

Portability and Packaging

Chapel Team, Cray Inc.

Chapel version 1.16

October 5, 2017



Safe Harbor Statement



This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.



Outline

- Default Executable Name
- Support Traditional Configure/Install Workflow
- Debian Packaging
- Other Portability Improvements



Default Executable Name





Default Executable Name: Background, Effort

Background: Historically chpl produced a.out by default

```
$ chpl myprogram.chpl // wrote executable to a.out  
$ ./a.out
```

- This behavior was inherited from C compilers
- The name refers to an outdated executable format

This Effort: By default, base output name on input

```
$ chpl myprogram.chpl // usually writes executable to myprogram  
$ ./myprogram
```

- Follow other modern languages in moving beyond a.out
- Base executable name on the main module name





Default Executable Name: Impact, Status

Impact: Compiler is more user-friendly

- Easier to work with many different Chapel programs at once
- ... but it can be jarring for people used to typing ./a.out
 - possible to restore old default behavior with
`export CHPL_EXE_NAME=a.out`

Status: Output file name derives from main module name

- But confusion can still arise if module and file names differ
- Confusion can also arise if main module is an inner module





Default Executable Name: File & Module Differ

- Main module, thus executable, is MyModule:

```
// MyProgram.chpl  
module MyModule {  
    writeln("Hello from MyModule");  
}
```

```
$ chpl MyProgram.chpl  
$ ./MyProgram  
-bash: ./MyProgram: No such file  
$ ./MyModule  
Hello from MyModule
```



Default Executable Name: Main Module Is Inner



- Main module, thus executable, is InnerModule:

```
// MyModules.chpl
module MyProgram {
  writeln("init MyProgram");
  module InnerModule {
    proc main {
      writeln("main");
    }
  }
}
$ chpl MyModules.chpl
$ ./MyModules
-bash: ./MyModules: No such file
$ ./InnerModule
main
```





Default Executable Name: Future Work

- **Warn when module and file names don't match?**
 - Users of the module might expect the names to match
 - Making the names match would at least be a best practice

- **Warn for inner main module compiled without -o?**

- **Name executable after file containing main module?**
 - rather than main module itself?



Support Traditional Configure/Install Workflow





Configure/Install: Background

- **Chapel has historically used a custom build process**

```
source util/setchplenv.bash
export CHPL_COMM=gasnet
make
chpl hello.chpl -o hello
./hello
```

- **This approach has drawbacks**
 - Can be confusing
 - users don't necessarily read the documentation
 - they try ./configure but find it's not there
- **Additionally, 'make install' was requested by users**





Configure/Install: This Effort

- **Added support for configure, make install**
- **Configure**
 - Is purpose-built for Chapel, not from the autoconf/automake tool string
 - Offers helpful text output
 - Saves the current CHPL_* settings to chplconfig
 - Selects installation mode and destination directory
- **Two installation modes:**
 1. Copy \$CHPL_HOME somewhere
 - what we have used historically
 2. Install to /usr/bin, /usr/lib, /usr/share
 - this mode is important for the Debian packaging effort





Configure/Install: Impact, Next Steps

Impact: 'configure' and 'make install' available

- Adds ability to install Chapel
- Enables Debian packaging effort
- Supports the common pattern:

```
./configure  
make  
make install
```
- Also supports the [Try It Online](#) site

Next Steps:

- Add 'make uninstall'
- Continue Debian packaging effort



Debian Packaging



Debian Packaging

Background: Debian Intent To Package (ITP) under review

- An ITP is like a pull request in Debian, where a sponsor reviews

This Effort: Implemented more Debian sponsor feedback

- 'configure' & 'make install'
- Including dependencies in *source package* for future
- A few other minor updates

Impact: Closer to acceptance for *buster* (Debian 10)

- Unfortunately, Debian sponsor has moved on

Status: Waiting for a new Debian sponsor (as of writing)

- All feedback provided has been addressed

Next steps:

- Host unofficial DEBs (Debian) and PPAs (Ubuntu) until acceptance
- Continue pushing forward on Debian package



Other Portability Improvements





Other Portability Improvements

- **printchplenv improvements**
 - Output distinguishes settings from configuration file and environment
 - Infers location of CHPL_HOME
- **Added support for using Chapel on an OmniPath cluster**
 - See <https://chapel-lang.org/docs/1.16/platforms/omnipath.html>
- **Improved code conformance with C++14**
- **Improved code portability across versions of gcc**
- **Improved portability of code with respect to Cygwin**
- **Dependences**
 - Using LLVM now requires CMake
 - Third-party RE2 and thus regexp module now requires C++11





Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

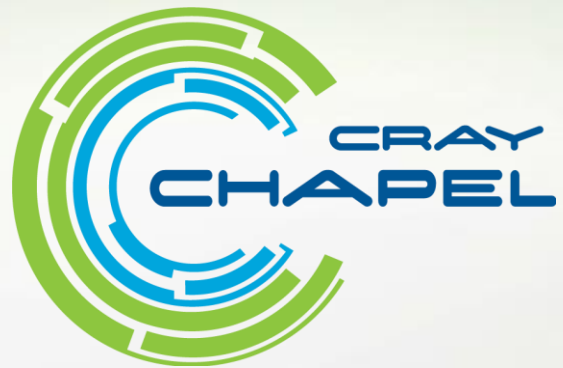
Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.





CRAY
THE SUPERCOMPUTER COMPANY