

# Documentation and Example Codes

Chapel Team, Cray Inc.  
Chapel version 1.15  
April 6, 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

- Users Guide Improvements
- Issue Tracking
- Quickstart
- Documentation Hierarchy
- Other Documentation Improvements
- Example Code Improvements



# Users Guide Improvements



# Users Guide Improvements

## Background:

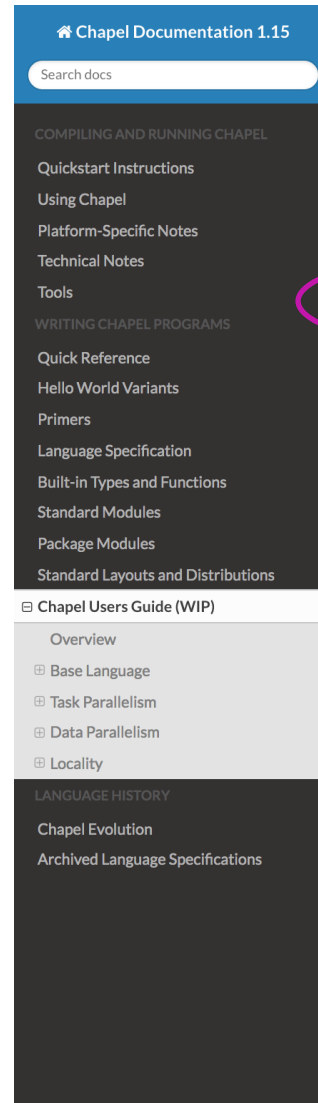
- Chapel 1.13 started including sections for a new users guide
  - each subsequent release has included additional sections

## This Effort:

- added new sections for:
  - promotion
  - constants ('const' and 'param')
  - type aliases
  - 'config' declarations
    - 'const', 'var', 'type', 'param'

## Next Steps:

- continue expanding coverage



Chapel Documentation 1.15

Search docs

COMPILING AND RUNNING CHAPEL

- Quickstart Instructions
- Using Chapel
- Platform-Specific Notes
- Technical Notes
- Tools

WRITING CHAPEL PROGRAMS

- Quick Reference
- Hello World Variants
- Primers
- Language Specification
- Built-in Types and Functions
- Standard Modules
- Package Modules
- Standard Layouts and Distributions

Chapel Users Guide (WIP)

- Overview
- Base Language
- Task Parallelism
- Data Parallelism
- Locality

LANGUAGE HISTORY

- Chapel Evolution
- Archived Language Specifications

### Base Language

This is the core of Chapel and what remains when all features in support are removed.

- Hello world: simple console output
- Variable Declarations
- Basic Types: booleans, numbers, and strings
- Literal Values for Basic Types
- Constant Declarations
- Type Aliases
- config Declarations: command-line overrides
- Casts: explicit type conversions
- for-loops: structured serial iteration
- Zippered Iteration

(more to come...)

### Task Parallelism

These are Chapel's lower-level features for creating parallel explicitly at them.

- Task Parallelism Overview
- begin Statements: unstructured tasking
- cobegin Statements: creating groups of tasks
- coforall-loops: loop-based tasking

(more to come...)

### Data Parallelism

These are Chapel's higher-level features for creating parallelism more at data structures.

- forall-loops: data-parallel loops
- Promotion: implicit data parallelism

(more to come...)

### Locality

# Issue Tracking



# Issue Tracking: Background

- **Chapel's supported bug-reports/tracking in several ways:**
  - chapel-bugs@lists.sourceforge.net
  - JIRA
    - read-only for non-core developers
  - Bugzilla
    - Cray customers
  - STATUS.devel / future tests in repository
  
- **The options for reporting bugs were lacking**
  - Typical users could only report bugs via mailing list
    - Required transcription over to JIRA
  - More advanced users could open a PR for a future test
  - Cray users could use Bugzilla if they preferred
    - Required transcription over to JIRA when appropriate

# Issue Tracking: This Effort

- **Enabled GitHub Issues for the chapel-lang/chapel repo**
- **GitHub offers many advantages as an issue-tracker:**
  - Integration with existing GitHub repository
    - Cross-referencing Issues / PR
    - Tagging GitHub IDs
  - Low barrier of entry
    - GitHub has a large user-base already familiar with Issues
  - High visibility
    - Easily discovered from the chapel-lang/chapel repository
  - Batteries included
    - Very little setup overhead necessary
    - Has almost all the features we need out of the box
- **Started migrating JIRA issues and STATUS.devel**



# Issue Tracking: Impact

- **Increased transparency**

- More discussions are happening in public via Issues
  - Increasing the involvement of non-core developers in discussions
- Issues are also tracking tasks, illuminating future plans for the project

- **Better archiving**

- GitHub search has been a vast improvement for our team
  - Compared to JIRA and the mailing list archives
- Finding duplicates and cross-referencing them is much easier

- **‘easy’ labels provide starting point for new contributors**

- this year’s Google Summer of Code applicants have used these





# Issue Tracking: Status and Next Steps

## Status: Active and well-received by team and community

- Most developers are finding it more productive
- Total JIRA issues: **297**
  - Started March, 2015
- Total GitHub issues: **260**
  - Started December, 2016

## Next Steps: Improve processes surrounding issue-tracking

- Migrate remaining public issues from JIRA and STATUS.devel file
- Possibly discontinue using JIRA in favor of private GitHub Issues
- Set up a long-term strategy for detecting new user-issues
- Continue to tune process
  - Milestones
  - Labels
  - Associating futures with issues



# Quickstart



# Quickstart

## Background: QUICKSTART.rst was a top-level document

- Contained many sphinx cross-references that looked odd on GitHub
- Complicated docs-build, since we included it in the html-docs
  - ‘make docs’ resulted in copying the file into the docs-directory
  - Relative file paths lost context in the docs-directory version

## This Effort: QUICKSTART.rst moved into the doc-directory

- The top-level README points to the online URL (preferred)
- The doc-level README points to restructured text file path

## Impact: Improved QUICKSTART.rst and simplified docs build

- Users are now consistently pointed to the preferred online version
  - Users are never pointed to the GitHub-rendered version
- QUICKSTART.rst does not get moved or copied in ‘make docs’

# Documentation Hierarchy





# Documentation Hierarchy

## Background: Documentation hierarchy was asymmetric

- In release, HTML documentation did not mirror text-documentation

```
doc/  
├─ *.rst  
├─ platforms/  
├─ technotes/  
└─ html/
```

- Also, developer and user views of documentation hierarchy differed

- GitHub / developer:

```
doc/  
├─ release/           // things intended for the release  
│   └─ foo.rst  
├─ developer/        // things not intended for the release  
└─ .../              // etc.
```

- Release / user:

```
doc/  
└─ foo.rst           // release contents moved to top-level
```



# Documentation Hierarchy

## This Effort: Documentation hierarchy is now symmetric

- HTML documentation mirrors text-documentation
- Developer view is now consistent with user view

```
doc/  
├─ README.rst  
├─ rst/  
│   └─ foo.rst  
└─ html/  
    └─ foo.html
```

## Impact: Documentation is more organized and consistent

# Other Documentation Improvements





# Other Documentation Improvements

- **Online documentation**

- bugs.rst now refers users to GitHub Issues
- Improved Docker README information
- Reorganized and categorized platform-specific documentation
- Unified references to download.html rather than install.html
- Improved some entries in “Quick Reference” document

- **Library documentation**

- Documented ‘dim()’ and ‘dims()’ on arrays
- Fixed ‘string.strip()’ documentation
- Updated documentation w.r.t. memory management of files/channels
- Added indication that IO module is used by default



# Example Code Improvements



# Example Code Improvements

- **learnChapellnYMinutes**
  - Added examples, fixed an incorrect one
  - Various comment, style, and formatting improvements
- **Updated example codes to:**
  - Use new 'RandomStream()' initializer
  - Use 'deinit()' rather than destructors
  - Reflect better const-checking
  - Reflect that 'Barrier' is now a record
- **Makefiles**
  - Fixed parallel build of examples directory
  - Fixed FFTW build target
  - Added missing Makefile to examples/patterns directory





# 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