

**NAME**

**assign** – Assign customizations to Fortran file I/O

**SYNOPSIS**

**assign** [*command-options*] [*customize-options*] [*filespec*]

**DESCRIPTION**

The **assign** command customizes the behavior of Fortran I/O for a certain set of files or logical units (for example, it can swap bytes when using big-endian files on a little-endian computer). By issuing the command several times, you can dictate different customizations for each of several sets of files or units.

The *command-options* relate to the command itself; the *customize-options* describe the desired behavior, and the *filespec* describes which files or logical units those customizations apply to. Each command allows multiple *customize-options*, but no more than one *command-option* and one *filespec*.

The command works by storing information into a file that the Fortran library inspects before performing I/O. You must set the environment variable FILENV to the name of that file before you type the **assign** command, and it must remain set when the program runs.

The **pathf95**(1) compiler also provides compile-time options for swapping bytes that are simpler to use. Setting FILENV overrides the compiled-in choices.

**FILESPEC**

The *filespec* can be:

**u:unit** The logical unit numbered *unit*.

**f:filename**

The file whose name is *filename*.

**p:pattern**

Every file whose name matches *pattern*, using the % symbol as a wildcard which matches zero or more characters.

**g:du** Every direct unformatted file (e.g., files opened using "access='direct' form='unformatted'").

**g:su** Every sequential unformatted file (e.g., files opened using "access='sequential' form='unformatted'").

**g:all** Every file.

**COMMAND OPTIONS**

**-R** Remove all customizations for the specified *filespec*. If no *filespec* is specified, remove all customizations.

**-I** Add this command to any previous ones. Normally each **assign** command erases the effect of all previous **assign** commands, even if they refer to a different set of files. To allow more than one command to take effect at the same time, all but the first command must use **-I**.

**-O** Erase the effect of previous **assign** commands before adding the current command. This is the default.

**-V** Display all **assign** commands now in effect.

**CUSTOMIZATION OPTIONS**

**-a name**

Use *name* as the actual file name for the file specified by *filespec*, overriding any name provided by the *FILE=* specifier in an *OPEN* statement.

**-F conversion** Convert the byte ordering of each record header

(a four-byte record length which appears at the beginning and end of each record) in a sequential unformatted file. It is allowed only with unformatted files. The option has no effect on the data itself (see the **-N** option for that). The *conversion* may be:

**f77.mips**

On a little-endian computer, use big-endian format for each record header.

**f77.vax**

On a big-endian computer, use little-endian format for each record header.

**-N conversion**

Convert the byte ordering of the data in an unformatted file. It is allowed only with unformatted files. For sequential unformatted files, which have record headers, you will usually want to use the **-F** option as well so that the record headers are also converted. The *conversion* may be:

**be** On a little-endian computer, use big-endian format for the data.

**ia64**

On a big-endian computer, use little-endian format for the data.

**-S onloff**

The **-S off** option generates commas in list-directed or namelist output on files specified by *filespec*. The default is **-S on**. For example, if the default output would be "4 5 6", **-S off** produces "4, 5, 6".

**-y onloff**

The **-y off** option allows the use of repetition factors in list-directed or namelist output on files specified by *filespec*. The default is **-y on**. For example, if the default output would be "4 4 4", **-y off** produces "3\*4".

**EXAMPLES**

Assign name *tmp* to unit 8, removing any previous **assign** commands whether or not they relate to unit 8:

```
% assign -a tmp u:8
```

Assign name *tmp* to unit 8 without removing any previous **assign** commands:

```
% assign -I -a tmp u:8
```

Convert numeric data from big-endian format when reading file *data*, and convert numeric data to big-endian format when writing that file (note that if the file is sequential, its record headers will remain in little-endian form unless you also use the **-F** option):

```
% assign -I -N be f:data
```

Allow repetition factors in list-directed output on any file whose name begins with **x** and ends with **y**:

```
% assign -I -y off p:x%y
```

For all direct unformatted files, convert numeric data from big-endian format on input, and generate big-endian format on output:

```
% assign -I -N be g:du
```

For all sequential unformatted files, convert data from big-endian format on input, and generate big-endian format on output ( the **-N** option); do likewise with the record headers (the **-F f77.mips** option):

```
% assign -I -N be -F f77.mips g:su
```

Allow the use of repetition factors on standard output (unit 6):

```
% assign -I -y off u:6
```

**USING ASSIGN WITH MPI**

When using **assign** with MPI, make sure that the environment variable **FILENV** is properly defined and that the **assign** file is accessible by each process. The **mpirun** program is not going to set the environment variable for you, so you need to have a wrapper-script that sets it.

You can define FILENV in your .cshrc or .bashrc file with these commands:

For bash:

**#The following will concatenate the line 'export FILENV=.assign' to the #end of your .bashrc. The \${HOME}/ is so that you can use this from any #directory.**

```
% echo "export FILENV=.assign" >> ${HOME}/.bashrc
```

For csh:

**#The following will concatenate the line 'setenv FILENV .assign' to the #end of your .cshrc. The \${HOME}/ is so that you can use this from any #directory**

```
% echo "setenv FILENV .assign" >> ${HOME}/.cshrc
```

If you are using an MPI process with a daemon startup, define FILENV by passing environment variables through a daemon. This is often MPI implementation-dependent.

For example, LAM/MPI supports defining FILENV through the `-x` option on `mpirun`, so your commands would look something like this (in bash):

```
% export FILENV=.assign
% mpirun -np 2 -x FILENV foo
```

This would launch the *foo* program on 2 processes, each aware of the FILENV variable.

## ENVIRONMENT

### FILENV

The location of the assign file.

## COPYRIGHT

Copyright (C) 2007 PathScale, LLC. All Rights Reserved.

Copyright (C) 2006, 2007 QLogic Corp. All Rights Reserved.

Copyright (C) 2004, 2005 PathScale, Inc. All Rights Reserved.

## SEE ALSO

**pathf95(1)**

PathScale Compiler Suite User Guide

Online documentation available at <http://www.pathscale.com/docs.html>