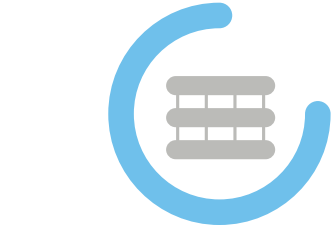

Meeting of the Technical Steering Committee (TSC) Board

Wednesday, May 22nd, 2019
11:00am ET

Meeting Logistics

- https://zoom.us/j/556149142

- United States : +1 (646) 558-8656
  - Meeting ID: 556 149 142

Antitrust Policy Notice

- Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

THE LINUX FOUNDATION

Agenda

- Reminders
  - TSC call for nominations is open till June 14th
    - submit nomination and CV to: tsc-nominations@OpenHPC.groups.io
  - ISC OpenHPC BoF
    - Wed, June 19th (8:30-9:30am)
  - PEARC'19 Tutorial: Monday afternoon (July 29, 2019)
  - SC BoF submission deadline: July 31, 2019

- Alternate build flag support: continued discussion

Alternate build flag support

- Recall some motivating discussion from previous TSC meeting:
  - existence of %{configure} macro: not really useful for us as it hard-codes install prefix
  - for autotools projects, we call ./configure directly
  - general interest expressed in potentially unifying default compiler flag options
    - right now, mostly rely on defaults from underlying build system (e.g. autotools, cmake, bjam, etc)
  - high level wish list:
    - minimize changes to existing .spec files
    - centralize default compiler flag options that we use for OHPC builds
    - allow override of default flags for users/sites who want to customize
      - need way to differentiate in resulting RPM name
      - also need way to differentiate resulting modulefile
- *Have explored a potential implementation to address the above and will walk thru for feedback/discussion*

Alternate build flag support

- Issue #1: choosing consistent build flags
  - RPM defines another macro, %{optflags}; potential starting point for build flag definition (already used by default for CFLAGS, CXXFLAGS, etc)
  - one potential issue though is that these are typically targeting gcc and may be different across distros, vary over time, and are certainly different across architectures
  - let's look at some examples

```
# rpm --eval "%{optflags}"
-O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong \
      --param=ssp-buffer-size=4 -grecord-gcc-switches
```

example from CentOS7/aarch64

```
# rpm --eval "%{optflags}"
-O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong \
      --param=ssp-buffer-size=4 -grecord-gcc-switches   -m64 -mtune=generic
```

example from CentOS7/x86_64

```
# rpm --eval "%{optflags}"
-O2 -g -m64 -fmessage-length=0 -D_FORTIFY_SOURCE=2 -fstack-protector \
      -funwind-tables -fasynchronous-unwind-tables
```

example from SLES12/x86_64

Alternate build flag support

- Issue #1: choosing consistent build flags (cont.)
  - As opposed to relying on %{optflags} directly, propose to leverage the settings and define default values in one of our existing macro files (could be OHPC_setup_compiler or OHPC_macros)
  - can be compiler_family specific (e.g. gcc, intel, arm, etc)
  - Allow external override via OHPC_CFLAGS, OHPC_CXXFLAGS, OHPC_FCFLAGS, OHPC_F77FLAGS

example relevant change for OHPC_setup_compiler

```
#------------------------------
# Set default compiler flags
# and check for user override
#------------------------------

arch=`uname -m`

if [ "$toolset" == "gcc" ];then
    DEFAULT_OPTS="-O3 -g -pipe -Wall -fexceptions -fstack-protector-strong -grecord-gcc-switches -mtune=generic"
    if [ "$arch" == "x86_64" ];then
      DEFAULT_OPTS="$DEFAULT_OPTS -m64"
    fi
elif [ "$toolset" == "intel" ];then
    DEFAULT_OPTS="-O3 -g -m64 -Wall -fexceptions -fstack-protector-strong -grecord-gcc-switches -mtune=generic"
    if [ "$arch" == "x86_64" ];then
      DEFAULT_OPTS="$DEFAULT_OPTS -m64"
    fi
fi
```

Alternate build flag support

- Issue #1: choosing consistent build flags (cont.)
  - If user provides flag override (via RPM macro or env variable), these are used instead for all of our packages that leverage %{ohpc_setup_compiler} macro
  - Look at relevant macro changes (focus on CFLAGS)

example relevant change for OHPC_macros

New

```
%global ohpc_setup_compiler %{expand:\
    %if 0%{?OHPC_CFLAGS:1} \
         export OHPC_CFLAGS=%{OHPC_CFLAGS} \
    %endif \
    %if 0%{?OHPC_CXXFLAGS:1} \
         export OHPC_CXXFLAGS=%{OHPC_CXXFLAGS} \
    %endif \
    %if 0%{?OHPC_FCFLAGS:1} \
         export OHPC_FCFLAGS=%{OHPC_FCFLAGS} \
    %endif \
    %if 0%{?OHPC_F77FLAGS:1} \
         export OHPC_F77FLAGS=%{OHPC_F77FLAGS} \
    %endif \
    . %{OHPC_ADMIN}/ohpc/OHPC_setup_compiler %{compiler_family} \
    %if 0%{?ohpc_mpi_dependent} == 1 \
        . %{OHPC_ADMIN}/ohpc/OHPC_setup_mpi %{mpi_family} \
    %endif \
}
```

Orig.

example relevant change for OHPC_setup_compiler

```
if [ -z "$OHPC_CFLAGS" ];then
    export CFLAGS=$DEFAULT_OPTS
else
    export CFLAGS=$OHPC_CFLAGS
    echo "--> CFLAGS   (user override) = $CFLAGS"
fi
```

Alternate build flag support

- Issue #2: allow naming override
  - previous changes allow fairly simple way for user to override compilation flags without touching .spec file (in most cases)
  - user may want to have this build co-installed with default ohpc variant
  - in that case, need way to include additional delimiter
  - can accommodate this thru change to OHPC_macros and changes to .spec file (leverage fact that we already rely on %{PROJ_DELIM} macro to add "ohpc" delimiter to packages
  - triggered via macro setting of OHPC_CUSTOM_DELIM

```
# check if user desires to override package and modulefile naming with
# custom delimiter (e.g optimized micro-architecture build)

%global OHPC_CUSTOM_PKG_DELIM %{nil}

%{?OHPC_CUSTOM_DELIM: %global OHPC_CUSTOM_PKG_DELIM -%{OHPC_CUSTOM_DELIM}}
%{?OHPC_CUSTOM_DELIM: %global PROJ_DELIM -%{OHPC_CUSTOM_DELIM}%{PROJ_DELIM}}
```

example relevant change for OHPC_macros

Alternate build flag support

- Issue #2: allow naming override (cont.)
  - changes to each .spec file would be required to accommodate naming override
  - two changes required:
    - update install path definition
    - update module name definition
  - consider simple autotools package changes below

example changes to .spec file

```
# Default library install path
%define install_path %{OHPC_LIBS}/%{compiler_family}/%{pname}%{OHPC_CUSTOM_PKG_DELIM}/%version

...


# OpenHPC module file
%{__mkdir} -p %{buildroot}%{OHPC_MODULEDEPS}/%{compiler_family}/%{pname}
%{__cat} << EOF >
%{buildroot}/%{OHPC_MODULEDEPS}/%{compiler_family}/%{pname}/%{version}%{OHPC_CUSTOM_PKG_DELIM}
```

Alternate build flag support

- Let's see this in action
  - have simple autotools package setup in my personal OBS project with these changes in place
    - opt_standard: standard (ohpc-style) package build using default build flags
    - opt_custom: customized build with alternate C compile flags and package name
  - Use our standard _link file overrides to define opt_custom

```
<link project='home:Admin' package='opt_standard'>
<patches>
  <topadd>%define OHPC_CUSTOM_DELIM nonzippy</topadd>
  <topadd>%define OHPC_CFLAGS "-O0 -g"</topadd>
</patches>
</link>
```

example _link file for OBS

  - also simple to do via local command-line (e.g user downloads src rpm and wants to rebuild)

```
# rpmbuild -bb --define 'OHPC_CFLAGS "-O0 -g"' --define "OHPC_CUSTOM_DELIM nonzippy" example.spec
```

Alternate build flag support

- Let's see this in action: resulting RPM names
  - example-gnu8-ohpc-1.0-12.2.ohpc.1.3.6.x86_64.rpm
  - example-gnu8-nonzippy-ohpc-1.0-19.1.ohpc.1.3.6.x86_64.rpm

- These two RPMs can be installed simultaneously (since we altered the install_path) and user sees both available in modules:

```
$ module avail example

---------------------------------- /opt/ohpc/pub/moduledeps/gnu8 ----------------------------------
   example/1.0-nonzippy    example/1.0 (D)

  Where:
   D:  Default Module
```

builds visible to user

Alternate build flag support

- Thoughts/feedback on this approach?
- Potential discussion points:
  - O3 vs. O2 (relying on -mtune=generic?)
  - restrict to packages that rely on an ohpc compiler_family? (these packages have modulefiles by definition)
  - willingness to update majority of current .spec files? (albeit fairly small changes)
  - try this out with some current packages with updates in 1.3.8 and get feedback? roll out uniformly for 2.0?