Quick Start Guide

For

DejaGNU

Prepared By
Kalycito Infotech Pvt Ltd.,
Coimbatore

Table of Contents

1 Introduction

1.1 Purpose

This document is focused on assisting users of DejaGNU to learn and configure dejaGNU faster. This document explains the usage of DejaGNU without any extra packages [automake, autoconf, etc]

1.2 Document Conventions

- The letters in `Courier New` font are code snippets.

1.3 Intended Audience and Reading Suggestions

This document is intended for novice users of DejaGNU. This document is a guideline for using DejaGNU.

1.4 Document Scope

This document will not explain all the features of DejaGNU. This documentation is a guideline for quick starting DejaGNU.

1.5 References

- DejaGNU document by Mr. Rob Savoye

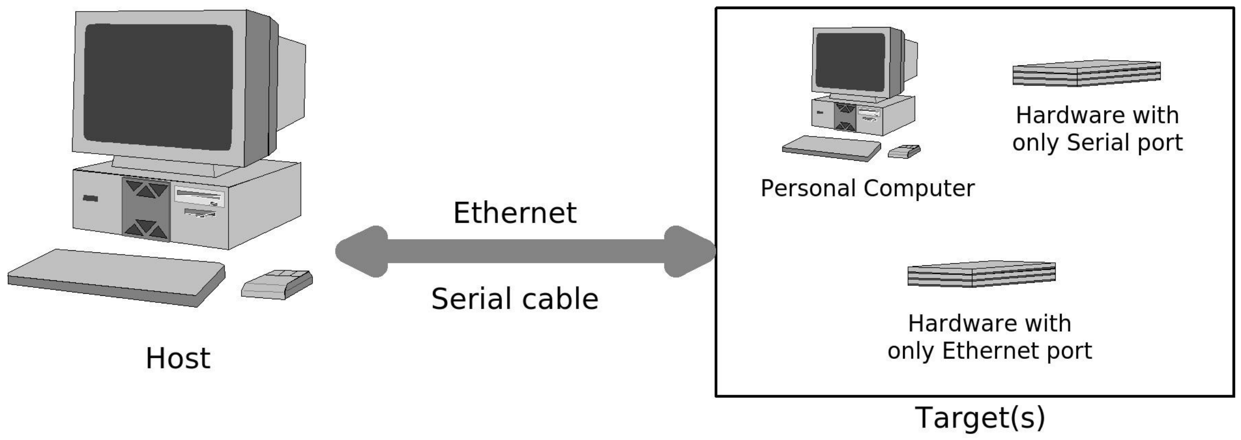
1.6 Disclaimer

The contents of this report are based on information generally available to the public from sources believed to be reliable. No representation is made that it is timely, accurate or complete.

Kalycito has taken due care and caution in compilation of data as this has been obtained from various sources including which it considers reliable and first hand. However, Kalycito does not guarantee the accuracy, adequacy or completeness of any information and it is not responsible for any errors or omissions or for the results obtained from the use of such information and especially states that it has no financial liability whatsoever to the subscribers / users of this report. The information herein, together with all estimates and forecasts, can change without notice.

This report does not purport to be a solicitation of any advice pertaining to trade and business and readers are advised to consult experts or study / evaluate individual business prospectus and other relevant legal documents before taking any decisions based on information provided in the site.
Neither Kalycito nor its Directors or Analysts or Employees accept any liability whatsoever nor do they accept responsibility for any financial consequences arising from the use of the research or information provided herein.

2 Overall Description


Host
Ethernet
Serial cable
Personal Computer
Hardware with only Serial port
Hardware with only Ethernet port
Target(s)

Software on Host

1. Linux
2. DejaGNU
3. Compiler [Cross] for Target
4. ssh/telnet client or Kermit - for communication

Software on Target

1. Boot monitor/ Unix/ Windows/ MacOS/ VxWorks/ any other varients of these
2. ssh/telnet server or Kermit - for communication

DejaGNU is a framework of the GNU project for testing other programs. Its purpose is to provide a single front end for all tests. DejaGNU is written in Expect, which in turn uses Tcl.

DejaGNU offers several advantages for testing:

1. The flexibility and consistency of the DejaGNU framework make it easy to write tests for any program.

2. DejaGNU provides a layer of abstraction which allows you to write tests that are portable to any host or target where a program must be tested. Currently DejaGNU runs tests on several single board computers, whose operating software ranges from just a boot monitor to a full- fledged, Unix-like real time OS.

3. All tests have the same output format. This makes it easy to integrate testing into other software development processes.

   DejaGNU has a very strong history in testing due to its Tcl base.

2.1 How it works

DejaGNU works this way:

- It looks into all the sub-folders of the `pwd` [present working directory]
- Finds an expect script
- Executes the expect script
- The control of executing the test case(s) lies inside the expect script.

2.2 Getting Started

Use your package manager to install DejaGNU or get the tar image and install it.

Create an empty directory, change directory into the new directory and run 'runtest'
```
$ mkdir test
$ cd test
$ runtest
```

You will be able to see traces on your terminal similar to the one below:

```
WARNING: No tool specified
Test Run By selva on Tue Jun 17 14:24:24 2008
Native configuration is i686-pc-linux-gnu

                ===  tests ===

Schedule of variations:
    unix

Running target unix
Using /usr/share/dejagnu/baseboards/unix.exp as board description file for target.
Using /usr/share/dejagnu/config/unix.exp as generic interface file for target.
WARNING: Couldn't find tool config file for unix, using default.

                ===  Summary ===
```

The test directory will have two files generated by DejaGNU, viz., `testrun.sum` and `testrun.log.`

Let's look into these files at later stage of this document. The above traces and files on your computer confirms that DejaGNU has been installed properly on your computer. Now it's time to see something working.

2.3 Testing on local machine

Testing on local machine is nothing but compiling the tests and running the tests on the same machine. The test cases will be executed on the same machine and the results [Pass, Fail, etc] will be returned by DejaGNU depending upon the return value of the test case(s). Many of us may known, when installing GCC, we do a 'make check' to see if GCC has been installed properly. This check is done by DejaGNU.

Let's look into a simple 'Hello world' test on the local machine, where the test case will be written in C language and will be controlled by expect scripts.

Please follow the steps closely till you understand the working of DejaGNU.

```
~$ mkdir local_test
~$ cd local_test
~/local_host$ mkdir srcdir
~/local_host/srcdir$ cd srcdir
~/local_host/srcdir$ touch hello.c
```

The following C code shall go into 'hello.c'

```
#include <stdio.h>
int main()
{
       printf("Hello world\n");
}
```

```
~/local_host/srcdir$ cc hello.c -o hello.o
~/local_host/srcdir$ touch hello.exp
```

The following expect script shall go into hello.exp

```
set test "Local Hello World"
spawn ./srcdir/hello.o
expect {
   -re "Hello world" { pass "$test" }
        default { fail "$test" }
}

~/local_host/srcdir$ cd ..
~/local_host$ runtest
```

You will get the traces similar to the one below.

```
WARNING: No tool specified
Test Run By selva on Wed Jun 18 10:47:34 2008
Native configuration is i686-pc-linux-gnu

                ===  tests ===

Schedule of variations:
    unix

Running target unix
Using /usr/share/dejagnu/baseboards/unix.exp as board description file for target.
Using /usr/share/dejagnu/config/unix.exp as generic interface file for target.
WARNING: Couldn't find tool config file for unix, using default.
Running ./srcdir/hello.exp ...

                ===  Summary ===

# of expected passes            1
```

2.4 Testing on Remote machine

Now we can run the same hello world sample on a remote machine. There are only minor modifications to be done. This document will help you do the remote test with ssh. The other communications methods supported by DejaGNU are pretty understandable if we understand ssh.

We need to add the following addition with the above 'test on local machine' to make it run on remote machine.
Assumption: The remote machine is similar to that of the host machine and runs a Linux operating system. This assumption is made to avoid cross compilation.

Softwares needed
on Host: ssh client, DejaGNU.
on Target: ssh server.
Ssh communication is secure and asks for password authenticated login. For our application we need password less authentication mechanism. There are many documents on the internet that helps you to achieve password less authentication via ssh.

I assume that you have achieved password less authentication via ssh. To test it,

```
~$ ssh <username>@<IP_address>
```

you should be able login without asking for authentication. Now we will be able to run of tests on remote machine. Create an empty directory

```
~$ mkdir remote_test
~$ cd remote_test
~/remote_test$ cp -r ~/local_host/srcdir .
```

```
Now we need to add two new files and modify an existing .exp file.
```

```
~/remote_test$ touch site.exp
```

The following shall go into site.exp

```
lappend boards_dir "."
```

```
~/remote_test$ touch myboard.exp
```

The following shall go into myboard.exp

```
set_board_info hostname <username>@<IP_Address>
```

Now we have to modify `~/remote_host/srcdir/hello.exp`

```
set test "Local Hello World"
 set result [remote_load target ./srcdir/hello.o]

if {[lindex $result 1] == "Hello world"} {
                                            pass "$test"
                                         } else {
                                            fail "$test"
                                           }
```

Now we are ready to run the hello world test on a remote machine.

```
~/remote_test$ runtest --target_board=myboard
```

You will be able to see traces similar to the one below:

```
WARNING: No tool specified
Test Run By selva on Wed Jun 18 16:14:50 2008
Native configuration is i686-pc-linux-gnu

                === tests ===

Schedule of variations:
    myboard

Running target myboard
Using ./myboard.exp as board description file for target.
Running ./srcdir/hello.exp ...

                === Summary ===

# of expected passes            1
```

The above are simplest examples for depicting the usage of DejaGNU. The above samples can be scaled up to run the tests for any number of times. Also the same hello.exp can be modified for running any number of test cases. Refer to a good Expect scripting book. One of the widely referred books is “Exploring Expect” - O'Reilly publication.

2.5 Tips

- The following tips can be used to debug DejaGNU and/or your expect script:
- Place `send_user` command in your expect scripts to print [on screen] the data.
- Increase the verbosity of DejaGNU when executing by giving more number of -v as CLI

```
runtest -v -v -v -v –target_board=myboard
```

- To see the internal traces of expect, use –strace N. 'N' specifies the depth needed.

```
runtest --strace 1 –target_board=myboard
```

3 About Kalycito

Kalycito is a technology services company specializing in embedded systems research and development. The company has a consultation office in France and a R&D office in India. Customers include global OEMs and first-tier manufacturers that draw on Kalycito’s expertise in embedded hardware & software, as well as on the company’s R&D off shoring competence, to reduce their time-to-market and expenses. Besides providing in-depth know-how for real-time operating systems, open source software, networking protocols, and Linux, Kalycito has extensive experience in helping users implement the POWERLINK protocol in their products. With services ranging from consultation, specification and design to testing and maintenance, the company is capable of assisting customers throughout all phases of product development.

Website: http://www.kalycito.com
General Information : info.europe@kalycito.com
Sales Contact : sales.europe@kalycito.com