SUN/234.5

Starlink Project
Starlink User Note 234.5

R.T.Platon & T. Jenness

3rd December 2013

ONE – Odds and Ends Library
Version 1.4
User's Guide

Abstract

This library is a set of Fortran and C routines of a general nature and usefulness, which are not suitable for inclusion in other more focussed libraries.

Contents

List of Figures

1 Introduction

The Odds and Ends (ONE) library is a collection of miscellaneous C and Fortran routines called from various packages in the Starlink Software Collection. In general these routines have been used in more than one package, so are suitable to be included in a public library, but are not suitable for inclusion in one of the other public libraries of more focussed routines.

2 List of routines

These routines were collected by examining possible entries and by a request for contributions. Extra routines may be added when necessary.

The contents of the library are (in alphabetical order):

ONE_EXEC(COMMAND, STATUS)
Executes a shell command. Extracted from CCDPACK (submitted by P. Draper).

ONE_FIND_FILE(FILESPEC, LISDIR, FILENAME, CONTEXT, STATUS)
Returns successive file names that match a file specification. Taken from FIGARO, SST, CONVERT, KAPPA and CCDPACK.

ONE_FIND_FILE_END(CONTEXT, STATUS)
Terminate a sequence of FIND_FILE calls. Taken from FIGARO, SST, CONVERT, KAPPA and CCDPACK.

ONE_SCRSZ(WIDTH, HEIGHT, STATUS)
A Fortran callable function to obtain the size of the output screen. Taken from CONVERT, KAPPA and IRCAMPACK.

ONE_SHELL_ECHO(FILESPEC, FILENAME, STATUS)
Expands shell metacharacters in a string without file globbing.

one_snprintf(str, size, format, status, ...)
Starlink compliant wrapper around the standard snprintf function.

one_strlcat(dest, src, sizedest, status)
Wrapper around the standard BSD strlcat C function (safer variant of strncat).

one_strlcpy(dest, src, sizedest, status)
Wrapper around the standard BSD strlcpy C function (safer variant of strncpy).

one_strtod(instr, status)
Extract a double precision value from a C string.

ONE_WORDEXP_FILE(WORDS, CONTEXT, EXPAN, STATUS)
WORDEXP wrapper that verifies a file exists.

ONE_WORDEXP_NOGLOB(WORDS, EXPAN, STATUS)
WORDEXP wrapper that expands shell variables without globbing.

3 Routines descriptions

There follows a full description of the library routines in alphabetical order:

ONE_EXEC
Executes a shell command

Description:

This routine gives FORTRAN programs access to the UNIX system(3) command. See the system(3) man page for how to construct a suitable command.

Invocation:

```
CALL ONE_EXEC(COMMAND, STATUS)
```

Arguments:

COMMAND = CHARACTER * (*) (Given)

The command to be executed.

STATUS = INTEGER (Given and Returned)

The global status. Set to SAI__ERROR if command fails.

ONE_FIND_FILE
Returns successive file names that match a file specification

Description:

This routine is intended to provide some of the faciltes provided on a VAX by the standard VMS routine LIB$FIND_FILE. It is passed a file specification that can contain wild card characters such as '*', eg '*.*'. On the first call the Context variable should be set to zero, and the routine will return the name of the first file that matches the file specification. On subsequent calls the calling routine should continue to call using the value of Context returned by the previous call, and each call will return the name of the next file that matches the specification. When the last file that matches the specification has been returned, subsequent calls will return a blank string as the file name and an error code (an even value) as the function value. Finally, a call to ONE_FIND_FILE_END with the last returned value of Context will close down any files opened or memory allocated by ONE_FIND_FILE.

Invocation:

```
FOUND = ONE_FIND_FILE (FILESPEC,LISDIR,FILENAME,CONTEXT,STATUS)
```

Arguments:

FILESPEC = CHARACTER (Given)

The file specification to be matched. May contain wildcards. Case sensitive.

LISDIR = LOGICAL (Given)

TRUE if directory contents are to be listed for directories that match the file specification. Should be set to FALSE if matching directory names should be returned without opening the directories themselves. Note that even if true, this routine will not recurse into all subsubdirectories that match. To be more explicit: TRUE is equivalent to 'ls', FALSE is equivalent to 'ls -d'. Neither is equivalent to 'find . -name "filespec"'

FILENAME = CHARACTER (Returned)

The name of a file that matches FILESPEC.

CONTEXT = INTEGER (Given and Returned)

A variable used to remember the context of a series of calls for the same file specification. Should be set to zero by the caller for the first call for a new specification and the value then returned by this routine should be used in subsequent calls.

STATUS = INTEGER (Given and Returned)

A status code as follows

- SAI__OK for success
- ONE__NOFILES - No more files found
- ONE__LENGTHERR - Bad parameter length
- ONE__PIPEERR - Pipe error

- ONE__MALLOCERR - Malloc error

Returned Value:

ONE_FIND_FILE = LOGICAL
TRUE if File Found FALSE if error or no more files

Notes:

This routine returns bad status (ONE__NOFILES) even when the status is not technically bad. In general, the caller should annul this particular status condition before proceeding.

Bugs:

This routine does not provide all the facilities offered by the original VAX version; it only accepts the first three arguments as listed above, and almost of necessity it uses and expects UNIX syntax for the file specs. This means that '/usr/users/ks/*.*' is OK, but '[ks]*.*' is not. Note that '*' and '*.*' will give quite different results under UNIX, whereas under VMS they would be the same. There is no way of specifying recursion; '/usr/user/ks/...' for example is meaningless. Nevertheless, it is hoped that it is close enough in functionality to the VMS original to act as a useable substitute in most cases. It cannot handle specifications that the standard shell (sh) cannot handle in an 'ls' command - there are some variations of the 'ls' command involving complex wildcarding that will cause sh on a SUN to hang, and they will also hang this routine.

It is not at all clear that the method used here is in any way the best solution to the problem, and there are a number of possible alternatives that could be tried, including using 'find' rather than 'ls', or using routines such as readdir() to search the file system and do any pattern matching in this routine itself. The program as it stands should be regarded (tolerantly!) as an initial attempt and the author would be glad to be sent a better version.

ONE_FIND_FILE_END
Terminate a sequence of ONE_FIND_FILE calls

Description:

This routine should be called after a sequence of calls to ONE_FIND_FILE in order to release any resources used by ONE_FIND_FILE. It should be passed in its Context argument the value of the Context argument as returned by the ONE_FIND_FILE in the last call in the sequence that is to be closed down.

Invocation:

```
CALL ONE_FIND_FILE_END (CONTEXT, STATUS )
```

Arguments:

CONTEXT = INTEGER (Given)

The context argument returned by the last call to ONE_FIND_FILE in the sequence to be closed down.

STATUS = INTEGER (Given and Returned)

Inherited status. Routine will attempt to free resources even if status is bad on entry.

ONE_SCRSZ
A Fortran callable function to obtain the size of the output screen

Description:

This routine interrogates the system to find the width and height of the screen on which it is running. Should an error occur or the width is not positive, set to the default of 80 characters by 0 lines.

Invocation:

```
CALL ONE_SCRSZ( WIDTH, HEIGHT, STATUS )
```

Arguments:

WIDTH = INTEGER (Returned)
The width of the screen in characters. (default 80)

HEIGHT = INTEGER (Returned)
The height of the screen in lines. (default 0)

STATUS = INTEGER (Given and Returned)
The global status. Set to SAI__ERROR if an error occurs..

Notes:

This is the UNIX version.

ONE_SHELL_ECHO
Interpret shell metacharacters in a string, without file globbing.

Description:

This routine is intended to expand shell metacharacters within a supplied file name, in the case where the file may not already exist. Any wild card characters within the string are ignored (i.e. there is no file globbing).

Invocation:

```
CALL ONE_SHELL_ECHO( FILESPEC, FILENAME, STATUS )
```

Arguments:

FILESPEC = CHARACTER (Given)

The file specification to be echoed.

FILENAME = CHARACTER (Returned)

The result of expanding any shell metacharacters (except wild cards) within FILESPEC.

STATUS = INTEGER (Given and Returned)

A status code as follows

- SAI__OK for success
- ONE__LENGTHERR - Bad parameter length
- ONE__PIPEERR - Pipe error
- ONE__MALLOCERR - Malloc error

one_snprintf
Starlink compliant wrapper around the standard snprintf function

Description:

The one_snprintf function replaces all calls to sprintf() and snprintf() to allow the use of inherited status and to trap for truncation.

Invocation:

```
retval = one_snprintf( char * str, size_t size, const char * format, int * status,
... );
```

Arguments:

str = char * (Returned)
Buffer to be filled from the format statement. Must be at least "size" characters in length.

size = size_t (Given)
Allocated size of buffer "str".

format = const char * (Given)
Standard format string as expected by snprintf.

status = int * (Given and Returned)
Inherited status. Will be set to ONE__TRUNC if the formatted version of the string does not fit into the buffer "str".

... (Given)
Variadic arguments required by snprintf.

Returned Value:

int retval
Length of the string after appending. If the value exceeds the size of the supplied buffer status will be set to ONE__TRUNC but the value returned will indicate the size of the buffer that would be required to completely expand the formatted string.

Notes:

- This is for use from C only.
- Use this routine in place of snprintf and sprintf.

one_strlcat
Starlink compliant wrapper around the BSD strlcat function

Description:

The strlcat function is similar to the strncat function except that it guarantees to nul terminate the destination string and returns the number of characters that will have been copied. This wrapper function provides standard Starlink inherited status semantics.

Invocation:

```
len = one_strlcat( char * dest, const char * src, size_t sizedest, int * status
);
```

Arguments:

dest = char * (Returned)

Destination buffer for "src". Must be nul-terminated. If status is bad on entry ”dest” will not be touched.

src = const char * (Given)

String to be copied.

sizedest = size_t (Given)

The actual buffer size of "dest" including space for a nul.

status = int * (Given and Returned)

Inherited status. Will be set to ONE__TRUNC if the string was truncated on copy.

Returned Value:

size_t retval

Length of the string after appending. Will either be the length of the source string or one less than the size of the destination buffer.

Notes:

- This is for use from C only.
- If available the system strlcat routine will be used.

one_strlcpy
Starlink compliant wrapper around the BSD strlcpy function

Description:

The strlcpy function is similar to the strncpy function except that it guarantees to nul terminate the destination string and returns the number of characters that will have been copied. This wrapper function provides standard Starlink inherited status semantics.

Invocation:

```
len = one_strlcpy( char * dest, const char * src, size_t sizedest, int * status
);
```

Arguments:

dest = char * (Returned)

Destination buffer for "src". Will be nul-terminated. If status is bad on entry, dest will be nul-terminated if it is non-NULL.

src = const char * (Given)

String to be copied.

sizedest = size_t (Given)

The actual buffer size of "dest" including space for a nul.

status = int * (Given and Returned)

Inherited status. Will be set to ONE__TRUNC if the string was truncated on copy.

Returned Value:

size_t retval

Length of the string after copying. Will either be the length of the source string or one less than the size of the destination buffer.

Notes:

- This is for use from C only.
- If available the system strlcpy routine will be used.

one_strtod
Starlink compliant wrapper around the standard strtod function

Description:

The strtod() function converts a string to a double precision number. This function is the same except that there is not endptr argument and it will set status to bad on failure and use inherited status.

Invocation:

```
dval = one_strtod( const char * instr, int * status );
```

Arguments:

instr = const char * (Given)

Input string to parse. Must be nul-terminated standard C string suitable for the strtod() library call.

status = int * (Given and Returned)

Inherited status. Will be set to ONE__CNVERR if the string did not contain a number.

Returned Value:

double dval

Converted double precision value. VAL__BADD on error.

Notes:

- This is for use from C only.
- Fortran D format is supported so 5.2D5 is converted to 5.2E5 before involving strtod.

See Also :

- CHR_CTOD subroutine.

ONE_WORDEXP_FILE
WORDEXP wrapper that verifies a file exists

Description:

A wrapper around PSX_WORDEXP that only returns files that match the supplied WORDS string following expansion. EXPAN will be an empty string if there are no matches and CONECT will be set to 0.

Invocation:

```
CALL ONE_WORDEXP_FILE( WORDS, CONTEXT, EXPAN, STATUS )
```

Arguments:

WORDS = CHARACTER∗(∗) (Given)

The string to be shell expanded.

CONTEXT = INTEGER (Given & Returned)

Should be initialised to 0 for initial shell expansion. Will be set to zero when no more results are available.

EXPAN = CHARACTER∗(∗) (Given)

Expanded string. A new string will be returned each call until all are returned an CONTEXT is set to 0.

STATUS = INTEGER (Given & Returned)

The global status.

Notes:

- Continue to call this routine until CONTEXT is set to 0, otherwise there will be a possible memory leak.
- If there are multiple returns from PSX_WORDEXP that are not files this routine will loop internally until a valid file is located.

ONE_WORDEXP_NOGLOB
WORDEXP wrapper that expands shell variables without globbing

Description:

A wrapper around PSX_WORDEXP that does shell expansion without doing globbing. Glob wildcard characters are escaped, as are illegal, to wordexp(), shell metacharacters such as "(" and ")".

Only a single expansion string is returned by this routine and it is an error for wordexp() to return multiple matches.

Invocation:

```
CALL ONE_WORDEXP_NOGLOB( WORDS, EXPAN, STATUS )
```

Arguments:

WORDS = CHARACTER∗(∗) (Given)
The string to be shell expanded.

EXPAN = CHARACTER∗(∗) (Given)
Expanded string.

STATUS = INTEGER (Given & Returned)
The global status.

See Also :

ONE_SHELL_ECHO is similar except that it guarantees to fork whereas wordexp() might not fork.