itemcontainer	A generic closure toolcontainer
misc	Misc tool
os	Functions to perform system-specific operations
plugins	attach code to tools at runtime
printer	Print files
sh	A persistent shell client
stopwatch	stopwatch style timer functions
substitute	Functions to substitute Glish variables and expressions
sysinfo	Miscellaneous information
timer	Execute functions periodically, wait for a specified time

Functions

about	Information about AIPS++
aips2loc	Plot lines of code in AIPS++
ask	Displays the AIPS++ question submission form
bug	Provides a GUI to for user's to report problems.
edit	Edit and optionally execute a file
executivetest	Perform a test of executive
faq	Drives the user's web browser to the AIPS++FAQ
ftptest	Test the ftp tool
help	Provides user help.
is_itemcontainer	Is this an itemcontainer tool?
make	Function for a production system
note	Send a message to the logger or terminal
register	Provides a GUI to for user's to register AIPS++.
serverexists	Does the specified server exist
throw	Send a SEVERE message, then fail
web	Drives a web browser to the last requested help

1.6.1 misc - Tool

Package utility
Module misc

Misc tool

Description
The misc tool contains a number of string, file, and miscellaneous functions.
NOTE Some (limited) functions emulating C-based input and output routines are available. These have been replaced by functions native to Glish. We strongly recommend that you use the Glish functions instead.

Some of these functions are implemented completely in Glish without using the gmisc client. They remain in this tool for now. These functions are indicated in the individual descriptions.
This uses a server process misc which is started when first needed and then automatically restarted if it fails (hopefully never!).

Constructors

misc	Construct a misc tool

Functions

dir	Return the directory for a given directory name
escapespecial	escape characters in a string special to the shell
fclose	close fileid
fgets	interface to fgets
fields	list fields in a record or tool
fileexists	Does this file exist?
filetype	returns the UNIX type of file
fopen	interface to c fopen
fprintf	C-style formatted output
initspinner	start an ascii activity spinner
killspinner	kill an ascii activity spinner
listfields	Hierarchically list fields in a record
parentdir	Return the parent directory
patternmatch	Find matching strings using shell expressions
readfile	reads a file into a buffer
shellcmd	Execute a shell command
stripleadingblanks	Strip leading blanks from a string
striptrailingblanks	Strip trailing blanks from a string
thisdir	Return the name of this directory
tovector	Convert variable to a vector
type	Returns tool type; “misc”

misc - Constructor

Package	**utility**
Module	**misc**
Tool	**misc**

Construct a misc tool

Synopsis
misc()

A misc tool is created by the global constructor function misc. A default misc tool called defaultmisc is created automatically on inclusion of misc.g. A convenient shorthand name for defaultmisc is dms. Normally the defaultmisc tool is all that you'll need.

---

misc.type - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Returns tool type; "misc"

Synopsis
type()

Description
Returns the type of tool, in this case, "misc".

Returns
String

Example

```
- dms.type()
misc
```

---

misc.shellcmd - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Execute a shell command

Synopsis
shellcmd(command, log)

Description

Execute a shell command, logging results optionally. The Glish function `shell()` can also be used directly but it does not split lines or log. This function does not use the gmisc client, just the Glish shell client.

Arguments

command	Command string Allowed:String
log	Log the output? Allowed:Bool Default: T

Example

```
- dms.shellcmd('ls -l', log=F)
total 54
lrwxrwxrwx   1 root     bin            31 Jul 29 16:45 RCS -> /aips++/daily/rcs/trial/
drwxr-xr-x   2 tcornwel staff         512 Aug 29 15:11 aips++.log.table
-r--r--r--   1 tcornwel staff         256 Aug  1 21:31 makefile
-rw-r--r--   1 tcornwel staff        1556 Aug 29 09:28 ms.cc
-r--r--r--   1 tcornwel staff        7275 Aug 29 09:29 ms.g
-rw-r--r--   1 tcornwel staff        8889 Aug 29 09:28 ms.help
-rw-r--r--   1 tcornwel staff       22235 Aug 29 09:28 msImpl.cc
-rw-r--r--   1 tcornwel staff        4200 Aug 29 09:28 msImpl.h
-rw-r--r--   1 tcornwel staff         269 Aug  1 21:41 templates
drwxrwsr-x   2 tcornwel staff         512 Aug  1 21:41 tmplinst
T
- shell('ls -l')
total 54 lrwxrwxrwx   1 root     bin            31 Jul 29 16:45 RCS -> /aips++/daily/r

cornwel staff        1556 Aug 29 09:28 ms.cc -r--r--r--   1 tcornwel staff        7275

l staff        4200 Aug 29 09:28 msImpl.h -rw-r--r--   1 tcornwel staff        3059 Ju
```

Returns

Bool

---

misc.escapespecial - Function

Package	**utility**
Module	**misc**
Tool	**misc**

escape characters in a string special to the shell

Synopsis

escapespecial(string)

Description

This simply replaces the characters `*`, `#`, `(`, `)`, and `|`, by the sequences `\*` – `\|`, respectively. This is mainly used on strings that are to be passed to the shell. In a compromise between speed and completeness, this is only a subset of all possible significant characters, but it covers the most commonly problematic cases. There is no escape mechanism to avoid the filtering. This uses the builtin regex substitution capabilities of Glish.

Arguments

string	String Allowed:String	

Returns

String

Example

```
- dms.escapespecial('abc def ghi jkl mno')
abc def ghi jkl mno
- dms.escapespecial('a#c d*f g(i j)l m|o')
a\#c d\*f g\(i j\)l m\|o
-
+ files := shell('ls -1 A*foo')
- files
A*foo Anotherfoo
-
+ for (f in files) { print shell(spaste('ls -l ',f)) }
-rw-rw-r--   1 dbriggs  visitor       5 Dec 19 18:02 A*foo
-rw-rw-r--   1 dbriggs  visitor       4 Dec 19 18:16 Anotherfoo
-rw-rw-r--   1 dbriggs  visitor       4 Dec 19 18:16 Anotherfoo
-
+ for (f in files) { print shell(spaste('ls -l ',dms.escapespecial(f))) }
-rw-rw-r--   1 dbriggs  visitor       5 Dec 19 18:02 A*foo
-rw-rw-r--   1 dbriggs  visitor       4 Dec 19 18:16 Anotherfoo
```

The * in the filename, which we don't know about a priori, has confused the shell. This is cured simply by wrapping all unknown strings in dms.escapespecial() before passing them to the shell.

---

misc.striptrailingblanks - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Strip trailing blanks from a string

Synopsis
striptrailingblanks(string)

Description
This uses the builtin regex substitution and not the gmisc client.

Arguments

string	String to be stripped Allowed:String

Returns
String

Example

```
- dms.striptrailingblanks("foo         ")
foo
```

---

misc.stripleadingblanks - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Strip leading blanks from a string

Synopsis
stripleadingblanks(string)

Description
This uses the builtin regex substitution and not the gmisc client.

Arguments

string	String to be stripped Allowed:String

Returns
String

Example

```
- dms.stripleadingblanks("       foo")
foo
```

---

misc.patternmatch - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Find matching strings using shell expressions

Synopsis
patternmatch(pattern, strings)

Description
This pattern matching function uses the shell expression syntax familiar to anyone who has used a UNIX shell. In brief:

***** Matches any string, including the null string.

? Matches any single character.

Note that `[...]` matches any one of the enclosed characters. A pair of characters separated by - matches any character lexically between the pair, inclusive. If the first character following the opening `[` is a !, any character not enclosed is matched.
The following characters have a special meaning to the shell and cause termination of a word unless quoted:

`;  &  (  )  |  ^  <  >  newline  space  tab`

A character may be quoted (that is, made to stand for itself) by preceding it with a backslash (\) or inserting it between a pair of quote marks (" or "").
For further definition of the syntax, see the UNIX man page for sh. This is a simple syntax that suffices for many applications. If a more capable matching syntax is needed, use the builtin regex capabilities of Glish.
This is implemented entirely in Glish and not the gmisc client.

Arguments

pattern	Pattern Allowed:Any shell expression
strings	Strings to be tested Allowed:String

Example

```
- dms.patternmatch ('12**', "12345 89712 123 12 11")
12345 123 12
```

---

misc.fields - Function

Package	**utility**
Module	**misc**
Tool	**misc**

list fields in a record or tool

Synopsis
fields(rec)

Description
This function is similar to the glish function field_names, except that it lists the names of fields in a human-readable format.

Arguments

rec	record name Allowed:record

Example

```
- pg := pgplotter()
- dms.fields(pg)
activatecallback    hist                qcol                settings            addredrawfunctio
```

---

misc.fileexists - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Does this file exist?

Synopsis
fileexists(file, opt)

Description
This provides a simple interface to the UNIX test command for testing for the existence of files. See the man page for test for a definition of the options that can be used to test for particular file types. The default test is to see if a file exists and has a size greater than zero.

Arguments

file	file name Allowed:String
opt	option for UNIX test command Allowed:String Default: -s

Example

```
- dms.fileexists('foo')
F
- dms.fileexists('.')
T
- dms.fileexists('.', '-f')
F
```

---

misc.dir - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Return the directory for a given directory name

Synopsis

dir(directoryname)

Description

This returns a set of strings containing the files in the specified directory. Note that the directory name '.' is always returned.

Arguments

directoryname	Directory name Allowed:String

Example

```
- dms.dir('.')
. tmplinst ms.help makefile RCS templates ms.cc ms.g msImpl.cc msImpl.h aips++.log.tab
- dms.dir('RCS')
RCS UVFitsWriter.h,v makefile,v ms.cc,v ms.g,v ms.help,v msImpl.cc,v templates,v msImp
```

---

misc.parentdir - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Return the parent directory

Synopsis
parentdir(directoryname)

Description
This returns the name of the parent directory.

Arguments

directoryname	Directory name Allowed:String

Example

```
- dms.parentdir('/home/tarzan/tcornwel')
/home/tarzan
- dms.parentdir('/home/tarzan/tcornwel/aips++')
/home/tarzan/tcornwel
- dms.parentdir('.')         # If current directory is /home/tarzan/tcornwel
/home/tarzan
```

---

misc.thisdir - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Return the name of this directory

Synopsis
thisdir(directoryname)

Description
This returns the full name of this directory.

Arguments

directoryname	Directory name Allowed:String

Example

```
- dms.thisdir('.')
/home/tarzan/tcornwel
```

---

misc.filetype - Function

Package	**utility**
Module	**misc**
Tool	**misc**

returns the UNIX type of file

Synopsis
filetype(filename)

Arguments

filename	File name Allowed:String

Example

```
- dms.filetype('.')
Directory
- dms.filetype('ms.cc')
Regular File
```

---

misc.fopen - Function

Package	**utility**
Module	**misc**
Tool	**misc**

interface to c fopen

Synopsis
fopen(filename, mode)

Description
This function and those following emulate the low-level input/output functions available in C. We expect to replace these in the near future.
This function opens a file, returning a file id (an integer) that must be used to close the file.

Arguments

filename	File name Allowed:String
mode	UNIX mode of access Allowed:Usually r, w, or a

Returns
fileid

Example

```
- bugReport := spaste('/tmp/bugreport.',emailID)
- fp := dms.fopen(bugReport, 'w')
- dms.fclose(fp)
T
```

In the example, a filename found in the string variable bugReport is opened for writing.

---

misc.fclose - Function

Package	**utility**
Module	**misc**
Tool	**misc**

close fileid

Synopsis
fclose(fileid)

Arguments

fileid	File id Allowed:file id as returned by fopen

Returns
fileid

Example

```
- fp := dms.fopen('myfile', 'w')
# Do some stuff here
- dms.fclose(fp)
T
```

---

misc.fprintf - Function

Package	**utility**
Module	**misc**
Tool	**misc**

C-style formatted output

Synopsis
fprintf(fileid, format, ...)

Description
This and the following routines provide input and output formatting capabilities similar to those provided in C libraries. See the UNIX man page on *e.g.* fprintf for more information. Differences from the C routines are described.
The ellipsis ... denotes that any number of arguments may be provided.

Arguments

fileid	File id Allowed:file id as returned by fopen
format	c-style format description Allowed:String
...	Additional arguments Allowed:Strings, floats, ints, etc...

Example

```
- fp := dms.fopen('myfile', 'w')
- sourcename := 'mySource'
- hour := 12
- minute := 4
- seconds := 42.3
- dms.fprintf(fp, '%s %02d:%02d:%05.2f\n', sourcename, hour, minute, seconds)
- dms.fclose(fp)
T
```

In this example you'd get the following in myfile
mySource 12:04:42.30

---

misc.fgets - Function

Package	**utility**
Module	**misc**
Tool	**misc**

interface to fgets

Synopsis
fgets(fileid, filler)

Description
Get a string buffer from a file. fgets is similar to the C function fgets but different in a couple of key ways. First it reads up to 500 characters or until a new line is reached in the file. Unfortunately 500 is hard-coded (the current implementation requires this, you don't want to know). Second, if you supply a fill character it will replace blank characters with your fill character.

Arguments

fileid	File id Allowed:file id as returned by fopen
filler	string buffer to replace blanks Allowed:String Default: ”

Returns
Contents of current line in the file

Example

```
- fp := dms.fopen('myfile', 'r')
- a := dms.fgets(fp)
- print a
mySource 12:04:42.30
```

dms.fgets(fp) will return 0 when it reaches the end of file.

---

misc.readfile - Function

Package	**utility**
Module	**misc**
Tool	**misc**

reads a file into a buffer

Synopsis
readfile(file)

Description
The file is read into a buffer. This is useful for reading binary files, for example.

Arguments

file	File name Allowed:String

Example

```
- buff:=dms.readfile("/export/home/estella/AIPS/DATA/CAD000000.0B4;")
```

---

misc.fopen - Function

Package	**utility**
Module	**misc**
Tool	**misc**

interface to c fopen

Synopsis

fopen(filename, mode)

Description

This function and those following emulate the low-level input/output functions available in C. We expect to replace these in the near future.
This function opens a file, returning a file id (an integer) that must be used to close the file.

Arguments

filename	File name Allowed:String
mode	UNIX mode of access Allowed:Usually r, w, or a

Returns

fileid

Example

```
- bugReport := spaste('/tmp/bugreport.',emailID)
- fp := dms.fopen(bugReport, 'w')
- dms.fclose(fp)
T
```

In the example, a filename found in the string variable bugReport is opened for writing.

---

misc.initspinner - Function

Package	**utility**
Module	**misc**
Tool	**misc**

start an ascii activity spinner

Synopsis
initspinner(interval)

Description
This initializes a traditional ascii activity spinner. The characters `|`, `/`, `-`, and `\` are printed periodically, followed by a backspace. The printing is done outside the standard Glish output mechanism, resulting in a spinning bar. If any Glish output is done during this time, orphaned characters will be left over as the cursor moves. In particular, this will probably confuse Glish's notion of where the cursor is during editing. It is probably best not to leave this active while the user is expected to type anything. Use killspinner to shut this down. As there should only be one of these things active at once, there can only be one spinner per misc tool, and the spinner argument is implicit.

Arguments

interval	Interval in seconds between characters Allowed:Real

---

misc.killspinner - Function

Package	**utility**
Module	**misc**
Tool	**misc**

kill an ascii activity spinner

Synopsis

killspinner()

Description
This shuts down the spinner as started by initspinner.

---

misc.listfields - Function

Package	**utility**
Module	**misc**
Tool	**misc**

Hierarchically list fields in a record

Synopsis
listfields(rec, listdata, depth)

Arguments

rec	Record to list Allowed:Glish record
listdata	List data at bottom of record Allowed:Boolean Default: T
depth	Depth in record Allowed:Integer Default: 0

Returns
T

Description
This function hiearchically lists the contents of a record by drawing a tree and optionally printing the data at the end of each branch.
If you choose not to list the data at the end of the branch, then its type will be printed for you instead.
The `depth` argument is just used to offset the printed output as it works its way down the record. If you put some other value, other than 0, then the text will be offset to the right differently. You shouldn't need to use it.

Example

```
r := [=]
r.x := [=]
r.y := [=];
r.x.a := [1,2,3]
r.x.b := 'trout'
r.y.a := [10,20,30]
r.y.b := 'salmon'
r.y.c := [=]
r.y.c.i := 1
r.y.c.k := [=]
r.y.c.k.x := [1,2,3,4]
r.y.c.k.y := -400
r.y.c.j := 1
#
dms.listfields(r)
  x
    a = [1 2 3]
    b = trout
  y
    a = [10 20 30]
    b = salmon
    c
      i = 1
      k
        x = [1 2 3 4]
        y = -400
      j = 1
#
dms.listfields(r, listdata=F)
  x
    a = integer [3]
    b = string
  y
    a = integer [3]
    b = string
    c
      i = integer
      k
        x = integer [4]
        y = integer
      j = integer
```

---

misc.tovector - Function

Package	utility
Module	misc
Tool	misc

Convert variable to a vector

Synopsis
tovector(thing, type, unsetvalue)

Arguments

thing	Variable to convert Allowed:Any Glish variable
type	Specify the output type of the vector Allowed:String chosen from 'string', 'float', 'double', 'integer' Default: 'string'
unsetvalue	Unset value for unset items in records Allowed:Numeric value Default: unset

Returns
Vector of given type or fail

Description
This function converts a Glish variable to a vector of the given type. It is a bit specialized, and ultimately this functionality should be moved to a central parsing location like entryparser.g
Its purpose is so that a vector variable can be specified as a vector, a string, or a vector of strings. Thus no matter how the vector is entered to Glish, the right thing comes back. It is really meant for numeric and string vectors. If you put in what you want out (e.g. input a float vector and ask for the output to be a float vector) that is ok. If the input is a string, you can use space or comma or both delimiters.
After attempting to convert the input (argument `thing`) to a vector, it is type converted to the specified type.
The `unsetvalue` argument is only relevant if the input `thing` is a record. Then, any fields in the record that are unset, are replaced by `unsetvalue`. If `unsetvalue=unset` and a record field is unset, then an exception will be thrown. The record fields are then converted to vector of the appropriate type.