ION for Windows
Alan Hylton, NASA Glenn Research Center

1 Introduction

The Windows port of ION is compiled with GCC in MinGW. MinGW – Minimalist GNU for Windows – lives up to its name and allows for the minimal number of changes to be made to compile ION Windows executables.

2 MinGW

To get MinGW go to the MinGW homepage (www.**mingw**.org*)* and click on downloads. As of this writing you should find the "Automated MinGW Installer" which is the preferred way of obtaining MinGW.

2.1 Install

Until you get to the screen asking you to select components, it is safe to assume the default settings. When you select components, add the MSYS Basic System. MSYS includes build tools, debugging tools (GDB), and a shell including BASH.

2.2 Environment

The shell has a Linux-like directory structure, but it is not always clear how it works from the Windows side. Fortunately, this is not an issue for two reasons. One is that there is a script to set up the environment, and the other is that the drives and folders on the computer are accessible from within the MinGW shell. The script will be explained in further detail below.

3 ION

ION needs to be unpacked and configured to compile under MinGW. When compiled, the ION executables will be installed in */opt/bin*.

3.1 Unpacking

Go to My Computer, and open the C drive. Create a folder called DTN and put the ION zip file in it. It will be assumed that the name of the file is *ion.tar.gz.* To navigate to this directory in the MinGW shell, enter

cd /c/DTN

To unpack the file, enter the command

tar xfz ion.tar.gz

This should create folder name *ion-open-source*. To get to this directory, type

cd ion-open-source

3.2 Configuration

There is a script called *mingw-setup*. This script will configure the MinGW environment and it will also make sure that ION compiles for the MinGW platform. In the *ion-open-source* directory, type

./mingw-setup

3.3 Build

To make ION, simply run

make

This will compile and link ION putting the executables, libraries, headers, and man files in their respective directories in */opt*. The *mingw-setup* script, among other things, puts */opt/bin* in the path. This means that ION is accessible from within the shell.

4 Testing

ION comes with several tests, located in *ion-open-source/tests*. Some of them are not applicable in MinGW since not all programs have Windows equivalents (e.g. bpchat). Moreover some tests require *netcat*. While there is a MinGW build of *netcat* available, these tests should be ignored. Several things to pay attention to are that some tests refer to configuration files in another directory. Each test has an explanation found in the *dotest* script. This write-up will walk through two tests.

4.1 ltp-retransmission

This test is in *ion-open-source/tests/ltp-retransmission*. Nodes 2 and 3 are started. The tool *bpdriver* will send 100 bundles from ipn:2.1 to ipn:3.1. Each bundle is 64000 bytes. The test is run over ltp and allows ten minutes for the 100 bundles to transfer. This test should run successfully by running

./dotest

If errors are encountered it may be due to loss. To overcome this problem the bit rate may be limited. Open *ltp-retransmission/2.ipn.ltp/amroc.ltprc* and change:

```
a span 3 20 1000000 200 100100 1200 100100 1 'udplso localhost:3113'
```

to

```
a span 3 20 1000000 200 100100 1200 100100 1 'udplso localhost:3113
100000'
```

where the 100000 is the rate limit in bits per second.

4.2 limbo

This test is in *ion-open-source/tests/*limbo. The limbo test sends bundles over TCP and then breaks the TCP connection. When the connection is restored, ION continues sending bundles. Since there is only one route listed when the link goes down the bundles cannot be rerouted. They have to wait for the link to come back up. Run the test with

./dotest

4.3 Others

Some tests take a little bit more work. Some tests have configuration file in *ion-open-source/configs*. For these tests to work the folders *ion-open-source* and *ion-open-source/configs* must be in the path. To do this, write in the shell:

export PATH=$PATH:/c/DTN/ion-open-source:/c/DTN/ion-open-source/configs

If the test fails due to the SDR not being defined, it is because of differences in *awk* between Linux and Windows. The tool *ionadmin* starts the SDR. If the configuration file lists a file name, then that file is expected to have the directives needed to define and start the SDR. If an empty string in quotes is in a directive, then ION used default parameters. The empty string should be in single quotes for Windows but may have been written in double quotes. Thus, in any *.ionrc* file, change:

```
1 1 ""
# start ion node
s
```

To:

```
1 1 ''
# start ion node
s
```

5 Notes

5.1 MinGW

MinGW does no emulation and respects the Windows structure. This means that certain Linux ideas and constructs simply do not translate over. For example, creating a symbolic link with *ln –s file linkname* will just make a copy of *file*.

MinGW's environment and the Windows environment are separate. If you add a directory to the path in one it will not affect the other.

The MinGW shell does not update applications upon resize. This means that if *vim* is open and the window is resized that *vim* will need to be closed and reopened to make it look right.

MinGW does not currently support man pages. This may change in the future, so the man pages are included in the build.

5.2 ION

ION uses shared memory to enable interprocess communication between its components. This is achieved in a different way in Windows than in Linux, but the difference should be transparent to the user. The Windows adaptation is a utility called *winion.exe*. The background process *winion* manages the shared memory objects. Upon an unclean termination, winion.exe or other programs may still be resident. The killm utility – similar to its Linux counterpart – will terminated any lingering ION processes and clean up the pieces.

5.3 Setup Script

The setup script accomplishes several feats. The breakdown of the script follows is in the following format –

`Line(s) from script`
Explanation

`#!/bin/bash`
The script is run in bash.

`makefiles=(ici ltp dgr bp cfdp ams)`
ION is a collection of programs that work together. `makefiles` is a list of directories each of which contain a set of these programs. Each of these directories has a *Makefile* that is edited to support the MinGW platform.

`ismingw=`uname | awk '{print match($0, "MINGW")}'``
This script is only useful under MinGW. This line detects if the user is running MinGW (1) or not (0).

```
if [ $ismingw -eq 1 ]; then
        echo "Configuring ION for compilation under MINGW..."
else
        echo "This setup is only for use under MINGW!"
        exit
fi
```
If the user is running MinGW (the variable ismingw = 1), the script proceeds. Otherwise it prints an error and the script exits.

`cp /bin/make.exe /bin/gmake.exe`
The *Makefiles* reference gmake. In MinGW this is a copy of make.

```
mkdir /opt/bin
mkdir /opt/lib
mkdir /opt/include
mkdir /opt/man
mkdir /opt/man/man1
mkdir /opt/man/man3
mkdir /opt/man/man5
```
MinGW does not create these directories automatically, and without them the compilation of ION fails.

```
for f in ${makefiles[@]}; do
        fname=`echo "$f/Makefile"`
        fnameold=`echo "$fname.old"`

        mv `echo $fname $fnameold`

        awk '
        {
                if($1=="PLATFORMS")
                        $3="i86-mingw"
                print $0
        }
```

```
        ' $fnameold > $fname

        rm $fnameold
done
```
The code above is a loop. For each item in the list `(ici ltp dgr bp cfdp ams)`the code sets the variable *fname* to e.g. *ici/Makefile*. It then makes the corresponding old filename, *fnameold* which in this example would be *ici/Makefile.old*. The original *Makefile* is renamed with the *.old* extension. Then the platform is changed to *i86-mingw* so that the component will compile correctly. This corrected *Makefile* is given the name in *fname* which in this case would be *ici/Makefile*. The old file is removed.

```
rm ~/.profile
```
The file *.profile* in the home directory sets up the bash environment. If there is an old profile, it is removed.

```
echo -e "export PATH=/opt/bin:/opt/lib:/c/DTN/test:\$PATH\nalias
ll='ls -l'\nalias vi='vim'" > ~/.profile
```
The new profile is created. Afterwards *vim* may be run by running *vi*, *ll* is defined, and the path is setup. The above is one line rather than two.

```
. ~/.profile
```
The new profile is used. This prevents the user from having to close the shell after running setup and reopening it.