SUN/111.2

Starlink Project
Starlink User Note 111.2

D L Terrett
P M Allan

1 February 1993

SPT — Software Porting Tools

Abstract

SPT is an inhomogeneous set of software tools designed to assist in the conversion of code from one operating system to another (principally from VMS to UNIX and vice-versa, and between different flavours of UNIX).

Contents

1 Introduction **1**

2 Running the Tools **1**
2.1 On UNIX 1
2.2 On VMS 1

3 FORCONV **1**

4 Softlink **3**
4.1 Logical names and soft links 3
4.2 Overview of softlink 4
4.3 Details 4
4.4 Defining Standard Soft Links 5

1 Introduction

SPT is an inhomogeneous set of software tools designed to assist in the conversion of code from one operating system to another (principally from VMS to UNIX and vice-versa, and between different flavours of UNIX).

The tools are:

FORCONV Converts file name specifications in Fortran INCLUDE statements and "escapes" backslash characters. The output file on VMS is also in Stream_lf format so can be read on a UNIX system using NFS.

SOFTLINK Manages the creation and removal of soft links on Unix systems. These soft links will generally point to Fortran INCLUDE files, thereby providing a machine-independent way of writing INCLUDE statements.

2 Running the Tools

2.1 On UNIX

The programs are stored in `/star/bin` so all that is required is that `/star/bin` is in your PATH. This is normally done when you log in.

2.2 On VMS

The tools must be defined as foreign DCL commands with the command:

```
$ SPT
```

Note that softlink is only available on Unix. Its effect is achieved on VMS by the use of logical names.

3 FORCONV

The principal job of FORCONV is to convert the file name specification in Fortran include statements but it will also do some other routine tasks that are necessary when converting a program from VMS to UNIX.

- Replace all backslash characters with two backslashes (UNIX compilers interpret a backslash as an "escape" character by default although on some systems there is a compiler switch to alter this behaviour).
- Generate a file containing a "make" dependency line for each include statement processed. This file can be used as a starting point for a make file.

- Insert a statement to include `dat_par` and `par_par` after every occurrence of the include file `sae_par`. This compensates for an incompatibility between `sae_par` on VMS and on UNIX.

The substitution of include file names is driven by a "substitution file" which lists the VMS file specifications and their equivalent UNIX file names, one pair per line. For example the file might contain:

```
PARS.FOR pars
INCLIB(COMMON) commons
```

and an example of the source code to be converted might be:

```
INCLUDE 'PARS.FOR'
INCLUDE 'INCLIB(COMMON)'
```

which, after processing, would be changed to:

```
INCLUDE 'pars'
INCLUDE 'commons'
```

The case of the VMS file specification is ignored when trying to find a match and spaces are ignored in the file specification in the INCLUDE statement.

If a match for an include file specification is not found then the name is folded to lower case and a prefix prepended; the default prefix is `/star/include/` but this can be changed with the `-p` command line option. So in the above example:

```
INCLUDE 'FRED'
```

would be converted to:

```
INCLUDE '/star/include/fred'
```

As a result of this, when converting programs that use Starlink libraries, the substitution file need only specify any "program-specific" include files; references to Starlink include files will all be converted automatically. If the program does not use Starlink libraries, then the prefix can be suppressed entirely by specifying the `-p` option with an empty string.

The program will not recognise INCLUDE statements that are continued on to a new line.

The command:

```
forconv
```

Will read from the standard input stream and write to the standard output stream, and will use `include.sub` as the name of the substitution file. The program's behaviour can be modified with the following UNIX-style command options:

`-i`***filename*** Read from *filename* instead of the standard input stream. Note that there is no space between the `i` and the file name in this or any other command options.

`-o`***filename*** Write to *filename* instead of the standard output stream.

`-s`***filename*** Use *filename* as the substitution file.

`-m`***filename*** Generate a skeleton make file. If *filename* is omitted a file called `makefile` will be written.

`-b`***name*** Use *name* as the target in the dependency rules written to the make file. In the absence of `-b` the name `module` is used.

`-p`***prefix*** Use *prefix* as the string to be inserted before all file specifications that did not match a file specification in the substitution file. The default prefix is `/star/include/`; `-p` on its own specifies no prefix.

`-h` Add

```
INCLUDE '/star/include/dat_par'
```

after every occurrence of `'/star/include/sae_par'`. This compensates for an incompatibility between the standard Starlink include file `sae_par` on VMS and on UNIX and is required if the program being converted uses any `DAT__` constants.

`-j` Add

```
INCLUDE '/star/include/par_par'
```

after every occurrence of `'/star/include/sae_par'`. This compensates for an incompatibility between the standard Starlink include file `sae_par` on VMS and on UNIX and is required if the program being converted uses any `PAR__` constants.

`-q` Suppress all informational messages. By default the program writes a message to the standard error stream each time it process an INCLUDE statement.

4 Softlink

Softlink is a tool to provide easy, portable use of standard Starlink include files on Unix systems. It is complementary to FORCONV and will be of general use when developing Fortran programs on Unix as well as in porting code to Unix.

4.1 Logical names and soft links

Starlink Fortran programs frequently make use of INCLUDE files to define symbolic constants or common blocks. When the INCLUDE file is one supplied by Starlink, a logical name is defined on VMS to point to the file. For example, the logical name `SAE_PAR` points to the file `LIB_COMMON_DIR:SAE_PAR.FOR`. This means that the file can be included in a Fortran program with the statement:

```
INCLUDE 'SAE_PAR'
```

There is no need to refer to the location of the file, just to its name. This makes the program more portable than referring to the explicit file.

Unix does not provide logical names, but you can achieve a similar effect by the use of soft links. A soft link looks like a file in a directory, but is actually just a pointer to another file. To use a soft link to refer to the INCLUDE file in the above example, you would type:

```
% ln -s /star/include/sae_par SAE_PAR
```

This command should be executed in the same directory as the source of the Fortran program. If you are working on Fortran programs in different directories, you will need the appropriate soft links in each directory. This is unlike VMS, where logical names exist independently of the file system. Note that the name of the file is in lower case, but the name in the include statement is in upper case. This distinction is important as Unix is case sensitive, unlike VMS.

4.2 Overview of softlink

Softlink is a Unix shell script that can be used to create and remove soft links to files. The purpose of writing it was to automate the setting up of soft links that point to standard Starlink Fortran include files. Rather than having programs contain lines like:

```
INCLUDE '/star/include/sae_par'
```

they will contain lines like:

```
INCLUDE 'SAE_PAR'
```

and a soft link to the file `/star/include/sae_par` will be created in the directory containing the source code. This makes the source code more portable; it may well run on Unix and VMS systems with no changes.

Before compiling a program on a Unix system, the soft links to the include files should be created. This need only be done once as the soft links are permanent. For the above example, you would type:

```
% softlink /star/include/sae_par upper
```

Note that the name of the file being pointed to must match the name of the file in the Fortran INCLUDE statement for softlink to be effective. If the names are different, then you should either set up the soft link by hand, or use FORCONV.

4.3 Details

Softlink accepts three arguments. Only the first one is mandatory. This specifies the full path name of the file that the soft link will point to. An example is `/star/include/sae_par`.

The second (optional) argument specifies whether the name of the soft link is in upper or lower case. If it is omitted (or given as the null string), then the name of the soft link matches exactly

the name of the file that it points to. The value of the argument can be one of 'lower', 'upper', 'both', 'all' or 'remove'. A value of 'lower' specifies that the name of the soft link will be in lower case and a value of 'upper' specifies that the name of the soft link will be in upper case. A value of 'both' specifies that two soft links are to be created, one in lower case and one in upper case. A value of 'all' specifies that links will be created in lower case, upper case and with the same name as the file that it is pointing to if that is not purely lower case or upper case. If the argument is 'remove', then all the soft links are removed. The script checks to see that it is only removing soft links and not files of the same name.

The third argument is normally absent. If it is present, then the script does not execute any commands to create soft links, but writes those command to the file named by the third argument. For example, the command

```
% softlink /star/include/par_par all fred
```

will create a file called `fred` containing the lines:

```
ln -s /star/include/par_par par_par
ln -s /star/include/par_par PAR_PAR
```

If the output file already exists, then the commands are appended to the file so that a set of link commands can be built up. If the name of the output file is `mfile`, then the commands are output in a form suitable for inclusion in a make file. In particular, the first character of each line is a tab character.

4.4 Defining Standard Soft Links

When writing programs that call Starlink libraries, you will find yourself setting up the same soft links over and over again. To ease this process, shell scripts are provided to define the soft links for all of the Starlink subroutine libraries. The scripts are called *lib*`_dev`, e.g. to set up the soft links needed when writing a program that calls FIO, just type:

```
% fio_dev
```

The soft links can be removed by typing:

```
% fio_dev remove
```

The scripts provide fewer functions than the softlink script. All links are created in upper case and the only option is `remove`, which does not check to see that the names referred to are soft link rather than files. To offset these restrictions, the *lib*`_dev` scripts run faster than they would if they provided all the features of softlink. The list of scripts is as follows:

Package	Script
AGI	agi_dev
ARY	ary_dev
CHR	chr_dev
EMS	ems_dev
ERR	err_dev
FIO	fio_dev
GKS	gks_dev
GNS	gns_dev
GRP	grp_dev
GWM	gwm_dev
HDS	hds_dev
IDI	idi_dev
NDF	ndf_dev
PAR	par_dev
PRIMDAT	prm_dev
PSX	psx_dev
REF	ref_dev
SGS	sgs_dev
TRANSFORM	trn_dev

There are also scripts `star_dev`, which creates the soft links to `SAE_PAR`, and `adam_dev`, which creates the soft links needed when writing programs that call the internals of ADAM.