Stream Files Utility

Overview

Sometimes one wants to send a large block of data to a device on a SpaceWire link, in order to test throughput and resilience to large traffic loads.

In software releases post 34.0 there is the **stream_files** utility which can be used to send block data.

If you don't have a copy of the software please visit https://www.4links.co.uk/ and download it. The release will contain instructions on how to use it.

The command has a simple usage:-

stream_files dsi speed linkno filelist [targetdirectory]

Where:-

dsi is the IP address or DNS name of the DSI

speed speed is the speed you wish the link to operate at in Mbs

linkno is the link you wish to send the data over

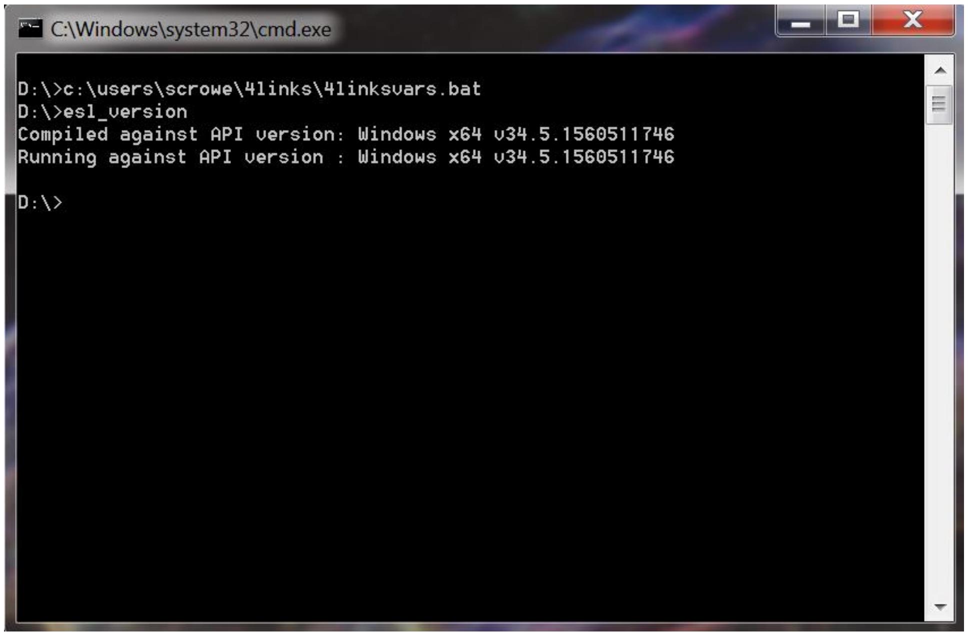
filelist is a file containing the list of files to send. Each file is sent as a single spacewire packet.

Targetdirectory is an optional parameter specifying where received data is stored, if it is ommited then data returned is discarded

Windows Example

Remember to setup the 4links environment by running the environment set up script

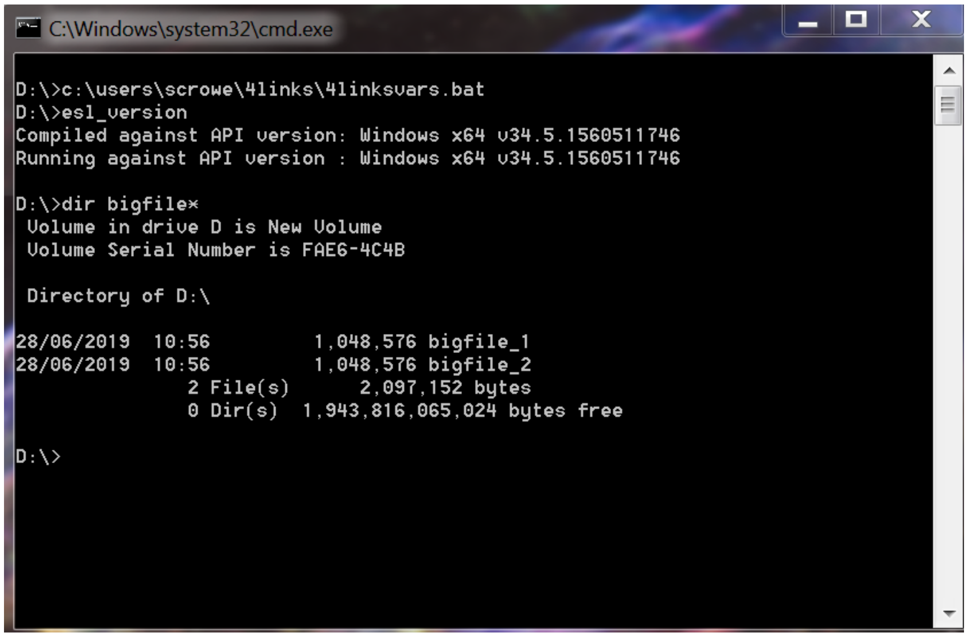
Here is a screenshot of the environment being set up



```
D:\>c:\users\scrowe\4links\4linksvars.bat
D:\>esl_version
Compiled against API version: Windows x64 v34.5.1560511746
Running against API version : Windows x64 v34.5.1560511746

D:\>
```

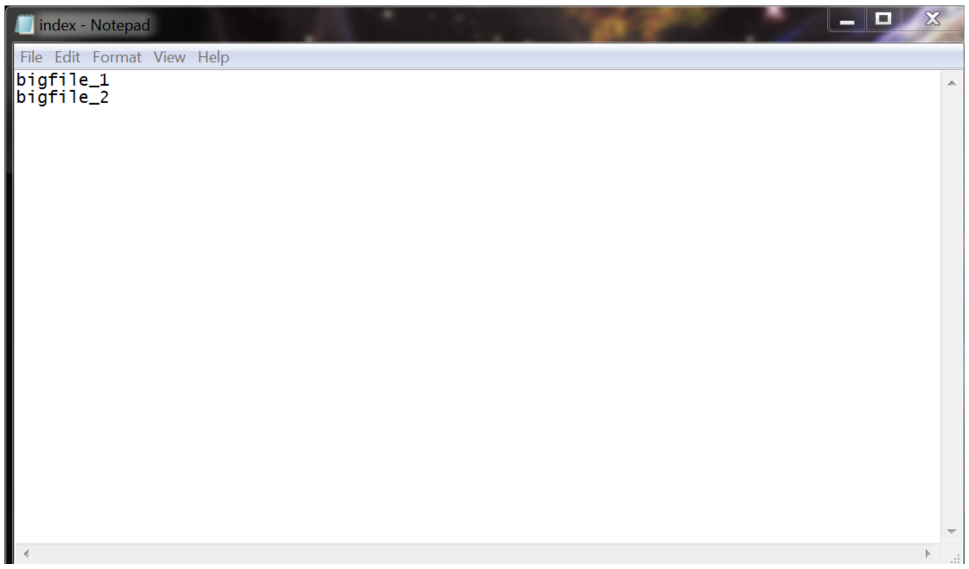
I run the command esl_version to ensure that the environment is set up correctly.

We have two files to send bigfile_1 and bigfile_2



```
D:\>c:\users\scrowe\4links\4linksvars.bat
D:\>esl_version
Compiled against API version: Windows x64 v34.5.1560511746
Running against API version : Windows x64 v34.5.1560511746

D:\>dir bigfile*
 Volume in drive D is New Volume
 Volume Serial Number is FAE6-4C4B

 Directory of D:\

28/06/2019  10:56         1,048,576 bigfile_1
28/06/2019  10:56         1,048,576 bigfile_2
               2 File(s)      2,097,152 bytes
               0 Dir(s)  1,943,816,065,024 bytes free

D:\>
```

We now need to create the indexfile, use notepad to create a file called **index** and put the names bigfile_1 and bigfile_2 in it as shown below and then save the file



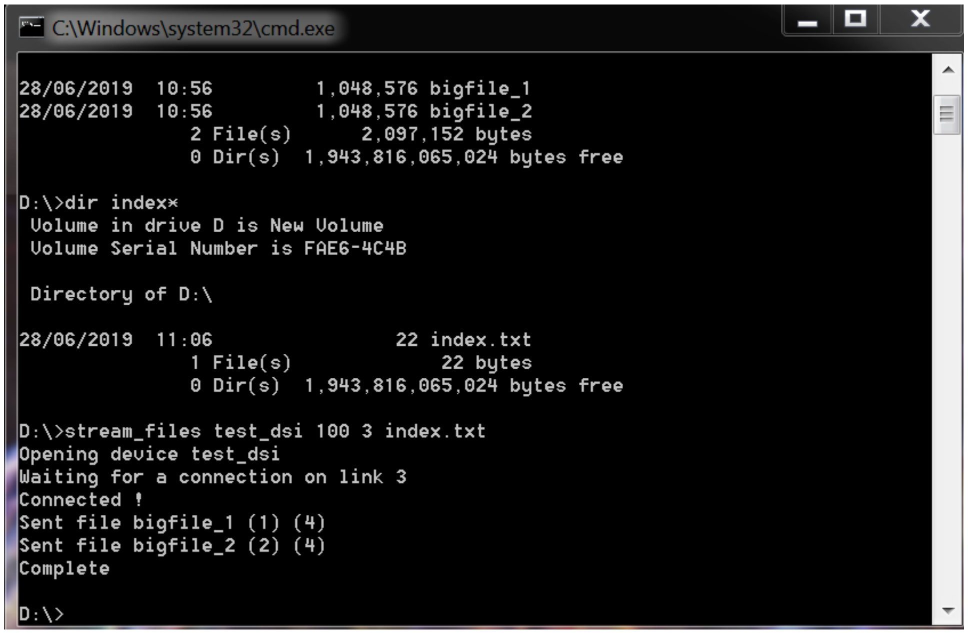
```
bigfile_1
bigfile_2
```

We can now stream the files, network has a DSI set up in DNS called **test_dsi**. It has links 3 and 4 connected to one another, so to start streaming

stream_files test_dsi 100 3 index.txt

It will begin streaming the files listed in index.txt through link 3 of test_dsi at 100Mbs.

Here is the run being performed



C:\Windows\system32\cmd.exe

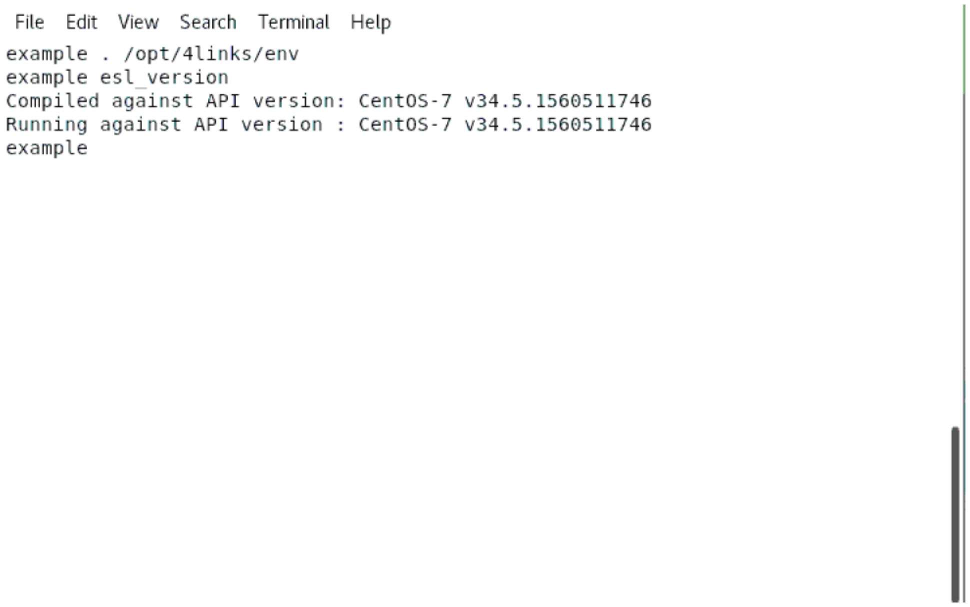
```
28/06/2019  10:56         1,048,576 bigfile_1
28/06/2019  10:56         1,048,576 bigfile_2
               2 File(s)      2,097,152 bytes
               0 Dir(s)  1,943,816,065,024 bytes free

D:\>dir index*
 Volume in drive D is New Volume
 Volume Serial Number is FAE6-4C4B

 Directory of D:\

28/06/2019  11:06                22 index.txt
               1 File(s)             22 bytes
               0 Dir(s)  1,943,816,065,024 bytes free

D:\>stream_files test_dsi 100 3 index.txt
Opening device test_dsi
Waiting for a connection on link 3
Connected !
Sent file bigfile_1 (1) (4)
Sent file bigfile_2 (2) (4)
Complete

D:\>
```

The output contains the file that has been sent and index of it in the stream, the final number is a diagnostic indicating how many files are open by the program

Linux Example

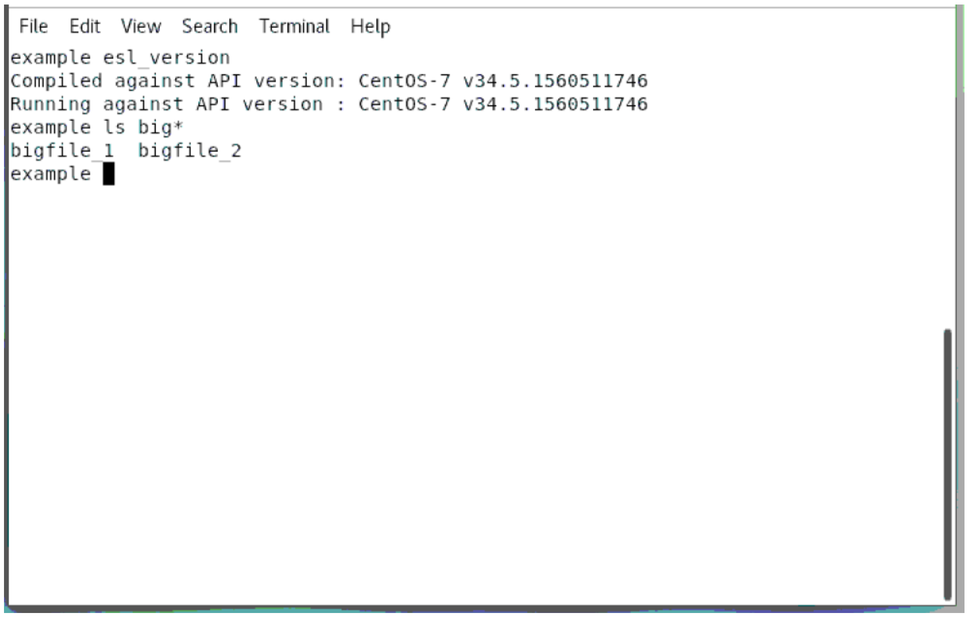
Remember to setup the 4links environment by running the environment set up script

Here is a screenshot of the environment being set up



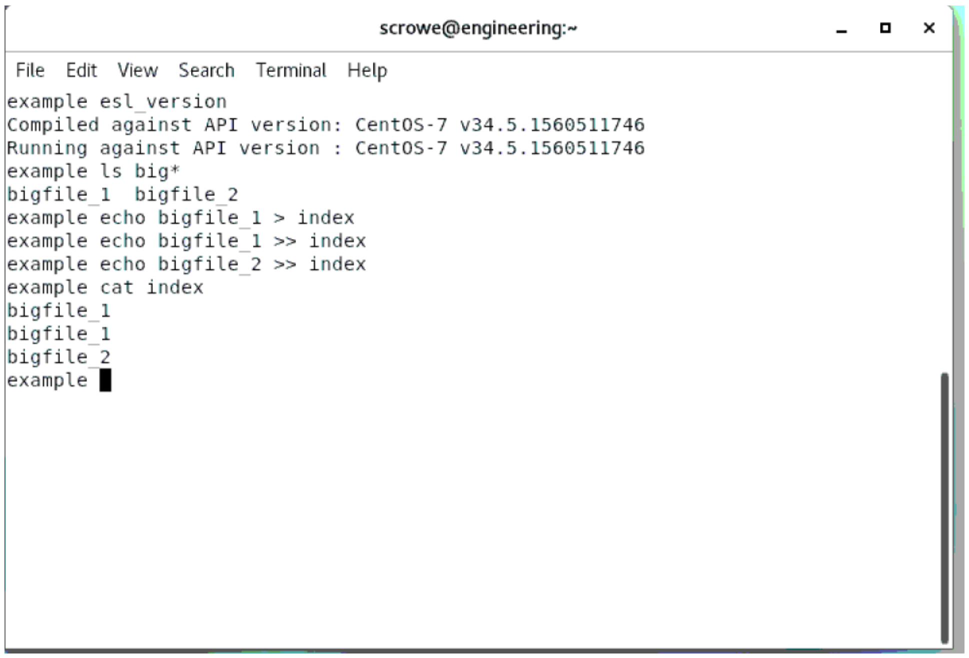
```
example . /opt/4links/env
example esl_version
Compiled against API version: CentOS-7 v34.5.1560511746
Running against API version : CentOS-7 v34.5.1560511746
example
```

We have two files to send **bigfile_1** and **bigfile_2**



```
example esl_version
Compiled against API version: CentOS-7 v34.5.1560511746
Running against API version : CentOS-7 v34.5.1560511746
example ls big*
bigfile_1  bigfile_2
example
```

We need to create the index file, this can be done in vim or any other text editor.

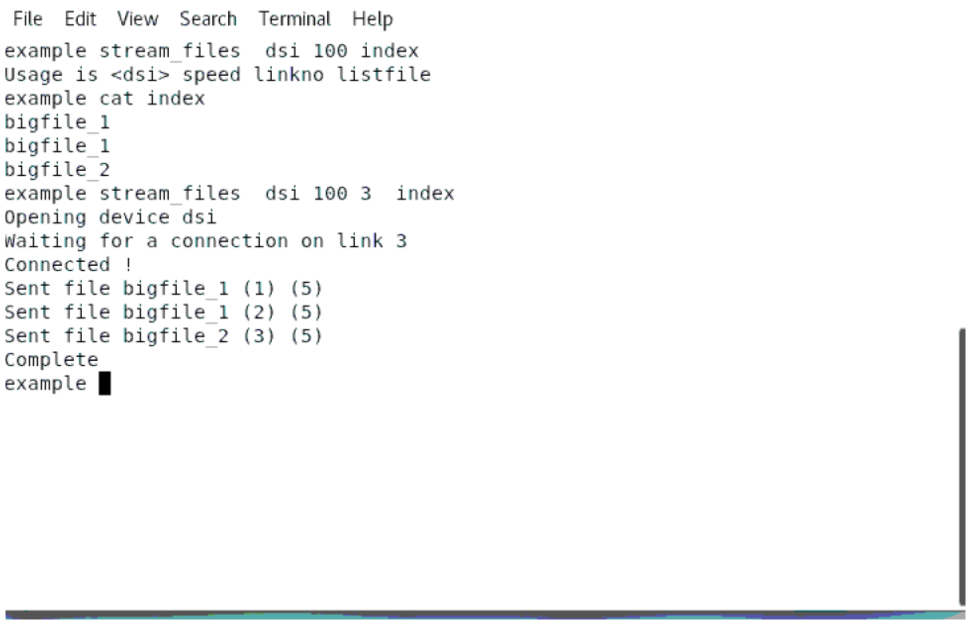


```
example esl_version
Compiled against API version: CentOS-7 v34.5.1560511746
Running against API version : CentOS-7 v34.5.1560511746
example ls big*
bigfile_1  bigfile_2
example echo bigfile_1 > index
example echo bigfile_1 >> index
example echo bigfile_2 >> index
example cat index
bigfile_1
bigfile_1
bigfile_2
example 
```

We can now stream the files, network has a DSI set up in DNS called **dsi**. It has links 3 and 4 connected to one another, so to start streaming

stream_files dsi 100 3 index

Here is the run output



```
example stream_files  dsi 100 index
Usage is <dsi> speed linkno listfile
example cat index
bigfile_1
bigfile_1
bigfile_2
example stream_files  dsi 100 3  index
Opening device dsi
Waiting for a connection on link 3
Connected !
Sent file bigfile_1 (1) (5)
Sent file bigfile_1 (2) (5)
Sent file bigfile_2 (3) (5)
Complete
example █
```

The output contains the file that has been sent and index of it in the stream, the final number is a diagnostic indicating how many files are open by the program.

Target Directory

The optional parameter **targetdirectory** specifies where data received is written to, it contains a file per packet, with the filename having the following format<

<rxlink>_<packetno>_<epoch>

Where

rxlink is the link that the packet has been received on

packetno is the packet number receieved on that link

epoch is the epoch time that the packet started being received on