VE Autotools User Guide

Revision 1.0

Revision History

Rev.	Date	Updates / Remarks
1.0	Sep-2021	The first revision This revision covers VEOS v2.9.1 or later.

Table of Contents

1. Introduction

Autotools is a quick and easy way to manage and package source code so users can compile and install software. Autotools is a collection of related packages which, when used together, remove much of the difficulty involved in creating portable software. These tools, together with a few relatively simple upstream-supplied input files, are used to create the build system for a package. Autotools consists of utility tools of autoconf, automake, and libtool packages.

Autotools are ported for VE to build the software packages for VE architecture

There are two major use cases for SX-Aurora TSUBASA.
1. Porting an existing program for VE using autotools
2. Building a new program for VE using autotools

For both the use cases, VE autotools are required. Users on VH Host OS may want to generate the configure script and makefiles for VE architecture. Hence, user can use VE autotools to generate Makefile and other configuration files for VE architecture.

This documents explains the usage of VE autotools for the above two requirements.
This user guide is intended to be used with VE autotools, v2.9.

2. Installation

2.1 Installing VE Autotools Packages

Install automake-ve, autoconf-ve, libtool-ve

```
# yum install automake-ve autoconf-ve libtool-ve
```

3. Use Cases

This section explains the use cases for porting a project/program for VE architecture. The key requirements are:
1. Support for porting of existing (project meant for x86) and building new project for VE
2. Support of both self-build and cross-build options to build a project for VE
3. Simplify the build mechanism where-in project should be built for VE with/without explicitly setting the architecture for VE.

The (2) and (3) points are briefly explained here before we move into the use cases steps.

Self-build and cross-build:

Both self-build and cross-build are supported for building a VE project.

In self-build, user does not provide host option with configure script.
For example:

```
$./configure
```

In cross build, user provides host option with configure script.
For example:

```
$./configure --host=ve-nec-linux
```

Setting architecture for VE:

One of the requirement is to build for VE even if the user do not explicitly set the architecture as VE.

With the default architecture, the configure script determines the build_triplet and host_triplet for host machine i.e. “x86_64-unknown-linux-gnu”, which is displayed while executing the configure script. However, configure would generate the Makefile for VE.

For example, output of configure script:

```
...
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
...
```

When setting the architecture for VE, the configure script determines the build_triplet as “x86_64-unknown-linux-gnu” and host_triplet as “ve-nec-linux-gnu”, which is displayed while executing the configure script. Here the configure would generate Makefile for VE (as expected). The steps to be followed for setting the VE architecture is explained in the use cases.

For example, output of configure script:

```
...
checking build system type... x86_64-unknown-linux-gnu
checking host system type... ve-nec-linux-gnu
...
```

3.1 Porting an Existing Program for VE

To port an existing program for VE, follow the below mentioned steps:

3.1.1 Self-Build

3.1.1.1 Default Architecture

Configuring project for VE without giving host option and architecture will be set for VE as “x86-unknown-linux-gnu”.

3.1.1.1.1 Non MPI Project

```
1. Move to the project directory
   $ cd <project_directory_path>

2. In case libtool is used in project then replace x86_64 specific
   'ltmain.sh' with VE specific 'ltmain.sh' file in project directory
   i.e. $ cp -f <path of VE specific ltmain.sh> .
   Path of 'ltmain.sh':
   RHEL7:/opt/nec/ve/share/libtool/config/ltmain.sh
   RHEL8:/opt/nec/ve/share/libtool/build-aux/ltmain.sh

   Note: There might be version mismatch error after replacing the VE
   'ltmain.sh' file. Hence, you can try without replacing ltmain.sh file.

3. Execute configure to generate Makefile
   $ ./configure CC=/opt/nec/ve/bin/ncc CXX=/opt/nec/ve/bin/nc++
   FC=/opt/nec/ve/bin/nfort AR=/opt/nec/ve/bin/nar
```

3.1.1.1.2 MPI Project

```
1.  Move to the project directory
    $ cd <project_directory_path>
```

```
2. To set up the MPI compilation environment, execute the following
   script (This execution step is only applicable for MPI programs)

   $ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
   Where{version} is the directory name corresponding to the version of
   NEC MPI you use.

3. In case libtool is used in project then replace x86_64 specific
   'ltmain.sh' with VE specific 'ltmain.sh' file in project directory
   i.e. $ cp -f <path of VE specific ltmain.sh> .
   Path of 'ltmain.sh':
   RHEL7:/opt/nec/ve/share/libtool/config/ltmain.sh
   RHEL8:/opt/nec/ve/share/libtool/build-aux/ltmain.sh

   Note: There might be version mismatch error after replacing the VE
   'ltmain.sh' file. Hence, you can try without replacing ltmain.sh file.

4. Execute configure to generate Makefile
   $ ./configure CC=mpicc CXX=mpic++ FC=mpifort AR=/opt/nec/ve/bin/nar
```

3.1.2 Cross Build

3.1.2.1 Setting Architecture for VE

Configuring project for VE with giving host option to set architecture for VE as “ve-nec-linux-gnu”.

3.1.2.1.1 Non-MPI Project

```
1. Move to the project directory
   $ cd <project_directory_path>

2. Replace x86_64 specific config.guess, config.sub files with VE
   specific files in project directory
   $ cp -f /opt/nec/ve/share/automake-<version>/config.sub .
   $ cp -f /opt/nec/ve/share/automake-<version>/config.guess .

3. In case libtool is used in project then replace x86_64 specific
   'ltmain.sh' with VE specific 'ltmain.sh' file in project directory
   i.e. $ cp -f <path of VE specific ltmain.sh> .
   Path of 'ltmain.sh':
   RHEL7:/opt/nec/ve/share/libtool/config/ltmain.sh
   RHEL8:/opt/nec/ve/share/libtool/build-aux/ltmain.sh

   Note: There might be version mismatch error after replacing the VE
   'ltmain.sh' file. Hence, you can try without replacing ltmain.sh file.

4. Execute configure to generate Makefile
   $ ./configure --host=ve-nec-linux CC=/opt/nec/ve/bin/ncc
   CXX=/opt/nec/ve/bin/nc++ FC=/opt/nec/ve/bin/nfort
   AR=/opt/nec/ve/bin/nar
```

3.1.2.1.2 MPI Project

```
1. Move to the project directory
   $ cd <project_directory_path>
```

```
2. To set up the MPI compilation environment, execute the following
   script (This execution step is only applicable for MPI programs)

   $ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
   Where{version} is the directory name corresponding to the version of
   NEC MPI you use.

3. Replace x86_64 specific config.guess, config.sub files with VE
   specific files in project directory
   $ cp -f /opt/nec/ve/share/automake-<version>/config.sub .
   $ cp -f /opt/nec/ve/share/automake-<version>/config.guess .

4. In case libtool is used in project then replace x86_64 specific
   'ltmain.sh' with VE specific 'ltmain.sh' file in project directory
   i.e. $ cp -f <path of VE specific ltmain.sh> .
   Path of 'ltmain.sh':
   RHEL7:/opt/nec/ve/share/libtool/config/ltmain.sh
   RHEL8:/opt/nec/ve/share/libtool/build-aux/ltmain.sh

   Note: There might be version mismatch error after replacing the VE
   'ltmain.sh' file. Hence, you can try without replacing ltmain.sh file.

5. Execute configure to generate Makefile
   $ ./configure --host=ve-nec-linux CC=mpicc CXX=mpic++ FC=mpifort
   AR=/opt/nec/ve/bin/nar
```

3.2 Building a New Program for VE

Users need to create Makefile.am, configure.ac and README in addition to source files. Refer the autotools tutorial available at gnu.org.

https://www.gnu.org/software/automake/manual/html_node/Creating-amhello.html

To build a new program for VE, follow the below mentioned steps:

3.2.1 Self-Build

3.2.1.1 Default Architecture

Configuring project for VE without giving host option and architecture will be set for VE as "x86-unknown-linux-gnu".

3.2.1.1.1 Non-MPI Projects

```
1.  Move to the project directory
    $ cd <project_directory_path>

2.  Execute autoreconf
    $ /opt/nec/ve/bin/autoreconf -isf

3.  Execute configure to generate Makefile
    $ ./configure CC=/opt/nec/ve/bin/ncc CXX=/opt/nec/ve/bin/nc++
    FC=/opt/nec/ve/bin/nfort AR=/opt/nec/ve/bin/nar
```

3.2.1.1.2 MPI Project

```
1. Move to the project directory
   $ cd <project_directory_path>
```

```
2. To set up the MPI compilation environment, execute the following
   script (This execution step is only applicable for MPI programs)

   $ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
   Where{version} is the directory name corresponding to the version of
   NEC MPI you use.

3. Execute autoreconf
   $ /opt/nec/ve/bin/autoreconf -isf

4.  Execute configure to generate Makefile
    $ ./configure CC=mpicc CXX=mpic++ FC=mpifort AR=/opt/nec/ve/bin/nar
```

3.2.2 Cross Build

3.2.2.1 Setting Architecture for VE

Configuring project for VE with giving host option to set architecture for VE as “ve-nec-linux-gnu”.

3.2.2.1.1 Non-MPI Project

```
1.  Move to the project directory
    $ cd <project_directory_path>

2.  Execute VE autoreconf
    $ /opt/nec/ve/bin/autoreconf -isf

3.  Execute configure to generate Makefile
    $ ./configure --host=ve-nec-linux
```

3.2.2.1.2 MPI Project

```
1. Move to the project directory
   $ cd <project_directory_path>

2. To set up the MPI compilation environment, execute the following
   script (This execution step is only applicable for MPI programs)

   $ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
   Where{version} is the directory name corresponding to the version of
   NEC MPI you use.

3. Execute autoreconf
   $ /opt/nec/ve/bin/autoreconf -isf

4. Execute configure to generate Makefile
   $ ./configure --host=ve-nec-linux
```

4. Examples

This section highlights the use cases with an example.

The example programs are available on github:
https://github.com/veos-sxarr-NEC/examples/tree/master/autotools

4.1 Example of Porting an Existing Program for VE

4.1.1 Self-Build

4.1.1.1 Default Architecture

4.1.1.1.1 Non-MPI Project

```
1.  Move to the project directory
    $  cd example/existing_project

2.  Execute configure to generate Makefile
    $ ./configure CC=/opt/nec/ve/bin/ncc CXX=/opt/nec/ve/bin/nc++
    FC=/opt/nec/ve/bin/nfort AR=/opt/nec/ve/bin/nar
    checking for a BSD-compatible install... /usr/bin/install -c
    checking whether build environment is sane... yes
    checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
    checking for gawk... gawk
    checking whether make sets $(MAKE)... yes
    checking whether make supports nested variables... yes
    checking build system type... x86_64-unknown-linux-gnu
    checking host system type... x86_64-unknown-linux-gnu
    checking for gcc... /opt/nec/ve/bin/ncc
    checking whether the C compiler works... yes
    checking for C compiler default output file name... a.out
    checking for suffix of executables...
    checking whether we are cross compiling... no
    checking for suffix of object files... o
    checking whether we are using the GNU C compiler... yes
    checking whether /opt/nec/ve/bin/ncc accepts -g... yes
    checking for /opt/nec/ve/bin/ncc option to accept ISO C89... none
    needed
    checking for style of include used by make... GNU
    checking dependency style of /opt/nec/ve/bin/ncc... tru64
    checking whether we are using the GNU C++ compiler... yes
    checking whether /opt/nec/ve/bin/nc++ accepts -g... yes
    checking dependency style of /opt/nec/ve/bin/nc++... tru64
    checking whether we are using the GNU Fortran compiler... no
    checking whether /opt/nec/ve/bin/nfort accepts -g... yes
    checking the archiver (/opt/nec/ve/bin/nar) interface... ar
    checking that generated files are newer than configure... done
    configure: creating ./config.status
    config.status: creating Makefile
    config.status: creating config.h
    config.status: config.h is unchanged
    config.status: executing depfiles commands

3.  Execute make
    $ make
    make  all-am
    make[1]: Entering directory `/home/user1/example/existing_project'
    source='sample_c.c' object='sample_c.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    /opt/nec/ve/bin/ncc -DHAVE_CONFIG_H -I.     -g -O2 -c sample_c.c
    /opt/nec/ve/bin/ncc  -g -O2 -Wl,-z,max-page-size=0x200000  -o sample_c
    sample_c.o
    source='sample_cpp.cpp' object='sample_cpp.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
```

```
    /opt/nec/ve/bin/nc++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o sample_cpp.o
    sample_cpp.cpp
    /opt/nec/ve/bin/nc++  -g -O2 -Wl,-z,max-page-size=0x200000  -o
    sample_cpp sample_cpp.o
    /opt/nec/ve/bin/nfort  -g -c -o sample_fortran.o sample_fortran.f90
    /opt/nec/ve/bin/nfort  -g -Wl,-z,max-page-size=0x200000  -o
    sample_fortran sample_fortran.o
    make[1]: Leaving directory `/home/user1/example/existing_project'

4.  Execute VE binaries (our example project generates 3 binaries for
    three sample programs)
    $ /opt/nec/ve/bin/ve_exec ./sample_c
    Hello from C test program
    $ /opt/nec/ve/bin/ve_exec ./sample_cpp
    Hello from CPP test program
    $ /opt/nec/ve/bin/ve_exec ./sample_fortran
    Hello from fortran test program
```

4.1.1.1.2 MPI Project

```
1.  Move to the project directory
    $  cd example/existingproject_mpi

2.  Setup environment for MPI program
    $ source /opt/nec/ve/mpi/2.7.0/bin/necmpivars.sh

3.  Execute configure to generate Makefile
    $ ./configure CC=mpicc CXX=mpic++ FC=mpifort AR=/opt/nec/ve/bin/nar
    checking for a BSD-compatible install... /usr/bin/install -c
    checking whether build environment is sane... yes
    checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
    checking for gawk... gawk
    checking whether make sets $(MAKE)... yes
    checking whether make supports nested variables... yes
    checking whether the C compiler works... yes
    checking for C compiler default output file name... a.out
    checking for suffix of executables...
    checking whether we are cross compiling... no
    checking for suffix of object files... o
    checking whether we are using the GNU C compiler... yes
    checking whether mpicc accepts -g... yes
    checking for mpicc optiosn to accept ISO C89... none needed
    checking for style of include used by make... GNU
    checking dependency style of mpicc... tru64
    checking whether we are using the GNU C++ compiler... yes
    checking whether mpic++ accepts -g... yes
    checking dependency style of mpic++... tru64
    checking whether we are using the GNU Fortran compiler... no
    checking whether mpifort accepts -g... yes
    checking the archiver (/opt/nec/ve/bin/nar) interface... ar
    checking build system type... x86_64-unknown-linux-gnu
    checking host system type... x86_64-unknown-linux-gnu
    checking that generated files are newer than configure... done
    configure: creating ./config.status
    config.status: creating Makefile
    config.status: creating config.h
    config.status: config.h is unchanged
    config.status: executing depfiles commands
```

```
4.  Execute make
    $ make
    make  all-am
    make[1]: Entering directory `/home/user1/example/existingproject_mpi'
    source='autotools_test_c.c' object='autotools_test_c.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    mpicc -DHAVE_CONFIG_H -I.     -g -O2 -c autotools_test_c.c
    mpicc  -g -O2   -o autotools_test_c autotools_test_c.o
    source='autotools_test_cpp.cpp' object='autotools_test_cpp.o'
    libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    mpic++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o autotools_test_cpp.o
    autotools_test_cpp.cpp
    mpic++  -g -O2   -o autotools_test_cpp autotools_test_cpp.o
    mpifort  -g -c -o autotools_test_fortran.o autotools_test_fortran.f90
    mpifort  -g   -o autotools_test_fortran autotools_test_fortran.o
    make[1]: Leaving directory `/home/user1/example/existingproject_mpi'

5.  Execute VE binaries (our example project generates 3 binaries for
    three sample programs)
    $ mpirun ./autotools_test_c
    Hello World from C program
    Rank ID: 0
    Number of processes: 1
    Host name: localhost
    $ mpirun ./autotools_test_cpp
    Hello World from C++ program
    Rank ID: 0
    Number of processes: 1
    $ mpirun ./autotools_test_fortran
    Hello World from Fortran program
    Rank ID:  0
    Number of processes:  1
```

4.1.2 Cross Build

4.1.2.1 Setting Architecture for VE

4.1.2.1.1 Non-MPI Project

```
1. Move to the project directory
   $ cd example/existing_project

2. Replace config.guess, config.sub files in project directory
   $ cp -f /opt/nec/ve/share/automake-1.13/config.guess .
   $ cp -f /opt/nec/ve/share/automake-1.13/config.sub .

3. Execute configure to generate Makefile
   $ ./configure --host=ve-nec-linux CC=/opt/nec/ve/bin/ncc
   CXX=/opt/nec/ve/bin/nc++ FC=/opt/nec/ve/bin/nfort
   AR=/opt/nec/ve/bin/nar
   checking for a BSD-compatible install... /usr/bin/install -c
   checking whether build environment is sane... yes
   checking for ve-nec-linux-strip... no
   checking for strip... strip
   checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
```

```
   checking for gawk... gawk
   checking whether make sets $(MAKE)... yes
   checking whether make supports nested variables... yes
   checking build system type... x86_64-unknown-linux-gnu
   checking host system type... ve-nec-linux-gnu
   checking for ve-nec-linux-gcc... /opt/nec/ve/bin/ncc
   checking whether the C compiler works... yes
   checking for C compiler default output file name... a.out
   checking for suffix of executables...
   checking whether we are cross compiling... no
   checking for suffix of object files... o
   checking whether we are using the GNU C compiler... yes
   checking whether /opt/nec/ve/bin/ncc accepts -g... yes
   checking for /opt/nec/ve/bin/ncc option to accept ISO C89... none
   needed
   checking for style of include used by make... GNU
   checking dependency style of /opt/nec/ve/bin/ncc... tru64
   checking whether we are using the GNU C++ compiler... yes
   checking whether /opt/nec/ve/bin/nc++ accepts -g... yes
   checking dependency style of /opt/nec/ve/bin/nc++... tru64
   checking for ve-nec-linux-gfortran... /opt/nec/ve/bin/nfort
   checking whether we are using the GNU Fortran compiler... no
   checking whether /opt/nec/ve/bin/nfort accepts -g... yes
   checking for ve-nec-linux-ar... /opt/nec/ve/bin/nar
   checking the archiver (/opt/nec/ve/bin/nar) interface... ar
   checking that generated files are newer than configure... done
   configure: creating ./config.status
   config.status: creating Makefile
   config.status: creating config.h
   config.status: config.h is unchanged
   config.status: executing depfiles commands
```

4. Execute make

```
   $ make
   make  all-am
   make[1]: Entering directory `/home/user1/example/existing_project'
   source='sample_c.c' object='sample_c.o' libtool=no \
   DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
   /opt/nec/ve/bin/ncc -DHAVE_CONFIG_H -I.     -g -O2 -c sample_c.c
   /opt/nec/ve/bin/ncc  -g -O2 -Wl,-z,max-page-size=0x200000  -o sample_c
   sample_c.o
   source='sample_cpp.cpp' object='sample_cpp.o' libtool=no \
   DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
   /opt/nec/ve/bin/nc++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o sample_cpp.o
   sample_cpp.cpp
   /opt/nec/ve/bin/nc++  -g -O2 -Wl,-z,max-page-size=0x200000  -o
   sample_cpp sample_cpp.o
   /opt/nec/ve/bin/nfort  -g -c -o sample_fortran.o sample_fortran.f90
   /opt/nec/ve/bin/nfort  -g -Wl,-z,max-page-size=0x200000  -o
   sample_fortran sample_fortran.o
   make[1]: Leaving directory `/home/user1/example/existing_project'
```

5. Execute VE binaries (our example project generates 3 binaries for three sample programs)

```
   $ /opt/nec/ve/bin/ve_exec ./sample_c
   Hello from C test program
   $ /opt/nec/ve/bin/ve_exec ./sample_cpp
```

```
   Hello from CPP test program
   $ /opt/nec/ve/bin/ve_exec ./sample_fortran
   Hello from fortran test program
```

4.1.2.1.2 MPI Project

```
1. Move to the project directory
   $ cd example/existingproject_mpi

2. Setup environment for MPI program
   $ source /opt/nec/ve/mpi/2.7.0/bin/necmpivars.sh

3. Replace config.guess, config.sub files in project directory
   $ cp -f /opt/nec/ve/share/automake-1.13/config.guess .
   $ cp -f /opt/nec/ve/share/automake-1.13/config.sub .

4. Execute configure to generate Makefile
   $ ./configure --host=ve-nec-linux CC=mpicc CXX=mpic++ FC=mpifort
   AR=/opt/nec/ve/bin/nar
   checking for a BSD-compatible install... /usr/bin/install -c
   checking whether build environment is sane... yes
   checking for ve-nec-linux-strip... no
   checking for strip... strip
   checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
   checking for gawk... gawk
   checking whether make sets $(MAKE)... yes
   checking whether make supports nested variables... yes
   checking for ve-nec-linux-mpicc... mpicc
   checking whether the C compiler works... yes
   checking for C compiler default output file name... a.out
   checking for suffix of executables...
   checking whether we are cross compiling... no
   checking for suffix of object files... o
   checking whether we are using the GNU C compiler... yes
   checking whether mpicc accepts -g... yes
   checking for mpicc option to accept ISO C89... none needed
   checking for style of include used by make... GNU
   checking dependency style of mpicc... tru64
   checking whether we are using the GNU C++ compiler... yes
   checking whether mpic++ accepts -g... yes
   checking dependency style of mpic++... tru64
   checking for ve-nec-linux-mpifort... mpifort
   checking whether we are using the GNU Fortran compiler... no
   checking whether mpifort accepts -g... yes
   checking for ve-nec-linux-ar... /opt/nec/ve/bin/nar
   checking the archiver (/opt/nec/ve/bin/nar) interface... ar
   checking build system type... x86_64-unknown-linux-gnu
   checking host system type... ve-nec-linux-gnu
   checking that generated files are newer than configure... done
   configure: creating ./config.status
   config.status: creating Makefile
   config.status: creating config.h
   config.status: config.h is unchanged
   config.status: executing depfiles commands

5. Execute make
   $ make
```

```
    cd . && /bin/sh /home/user1/example/existingproject_mpi/missing
   automake-1.13 --foreign Makefile
    cd . && /bin/sh ./config.status Makefile depfiles
   config.status: creating Makefile
   config.status: executing depfiles commands
   make  all-am
   make[1]: Entering directory `/home/user1/example/existingproject_mpi'
   source='autotools_test_c.c' object='autotools_test_c.o' libtool=no \
   DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
   mpicc -DHAVE_CONFIG_H -I.     -g -O2 -c autotools_test_c.c
   mpicc  -g -O2   -o autotools_test_c autotools_test_c.o
   source='autotools_test_cpp.cpp' object='autotools_test_cpp.o'
   libtool=no \
   DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
   mpic++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o autotools_test_cpp.o
   autotools_test_cpp.cpp
   mpic++  -g -O2   -o autotools_test_cpp autotools_test_cpp.o
   mpifort  -g -c -o autotools_test_fortran.o autotools_test_fortran.f90
   mpifort  -g   -o autotools_test_fortran autotools_test_fortran.o
   make[1]: Leaving directory `/home/user1/example/existingproject_mpi'

6. Execute VE binaries (our example project generates 3 binaries for
   three sample programs)
   $ mpirun ./autotools_test_c
   Hello World from C program
   Rank ID: 0
   Number of processes: 1
   Host name: localhost
   $ mpirun ./autotools_test_cpp
   Hello World from C++ program
   Rank ID: 0
   Number of processes: 1
   $ mpirun ./autotools_test_fortran
   Hello World from Fortran program
   Rank ID:  0
   Number of processes:  1
```

4.2 Example of Building a New Program for VE

4.2.1 Self-Build

4.2.1.1 Default Architecture

4.2.1.1.1 Non-MPI Project

```
1. Move to the project directory
    $ cd example/new_project

2. Execute autoreconf
    $ /opt/nec/ve/bin/autoreconf -isf
    configure.ac:11: installing './ar-lib'
    configure.ac:7: installing './config.guess'
    configure.ac:7: installing './config.sub'
    configure.ac:4: installing './install-sh'
    configure.ac:4: installing './missing'
    Makefile.am: installing './depcomp'

3. Execute configure to generate Makefile
```

```
    $ ./configure CC=/opt/nec/ve/bin/ncc CXX=/opt/nec/ve/bin/nc++
    FC=/opt/nec/ve/bin/nfort AR=/opt/nec/ve/bin/nar
    checking for a BSD-compatible install... /usr/bin/install -c
    checking whether build environment is sane... yes
    checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
    checking for gawk... gawk
    checking whether make sets $(MAKE)... yes
    checking whether make supports nested variables... yes
    checking build system type... x86_64-unknown-linux-gnu
    checking host system type... x86_64-unknown-linux-gnu
    checking for gcc... /opt/nec/ve/bin/ncc
    checking whether the C compiler works... yes
    checking for C compiler default output file name... a.out
    checking for suffix of executables...
    checking whether we are cross compiling... no
    checking for suffix of object files... o
    checking whether we are using the GNU C compiler... yes
    checking whether /opt/nec/ve/bin/ncc accepts -g... yes
    checking for /opt/nec/ve/bin/ncc option to accept ISO C89... none
    needed
    checking for style of include used by make... GNU
    checking dependency style of /opt/nec/ve/bin/ncc... tru64
    checking whether we are using the GNU C++ compiler... yes
    checking whether /opt/nec/ve/bin/nc++ accepts -g... yes
    checking dependency style of /opt/nec/ve/bin/nc++... tru64
    checking whether we are using the GNU Fortran compiler... no
    checking whether /opt/nec/ve/bin/nfort accepts -g... yes
    checking the archiver (/opt/nec/ve/bin/nar) interface... ar
    checking that generated files are newer than configure... done
    configure: creating ./config.status
    config.status: creating Makefile
    config.status: creating config.h
    config.status: executing depfiles commands

4. Execute make
    $ make
    (CDPATH="${ZSH_VERSION+.}:" && cd . && /bin/sh
    /home/user1/example/new_project/missing autoheader)
    rm -f stamp-h1
    touch config.h.in
    cd . && /bin/sh ./config.status config.h
    config.status: creating config.h
    config.status: config.h is unchanged
    make  all-am
    make[1]: Entering directory `/home/user1/example/new_project'
    source='sample_c.c' object='sample_c.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    /opt/nec/ve/bin/ncc -DHAVE_CONFIG_H -I.     -g -O2 -c sample_c.c
    /opt/nec/ve/bin/ncc  -g -O2 -Wl,-z,max-page-size=0x200000  -o sample_c
    sample_c.o
    source='sample_cpp.cpp' object='sample_cpp.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    /opt/nec/ve/bin/nc++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o sample_cpp.o
    sample_cpp.cpp
    /opt/nec/ve/bin/nc++  -g -O2 -Wl,-z,max-page-size=0x200000  -o
    sample_cpp sample_cpp.o
    /opt/nec/ve/bin/nfort  -g -c -o sample_fortran.o sample_fortran.f90
```

```
    /opt/nec/ve/bin/nfort  -g -Wl,-z,max-page-size=0x200000  -o
    sample_fortran sample_fortran.o
    make[1]: Leaving directory `/home/user1/example/new_project'

5. Execute VE binaries (our example project generates 3 binaries for
   three sample programs)
    $ /opt/nec/ve/bin/ve_exec ./sample_c
    Hello from C test program
    $ /opt/nec/ve/bin/ve_exec ./sample_cpp
    Hello from CPP test program
    $ /opt/nec/ve/bin/ve_exec ./sample_fortran
    Hello from fortran test program
```

4.2.1.1.2 MPI Project

```
1.  Move to the project directory
    $ cd example/newproject_mpi

2.  Setup environment for MPI program
    $ source /opt/nec/ve/mpi/2.7.0/bin/necmpivars.sh

3.  Execute autoreconf
    $ /opt/nec/ve/bin/autoreconf -isf

4.  Execute configure to generate Makefile
    $ ./configure CC=mpicc CXX=mpic++ FC=mpifort AR=/opt/nec/ve/bin/nar
    checking for a BSD-compatible install... /usr/bin/install -c
    checking whether build environment is sane... yes
    checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
    checking for gawk... gawk
    checking whether make sets $(MAKE)... yes
    checking whether make supports nested variables... yes
    checking whether the C compiler works... yes
    checking for C compiler default output file name... a.out
    checking for suffix of executables...
    checking whether we are cross compiling... no
    checking for suffix of object files... o
    checking whether we are using the GNU C compiler... yes
    checking whether mpicc accepts -g... yes
    checking for mpicc option to accept ISO C89... none needed
    checking for style of include used by make... GNU
    checking dependency style of mpicc... tru64
    checking whether we are using the GNU C++ compiler... yes
    checking whether mpic++ accepts -g... yes
    checking dependency style of mpic++... tru64
    checking whether we are using the GNU Fortran compiler... no
    checking whether mpifort accepts -g... yes
    checking the archiver (/opt/nec/ve/bin/nar) interface... ar
    checking build system type... x86_64-unknown-linux-gnu
    checking host system type... x86_64-unknown-linux-gnu
    checking that generated files are newer than configure... done
    configure: creating ./config.status
    config.status: creating Makefile
    config.status: creating config.h
    config.status: executing depfiles commands

5.  Execute make
    $ make
```

```
    make  all-am
    make[1]: Entering directory `/home/user1/example/newproject_mpi'
    source='autotools_test_c.c' object='autotools_test_c.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    mpicc -DHAVE_CONFIG_H -I.     -g -O2 -c autotools_test_c.c
    mpicc  -g -O2   -o autotools_test_c autotools_test_c.o
    source='autotools_test_cpp.cpp' object='autotools_test_cpp.o'
    libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    mpic++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o autotools_test_cpp.o
    autotools_test_cpp.cpp
    mpic++  -g -O2   -o autotools_test_cpp autotools_test_cpp.o
    mpifort  -g -c -o autotools_test_fortran.o autotools_test_fortran.f90
    mpifort  -g   -o autotools_test_fortran autotools_test_fortran.o
    make[1]: Leaving directory `/home/user1/example/newproject_mpi'

6.  Execute VE binaries (our example project generates 3 binaries for
    three sample programs)
    $ mpirun ./autotools_test_c
    Hello World from C program
    Rank ID: 0
    Number of processes: 1
    Host name: localhost
    $ mpirun ./autotools_test_cpp
    Hello World from C++ program
    Rank ID: 0
    Number of processes: 1
    $ mpirun ./autotools_test_fortran
    Hello World from Fortran program
    Rank ID:  0
    Number of processes:  1
```

4.2.2 Cross Build

4.2.2.1 Setting Architecture for VE

4.2.2.1.1 Non-MPI Project

```
1.  Move to the project directory
    $ cd example/new_project

2.  Execute VE autoreconf
    $ /opt/nec/ve/bin/autoreconf -isf
    aclocal: warning: couldn't open directory 'm4': No such file or
    directory
    configure.ac:15: installing './ar-lib'
    configure.ac:10: installing './compile'
    configure.ac:7: installing './config.guess'
    configure.ac:7: installing './config.sub'
    configure.ac:4: installing './install-sh'
    configure.ac:4: installing './missing'
    Makefile.am: installing './depcomp'

3.  Execute configure to generate Makefile
    $./configure --host=ve-nec-linux
    checking for a BSD-compatible install... /usr/bin/install -c
    checking whether build environment is sane... yes
```

```
    checking for ve-nec-linux-strip... no
    checking for strip... strip
    checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
    checking for gawk... gawk
    checking whether make sets $(MAKE)... yes
    checking whether make supports nested variables... yes
    checking build system type... x86_64-unknown-linux-gnu
    checking host system type... ve-nec-linux-gnu
    checking for ncc... /opt/nec/ve/bin/ncc
    checking for ve-nec-linux-gcc... (cached) /opt/nec/ve/bin/ncc
    checking whether the C compiler works... yes
    checking for C compiler default output file name... a.out
    checking for suffix of executables...
    checking whether we are cross compiling... no
    checking for suffix of object files... o
    checking whether we are using the GNU C compiler... yes
    checking whether /opt/nec/ve/bin/ncc accepts -g... yes
    checking for /opt/nec/ve/bin/ncc option to accept ISO C89... none
    needed
    checking for style of include used by make... GNU
    checking dependency style of /opt/nec/ve/bin/ncc... tru64
    checking for nc++... /opt/nec/ve/bin/nc++
    checking whether we are using the GNU C++ compiler... yes
    checking whether /opt/nec/ve/bin/nc++ accepts -g... yes
    checking dependency style of /opt/nec/ve/bin/nc++... tru64
    checking for nfort... /opt/nec/ve/bin/nfort
    checking whether we are using the GNU Fortran compiler... no
    checking whether /opt/nec/ve/bin/nfort accepts -g... yes
    checking for nar... /opt/nec/ve/bin/nar
    checking the archiver (/opt/nec/ve/bin/nar) interface... ar
    checking that generated files are newer than configure... done
    configure: creating ./config.status
    config.status: creating Makefile
    config.status: creating config.h
    config.status: executing depfiles commands

4.  Execute make
    $ make
    make  all-am
    make[1]: Entering directory `/home/user1/example/new_project'
    source='sample_c.c' object='sample_c.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    /opt/nec/ve/bin/ncc -DHAVE_CONFIG_H -I.     -g -O2 -c sample_c.c
    /opt/nec/ve/bin/ncc  -g -O2 -Wl,-z,max-page-size=0x200000  -o sample_c
    sample_c.o
    source='sample_cpp.cpp' object='sample_cpp.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    /opt/nec/ve/bin/nc++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o sample_cpp.o
    sample_cpp.cpp
    /opt/nec/ve/bin/nc++  -g -O2 -Wl,-z,max-page-size=0x200000  -o
    sample_cpp sample_cpp.o
    /opt/nec/ve/bin/nfort  -g -c -o sample_fortran.o sample_fortran.f90
    /opt/nec/ve/bin/nfort  -g -Wl,-z,max-page-size=0x200000  -o
    sample_fortran sample_fortran.o
    make[1]: Leaving directory `/home/user1/example/new_project'
```

```
5.  Execute VE binaries (our example project generates 3 binaries for
    three sample programs)
    $ /opt/nec/ve/bin/ve_exec ./sample_c
    Hello from C test program
    $ /opt/nec/ve/bin/ve_exec ./sample_cpp
    Hello from CPP test program
    $ /opt/nec/ve/bin/ve_exec ./sample_fortran
    Hello from fortran test program
```

4.2.2.1.2 MPI Project

```
1.  Move to the project directory
    $ cd example/newproject_mpi

2.  Setup environment for MPI program
    $ source /opt/nec/ve/mpi/2.7.0/bin/necmpivars.sh

3.  Execute VE autoreconf
    $ /opt/nec/ve/bin/autoreconf -isf
    aclocal: warning: couldn't open directory 'm4': No such file or
    directory
    configure.ac:11: installing './ar-lib'
    configure.ac:12: installing './config.guess'
    configure.ac:12: installing './config.sub'
    configure.ac:4: installing './install-sh'
    configure.ac:4: installing './missing'
    Makefile.am: installing './depcomp'

4.  Execute configure to generate Makefile
    $ ./configure --host=ve-nec-linux
    checking for a BSD-compatible install... /usr/bin/install -c
    checking whether build environment is sane... yes
    checking for ve-nec-linux-strip... no
    checking for strip... strip
    checking for a thread-safe mkdir -p... /usr/bin/mkdir -p
    checking for gawk... gawk
    checking whether make sets $(MAKE)... yes
    checking whether make supports nested variables... yes
    checking for ve-nec-linux-mpicc... no
    checking for mpicc... mpicc
    checking whether the C compiler works... yes
    checking for C compiler default output file name... a.out
    checking for suffix of executables...
    checking whether we are cross compiling... no
    checking for suffix of object files... o
    checking whether we are using the GNU C compiler... yes
    checking whether mpicc accepts -g... yes
    checking for mpicc option to accept ISO C89... none needed
    checking for style of include used by make... GNU
    checking dependency style of mpicc... tru64
    checking for ve-nec-linux-mpic++... no
    checking for mpic++... mpic++
    checking whether we are using the GNU C++ compiler... yes
    checking whether mpic++ accepts -g... yes
    checking dependency style of mpic++... tru64
    checking for ve-nec-linux-mpifort... no
    checking for mpifort... mpifort
    checking whether we are using the GNU Fortran compiler... no
```

```
    checking whether mpifort accepts -g... yes
    checking for nar... /opt/nec/ve/bin/nar
    checking the archiver (/opt/nec/ve/bin/nar) interface... ar
    checking build system type... x86_64-unknown-linux-gnu
    checking host system type... ve-nec-linux-gnu
    checking that generated files are newer than configure... done
    configure: creating ./config.status
    config.status: creating Makefile
    config.status: creating config.h
    config.status: executing depfiles commands

5.  Execute make
    $ make
    make  all-am
    make[1]: Entering directory `/home/user1/example/newproject_mpi'
    source='autotools_test_c.c' object='autotools_test_c.o' libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    mpicc -DHAVE_CONFIG_H -I.     -g -O2 -c autotools_test_c.c
    mpicc  -g -O2   -o autotools_test_c autotools_test_c.o
    source='autotools_test_cpp.cpp' object='autotools_test_cpp.o'
    libtool=no \
    DEPDIR=.deps depmode=tru64 /bin/sh ./depcomp \
    mpic++ -DHAVE_CONFIG_H -I.     -g -O2 -c -o autotools_test_cpp.o
    autotools_test_cpp.cpp
    mpic++  -g -O2   -o autotools_test_cpp autotools_test_cpp.o
    mpifort  -g -c -o autotools_test_fortran.o autotools_test_fortran.f90
    mpifort  -g   -o autotools_test_fortran autotools_test_fortran.o
    make[1]: Leaving directory `/home/user1/example/newproject_mpi'

6.  Execute VE binaries (our example project generates 3 binaries for
    three sample programs)
    $ mpirun ./autotools_test_c
    Hello World from C program
    Rank ID: 0
    Number of processes: 1
    Host name: localhost
    $ mpirun ./autotools_test_cpp
    Hello World from C++ program
    Rank ID: 0
    Number of processes: 1
    $ mpirun ./autotools_test_fortran
    Hello World from Fortran program
    Rank ID:  0
    Number of processes:  1
```