus-UAE, WinUAE and other UAE ports. Other architectures still need other APIC components to be coded, before the system will be actually powerable off.

Skip

Usage: Skip <Label> [BACK]

Synopsis	LABEL, BACK/S
Location	Sys:C
Function	Skip commands in a script file until a certain label (declared with Lab) or an EndSkip command is reached.
Inputs	LABEL -- The label to skip to. BACK -- Specify this if the label appears before the Skip statement in the script file.
See also	Lab, EndSkip
Notes	SKIP BACK won't work for nested scripts. In a script similar to the following one, SKIP BACK won't work: <pre>; script 1 which launches script 2 LAB Test ; do something EXECUTE Script2 ; when Script2 has ended, ; and execution gets back to Script1, ; do something else... SKIP Test BACK</pre>

Sort

Usage: `Sort <From> <To> [<Colstart>] [CASE] [NUMERIC]`

Synopsis `FROM/A,TO/A,COLSTART/K,CASE/S,NUMERIC/S`

Location `Sys:C`

Function Sorts the contents of a text file

Inputs `FROM` -- file to read from
`TO` -- file to output to
`COLSTART` -- column at which the comparison begins
`CASE` -- sort is case sensitive. Uppercase items are output first
`NUMERIC` -- lines are interpreted as numbers

Example `Format Drive DH1: Name "MyDisk"`

This will format the secondary disk partition (or a second hard disk) with the name "Mydisk" and with Smart File System.

Notes This command is only usable with text or similar files (don't use on binary files).

SortCopy

Usage: `SortCopy FROM sources TO destination [arguments]`

Synopsis `FROM/A,TO/A,QT=QUIET/S,QTR=QUIETER/S,SETA=SETARCHIVE/S, NONA=NONAONLY/S,OVR=OVERWRITE/S,LVE=LEAVEEMPTY/S,DELC =DELETECOPIED/S,FORCE/S,UPD=UPDATE/S,PRB=PURGEBACKUP/S, PRO=PURGEONLY/S`

Location `Extras:Misc`

Function SortCopy is a tiny shell-only backup program, it copies one or multiple files and directories, like the usual "copy" command does, trying to copy links as well. If the link source and the link destination is copied, "SortCopy" will erase the link destination and re-create the link to the copy of the link source. This works for both, SoftLinks and HardLinks, as well as links to directories as long as the directory and the link to the directory is copied. As a tiny extra, it keeps the destination directories alphabetically sorted.

Inputs `FROM:` Source specifications.
`TO:` Destination of the copy operation.
`QUIET:` Suppresses printing the file names being copied.
`QUIETER:` Suppresses all messages except errors and warnings.
`SETARCHIVE:` Set the archive bits of the copied files afterwards.

NONAONLY: Copy only files with the archive bit cleared.
 OVERWRITE: Overwrite destination files in any case.
 LEAVEEMPTY: Don't delete empty directories in the destination.
 DELETEDCOPIED: Delete files after copying them.
 FORCE: Delete also files that are protected from deletion.
 UPDATE: Update only files with newer creation date.
 PURGEBACKUP: Remove old files from the backup.
 PURGEONLY: Remove only old files, but do not copy.

Examples `SortCopy FROM SCSI: TO DH3:SCSI QUIET SETARCHIVE
NONAONLY OVERWRITE`

Copies all files from my HD (called SCSI) to DH3, into a directory called "SCSI". Be QUIET - don't print too much information, print only directory names and not file names. SETARCHIVE and NONAONLY are used to perform a consistent incremental backup, OVERWRITE is given even to overwrite write-protected files in the destination.

`SortCopy FROM SCSI: TO DH3:SCSI QUIET SETARCHIVE
NONAONLY OVERWRITE PURGEBACKUP FORCE`

In addition to the line above, SortCopy is instructed to remove obsolete files from the destination ("PURGEBACKUP"), even if they are write protected ("FORCE").

`SortCopy FROM DH3:SCSI to SCSI: QUIET OVERWRITE`

Restores a backup back to HD. It doesn't set any archive bits in the backup, given here as "FROM" argument.

`SortCopy FROM SCSI: to DH3:SCSI PURGEONLY FORCE`

It will "cleanup" my backup by removing all old files, even if they are write or delete-protected.

Notes Excerpts from the original "SortCopy" documentation of version 1.22 by Thomas Richter. The original, complete document is available here:
http://cd.textfiles.com/amigaformat/aformat-2719980515/Seriously_Amiga-/Shareware/Misc/SortCopy/SortCopy.guide

Stack

Usage: `Stack [[SIZE] (stack size)]`

Synopsis `SIZE/N`

Location `Sys:C`

Function	Stack sets the default stack size of the current Shell. This is the stack size of the commands run from the Shell. If you use Stack without arguments, the current stack size will be written out.
Inputs	SIZE -- Sets the stack size
Notes	From wanderer you can set the stack size in the information window of a given tool icon.

Status

Usage: Status [<Process>] [FULL] [TCB] [CLI|ALL] [COM|COMMAND=<Com>]

Synopsis	PROCESS/N,FULL/S,TCB/S,CLI=ALL/S,COM=COMMAND/K
Location	Sys:C
Function	Display information about the processes that are executing within Shells/CLIs.
Inputs	<p>PROCESS -- Process Identification number.</p> <p>FULL -- Display all information about the processes.</p> <p>TCB -- As for Full, except that this option omits the process name.</p> <p>CLI=ALL -- Default. Displays all processes.</p> <p>COM=COMMAND -- Show the process id of the command given. Specify the command name.</p>

Example	<pre>Status Process 2: Loaded as command: c:status Process 3: Loaded as command: c:NewIcons Process 4: Loaded as command: GG:Sys/L/fifo-handler Process 5: Loaded as command: Workbench Status full Process 2: stk 300000, pri 0 Loaded as command: c:status Process 3: stk 4096, pri 0 Loaded as command: c:NewIcons Process 4: stk 4096, pri 0 Loaded as command: GG:Sys/L/fifo- handler Process 5: stk 6000, pri 1 Loaded as command: Workbench</pre>
---------	---

Type

Usage: Type {<file | pattern>} [TO <name>] [OPT H | N] [HEX | NUMBER]]

Synopsis	FROM/A/M,TO/K,OPT/K,HEX/S,NUMBER/S
Location	Sys:C

Function	Displays content of a file
Inputs	FROM -- one or more files to display TO -- print output to file OPT -- H or N (see HEX or NUMBER) HEX -- displays output in hexadecimal format NUMBER -- the lines are numbered
Example	<code>type abc.txt</code> <code>type xyz.dat hex</code>
Notes	HEX and NUMBER are mutually exclusive

UnACE

Usage: UNACE <command> [<switches>] <archive[.ace]>

Synopsis	Unix-like command
Location	Extras/Aminet/C
Function	Extracts files from ACE-compressed archives.
Inputs	Where <command> is one of: <ul style="list-style-type: none"> e Extract files l List archive t Test archive integrity v List archive (verbose) x Extract files with full path <p>And <switches> is zero or more of:</p> <ul style="list-style-type: none"> -y Assume 'yes' on all questions, never ask for input
Notes	UNACE v1.2 public version

Unalias

Usage: Unalias [<name>]

Synopsis	NAME/K
Location	Sys:C

Function	Removes a previously defined shell alias. If no parameters are specified, the current list of aliases are displayed.
Inputs	NAME - The name of the alias to unset.
Result	Standard DOS error codes.
Example	<code>Unalias DF</code>
See also	Alias

UnARJ

Usage: UNARJ <command> <archive> [<file>...] [<destdir>]

Synopsis	Unix-like command
Location	Extras/Aminet/C
Function	Extracts files from ARJ-compressed archives.
Inputs	Where <command> is one of: e Extract file(s) from archive l List file(s) t Test file(s) in archive x Extract with pathnames

UnDMS

Usage: UNDMS [source] [destination]

Synopsis	Unix-like command
Location	Extras/Aminet/C
Function	Extracts files from DMS-compressed archives.

UnLZX

Usage: UNLZX <command> archive

Synopsis	Unix-like command
Location	Extras/Aminet/C
Function	Extracts files from LZX-compressed archives.
Inputs	-v <archive> : list archive -x <archive> : extract archive -p <pattern> : only matching files -o <outpath> : destination path

Unpack

Usage: Unpack [<name>] TO [<path>]

Synopsis	FILE/A, TO/A:
Location	Sys:C
Function	Command to unpack/unarchive AROS .pkg files.
Inputs	NAME - The name of the file to unpack. TO – The drive or path to be unpacked.
Result	Standard DOS error codes.
Example	Unpack AROS.pkg TO Ram:
Notes	This command is not a tool like lha, lzx or unzip. The .pkg files are not compressed.

Unset

Usage: Unset <name>

Synopsis	NAME/K
Location	Sys:C
Function	Unset a local variable
Inputs	NAME - The name of the local variable to unset.
Result	Standard DOS error codes.
See also	Set

Unsetenv

Usage: `Unsetenv <name>`

Synopsis	NAME/K
Location	Sys:C
Function	Unset a global variable
Inputs	NAME - The name of the global variable to unset.
Result	Standard DOS error codes.
See also	Set

UnShar

Usage: `unshar {-overwrite} {-nosort} <filename> ...`

Synopsis	Unix-like command
Location	Extras/Aminet/C
Function	Extracts files from UNIX Shar-compressed archives.

UnZip

Usage: `unzip [-Z] [-opts[modifiers]] file[.zip] [list] [-x xlist]
[-d exdir]`

Synopsis	COMMANDS/S,MODIFIERS,FILE/A
Location	Sys:C
Function	Command to unpack/unarchive ZIP files.
Inputs	FILE - The name of the file to unpack. COMMANDS – Tell UnZip what to do exactly: -p extract files to pipe, no messages -l list files (short format) -f freshen existing files, create none -t test compressed archive data -u update files, create if necessary

- z display archive comment only
- v list verbosely/show version info
- T timestamp archive to latest
- x exclude files that follow (in xlist)
- d extract files into exdir

MODIFIERS – parameters which customize extraction:

- n never overwrite existing files
- q quiet mode (-qq => quieter)
- o overwrite files WITHOUT prompting
- a auto-convert any text files
- j junk paths (do not make directories)
- aa treat ALL files as text
- C match filenames case-insensitively
- L make (some) names lowercase
- N restore comments as filenotes
- V retain VMS version numbers
- M pipe through "more" pager

Examples 1.sys> unzip data1 -x joe

extract all files except joe from zipfile data1.zip

1.sys> unzip -p foo | more

send contents of foo.zip via pipe into program more

1.sys> unzip -fo foo ReadMe

quietly replace existing ReadMe if archive file newer

Notes UnZip 6.00 of 20 April 2009, by Info-ZIP. Maintained by C. Spieler.

Version

Usage: Version [<library|device|file>] [<version #>] [<revision #>] [FILE]
[FULL] [RES]

Synopsis NAME/M,MD5SUM/S,VERSION/N,REVISION/N,FILE/S,FULL/S,RES/S

Location Sys:C

Function Prints or checks the version and revision information of a file,
library or device.

Inputs NAME -- name of file, library or device to check. If not given it
 prints version and revision of Kickstart.
 MD5SUM -- #FIXME what is that?
 VERSION -- checks for version and returns error code 5 (warn) if

the version of the file is lower.
 REVISION -- checks for revision and returns error code 5 (warn) if
 the revision of the file is lower.
 FILE -- reads from file and ignores currently loaded libraries and
 devices
 FULL -- prints additional information
 RES -- gets version of resident commands

Example `Version libs:ptplay.library`

Wait

Usage: `Wait [(n)] [SEC | SECS | MIN | MINS] [UNTIL (time)]`

Synopsis `TIME/N,SEC=SECS/S,MIN=MINS/S,UNTIL/K`

Location `Sys:C`

Function Wait a certain amount of time or until a specified time. Using
 Wait without any arguments waits for one second.

Inputs `TIME` -- the number of time units to wait (default is seconds)
 `SEC=SECS` -- set the time unit to seconds
 `MIN=MINS` -- set the time unit to minutes
 `UNTIL` -- wait until the specified time is reached. The time
 is given in the format HH:MM.

Example `Wait 5 SECS`

WaitX

Usage: `WaitX [time unit=time] [L=loop] [A] [V] <cmd line>`

Synopsis `D=DATE/K,T=TIME/K,YR=YEARS/K/N,MN=MONTHS/K/N,DY=DAYS
 /K/N,H=HOURS/K/N,M=MINS/K/N,S=SECS/K/N,L=LOOP/K/N,A=AL
 WAYS/S,V=VERBOSE/S,HELP/S,CMDLINE/F`

Location `Sys:C`

Function WaitX will wait for a given amount of time and then it will execute
 the given command. It will not return to prompt while waiting: this
 is intended.

Inputs `D=DATE` -- Waits until DATE has been reached
 `T=TIME` -- Waits until TIME has been reached
 `YR=YEARS` -- How many years to wait

MN=MONTHS -- How many months to wait
DY=DAYS -- How many days to wait
H=HOURS -- How many hours to wait
M=MINS -- How many minutes to wait
S=SECS -- How many seconds to wait
L=LOOP -- How many times to execute CMDLINE
A=ALWAYS -- Execute CMDLINE every set interval/time/date
V=VERBOSE -- Print extra info on what waitx is doing

Notes Based on Public Domain WaitX:
<http://aminet.net/package/util/cli/waitx>
Programming: Sigbjørn Skjæret <cisc@c2i.net>
Idea & Docs: Nicholas Stallard <snowy@netphile.de>

Which

Usage: Which <file> [NORES|RES] [ALL]

Synopsis FILE/A, NORES/S, RES/S, ALL/S

Location Sys:C

Function Find and print the location of a specific program or directory. Resident programs are marked as RESIDENT if they are not integrally resident in which case they are marked as INTERNAL.

Which searches the resident list, the current directory, the command paths and the C: assign. If the item was not found the condition flag is set to WARN but no error is printed.

Inputs FILE -- the command/directory to search for
NORES -- don't include resident programs in the search
RES -- consider resident programs only
ALL -- find all locations of the FILE. This may cause the printing of the same location several times, for instance if the current directory is C: and the FILE was found in C:

Example Which list

See also Resident

Why

Usage: Why

Synopsis	(N/A)
Location	Sys:C
Function	Print additional information why an operation failed. Ordinarily when a command fails a brief message is printed that typically includes the name of the command that failed but provides few details. Why fills in details related to the failed operation.
Example	<pre>>Dir ram:noexistingdir Could not get information for ram:noexistingdir >Why The last command failed, reason: Object not found</pre>
See also	Fault

Poseidon USB devices utilities

AROS:C directory includes tools for some USB device. They came with Poseidon when it was ported to AROS. While general handling of USB classes, devices and stack have been already covered in the general AROS command reference guide, here follow the list of these device-specific tools and commands.

DRadioTool

Usage: DRadioTool

Synopsis	ON/S,OFF/S,FREQ/K/N,SCAN/S,AUTO/S,SIGNAL/S,UNIT/N/K
Location	Sys:C
Function	Very simple shell tool to control a USB Radio manufactured by D-LINK or GemTek. Only radios with Vendor ID = 0x04b4 and Product ID = 0x1002 are supported.
Inputs	ON/S - turns the radio on. OFF/S - turns the radio off again. FREQ/K/N - sets the current frequency to the given value. It must be given in KHz and range between 87 MHz and 108 MHz. SCAN/S - starts a frequency scan. It starts at 87 MHz, if no FREQ value had been given and stops at 107 MHz. If a radio channel is detected it will output its frequency in KHz on the shell. The last found channel will be kept. The scan can be aborted at any time using Ctrl-C. AUTO/S - only useful in conjunction with the SCAN switch. If a station is found, the program will pause for three seconds, asking the user to press Ctrl-C to keep the radio station found. SIGNAL/S - sets the shell return value to WARN (5), if no radio station is detected on the current frequency. If there's a stereo signal, it will return OK (0). This switch can be used to implement a manual scan routine. UNIT/N/K - if multiple radios are connected, you can choose the right unit with this argument. Defaults to unit 0 of course.
Example	>DRadioTool ON SCAN AUTO >DRadioTool FREQ 104000

PencamTool

Usage: PencamTool

Synopsis	TO/A,PICNUM/N,INTERVAL/N,UPTO/N/K,NOBEEP/S,GAMMA/K,SHARPEN/S,TEXT/K,FONT/K,Fontsize/N/K,UNIT/N/K
Location	Sys:C

Function Command line tool to read out images from a USB webcam using the STV680 chip (Vendor ID = 0x0553, Product ID = 0x0202). This includes the Aiptek Pencam series as well as a few more cheap cameras out there. Images are saved as true colour graphics in the Portable Anymap format (PPM), see NetPBM package on Aminet for a lot of conversion tools.

Moreover, gamma correction and white balance may be applied to the picture as well as a sharpening filter. Optionally, text may be pasted directly into the picture using a user definable font.

Inputs TO/A - mandatory filename to save the picture to. This filename may also contain a format string such as "%ld" (do not forget the 'l') to generate a number when using the INTERVAL option.
PICNUM/N - number of the picture to load from the camera's RAM, starting with 0 for the first picture. If no picture exist with this number, you will get garbage. Omitting this parameter will take a current snapshot.
INTERVAL/N - if this numeric parameter is given, PencamTool will loop and take pictures at the given interval (in ticks, 50 ticks is one second). Interval is only sensible, if you don't use the PICNUM argument. Use Ctrl-C to abort the PencamTool.
UPTO/N/K - if specified, multiple pictures can be grabbed in one go, starting at the PICNUM number and stopping at the UPTO number. Be sure to give a format string such as "%ld" inside the filename or you will write all pictures to the same image. If no PICNUM is given, but INTERVAL instead, UPTO describes the image number to stop the regular picture taking.
NOBEEP/S - disable BEEP on downloading an image.
GAMMA/K - enable white balance and gamma correction with the given floating point gamma value. 0.45 is a good setting. If you only want white balance and no gamma correction, use a value of 1.0.
SHARPEN/S - apply a highly optimized 5x5 sharpen filter on the image.
TEXT/K - optionally adds the given line of text to the bottom of the picture. If the line is too long to fit, it will be truncated.
FONT/K - name of the font to use (e.g. xen.font). If this parameter is missing, the default system font will be used.
FONTSIZE/N/K - size of the font in pixels
UNIT/N/K - if several cameras are connected, specify the unit to use. Defaults to unit 0.

Example

```
PencamTool Snap.ppm
PencamTool Snap.ppm 0 GAMMA 0.45 SHARPEN
PencamTool Movie%04ld.ppm INTERVAL 0 GAMMA 0.5
PencamTool Webcam.ppm GAMMA 0.45 SHARPEN TEXT
"Platon's Cam"
        FONT small.font FONTSIZE 6 NOBEEP
PencamTool Shotseries%03ld.ppm 0 UPTO 79 GAMMA 0.45
SHARPEN
```

PowManTool

Usage: PowManTool

Synopsis SOCKET=OUTLET/N,ON/S,OFF/S,TOGGLE/S,STATUS/S,UNIT/N/K

Location Sys:C

Function Very simple shell tool to control GemBird SIS-PM Silvershield PowerManager sockets/outlets. Only devices with Vendor ID = 0x04b4 and Product ID = 0xfd11 are supported.

Note that the devices will mistakenly report as HID device, but they don't actually speak HID conformant commands (which is a pity). On the first launch of PowManTool, any HID binding will be removed automatically.

Inputs SOCKET=OUTLET/N - number of the socket to change. If not given, command applies to all four sockets.
ON/S - enable power on the given socket(s), or when used with STATUS/S, return OK when socket is powered, WARN otherwise.
OFF/S - disable power on the given socket(s), or when used with STATUS/S, return WARN when socket is powered, OK otherwise.
TOGGLE/S - switches power off, if it was powered before, or on, if it wasn't. When used with STATUS/S, it will first toggle the socket(s), and then check, if it or they are now on (see ON/S).
STATUS/S - checks the status of one or more sockets. Normally returns WARN, if one or more sockets are turned off. OFF/S inverts this check.
UNIT/N/K - if multiple power managers are connected, you can choose the right unit with this argument. Defaults to unit 0 of course.

Example >PowManTool 1 ON
 Turns socket 1 on

 >PowManTool 2 OFF
 Turns socket 2 off

 >PowManTool 3 TOGGLE
 Toggles power at socket 3

 >PowManTool OFF
 Turns all sockets off

 >PowManTool 4 STATUS
 Returns WARN if socket 4 is not powered.

 >PowManTool OFF STATUS
 Returns WARN if any socket is powered

RocketTool

Usage: `RocketTool`

Synopsis `LEFT/S,RIGHT/S,UP/S,DOWN/S,FIRE/S,TIME/N/K,JOYPORT/N/K,UNIT/N/K`

Location `Sys:C`

Function Very simple shell tool to control a USB Rocket or Missile Launcher available from various sources on the internet. Only weapons of mass destruction with Vendor ID = 0x1130 and Product ID = 0x0202 are supported.

Note that the devices will mistakenly report as HID device, but they don't actually speak HID conformant commands (which is a pity). On the first launch of RocketTool, any HID binding will be removed automatically.

Inputs `LEFT/S` - turn the rocket pad to the left for some time.
`RIGHT/S` - turn the rocket pad to the right for some time.
`UP/S` - change the pitch up.
`DOWN/S` - aim lower.
`FIRE/S` - launch one of the three missiles.
`TIME/N/K` - optionally give the time of the movement in ticks (one tick is 1/50sec).
`JOYPORT/N/K` - instead of giving the direction, connect controls to the joystick or joypad at the given port. Of course, you can also use a USB joypad for this job. To exit, press Ctrl-C.
`UNIT/N/K` - if multiple rocket launchers are connected, you can choose the right unit with this argument. Defaults to unit 0 of course.

Examples `>RocketTool LEFT TIME 100`

`>RocketTool RIGHT UP 50 FIRE`

`>RocketTool FIRE`

`>RocketTool JOYPORT 1`

APENDIX I – Printer Escape Sequences (or commands)

For example:

Text in Italics

Example: Echo "`*E[3mItalics*E[23m`"

Purpose: `*E[3m` turns on italics and `*E[23m` turns off italics

Text in Bold

Example: Echo "`*E[1mBold*E[22m`"

Purpose: `*E[1m` turns on bold, and `*E[22m` turns off bold.

Underline Text

Example: Echo "`*E[4mUnderline*E[24m`"

Purpose: `*E[4m` turns on underline, and `*E[24m` turns off underline.

Coloured Text

Example: Echo "`*E[32mRed Text*E[0m`"

Purpose: `*E[nm` where `n=30-39` for foreground color or `n=40-49` for background colour.

`*E[0m` resets to normal character set.

Appendix II – AROS-DOS Error messages

#	Description	#	Description
103	not enough memory available	219	seek failure
105	process table full	220	comment is too long
114	bad template	221	disk is full
115	bad number	222	object is protected from deletion
116	required argument missing	223	file is write protected
117	value after keyword missing	224	file is read protected
118	wrong number of arguments	225	not a valid DOS disk
119	unmatched quotes	226	no disk in drive
120	argument line invalid or too long	232	no more entries in directory
121	file is not executable	233	object is soft link
122	invalid resident library	234	object is linked
202	object is in use	235	bad loadfile hunk
203	object already exists	236	function not implemented
204	directory not found	240	record not locked
205	object not found	241	record lock collision
206	invalid window description	242	record lock timeout
207	object too large	243	record unlock error
209	packet request type unknown	303	buffer overflow
210	object name invalid	304	***Break
211	invalid object lock	305	file not executable
212	object is not of required type		
213	disk is not validated		
214	disk is write-protected		
215	rename across devices attempted		
216	directory not empty		
217	too many levels		
218	device (or volume) is not mounted		

