Chapel: Hands-On Session

Chapel Team

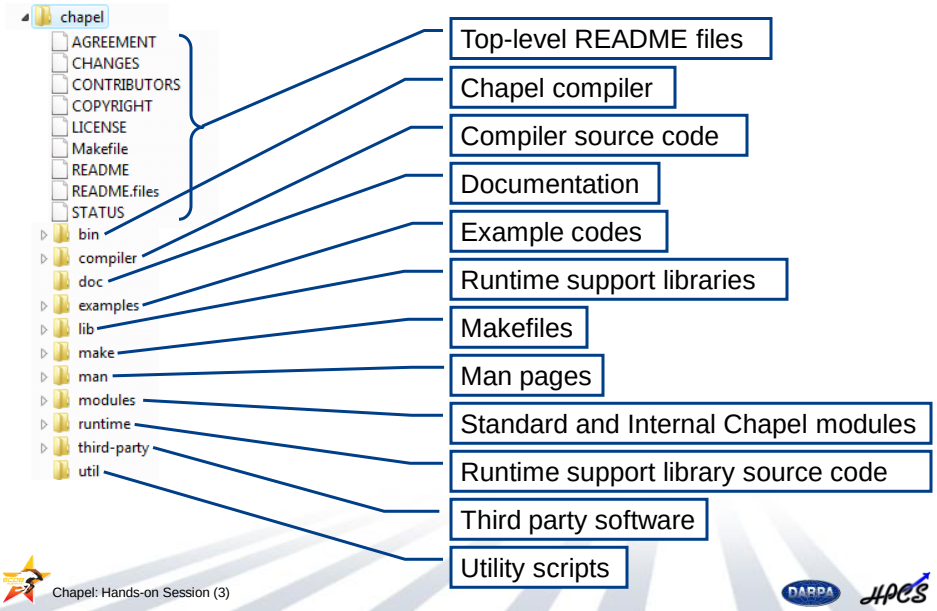
SC08: Tutorial S07 – 11/16/08

Outline

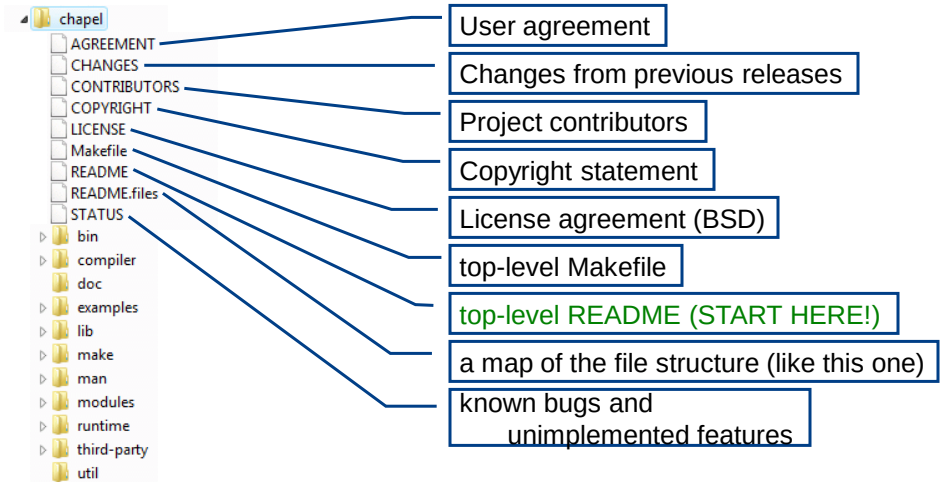
- Overview of the release structure
  - overall structure
  - documentation structure
  - examples structure
- Getting started with the hands-on session
- Chapel environment settings

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Release Directory Structure

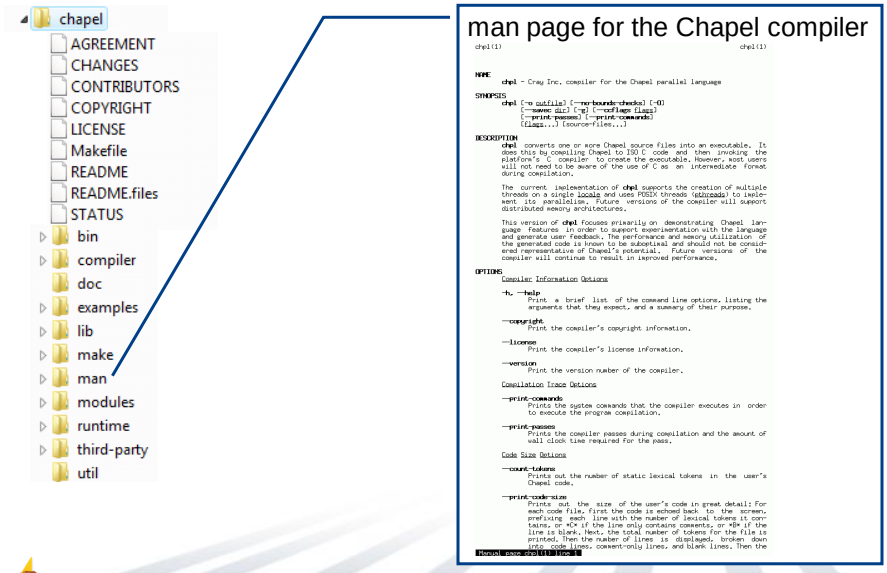


Top-Level Documentation

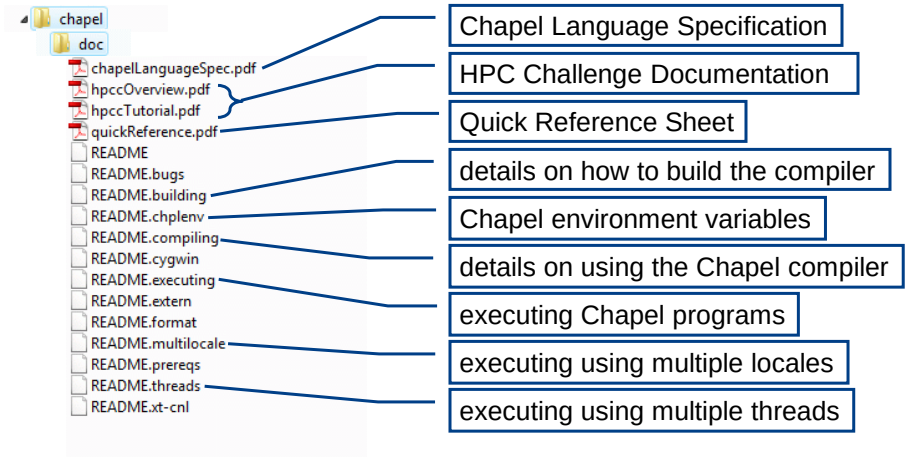


Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

Man Page

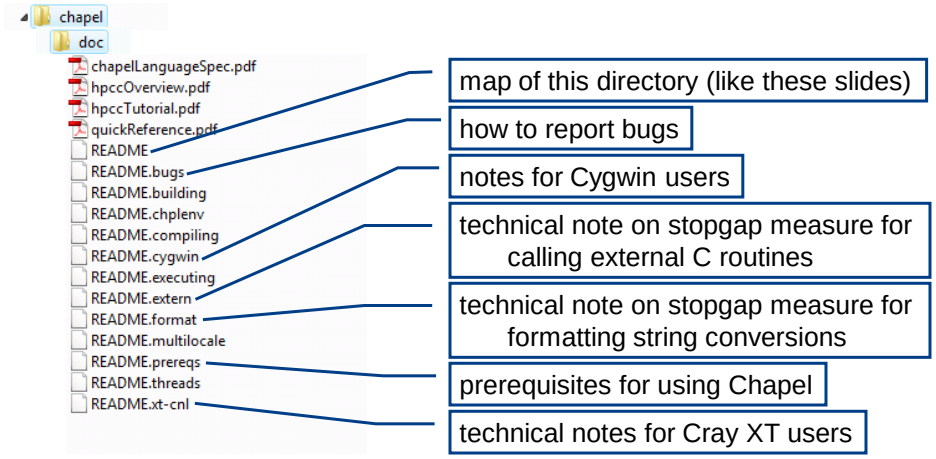


Doc Directory: Main files

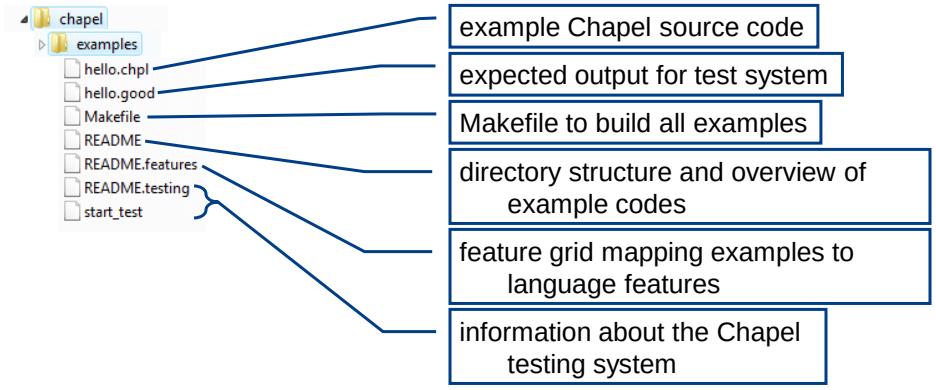


**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Doc Directory: Other files

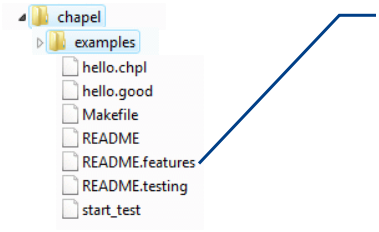


Examples Directory: Notable files



Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

Examples Directory: Feature Grid



test	frall	prmte	slice	reind	reduc	scan	xxxxx	xxxxx
	Data Parallel Features							
hello								
hello-module								
beer	x							
blockLU	x		x	*	x			
fileIO					x			
genericStack								
iterators								
jacobi	*		*		*			
linkedList								
norm	x	x						
prodCons								
quicksort			x					
reductions		*	*		x			
slices	*	*	x	x				
sparse	x				x			
tree								
hpcc/stream	x	*			x			
hpcc/ra	x				x			
hpcc/fft	x	x	x		x			

key:
- frall = uses forall loops
- prmte = uses promotion and/or whole-array operators/assignment
- slice = uses array slices
- reind = uses array reindexing/array views
- reduc = uses reductions
- scan = uses scans

Outline

- Overview of the release structure
- Getting started with the hands-on session
  - platform notes
  - getting started
  - then what?
- Chapel environment settings

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Supported Platforms for hands-on session

- **Linux, Mac, UNIX users:** should have no problems
- **Windows users:** have three options:
  - use Cygwin (UNIX emulation environment)
    - works fairly well in practice, particularly for experienced users
    - get a copy of README.cygwin from us before getting started
    - we can help you install Cygwin if you're not familiar with it
  - ssh/telnet into a UNIX platform and work there
  - find someone to buddy up with
- **No computer?** find someone to buddy up with

Steps to getting started (from the README)

1. Make sure you're in the `chapel/` directory
2. Build the compiler and runtime libraries using `gmake`
   - or `make` if your copy is GNU-make-compatible (as on Cygwin)
3. Set up your shell's environment to use Chapel
   - if you use… then type…

     …csh, tcsh `source util/setchplenv.csh`

     …bash `source util/setchplenv.bash`

     …sh `. util/setchplenv.sh`

     …something else? Come talk to us
4. Compile an example program using:

   ```
   chpl -o hello examples/hello.chpl
   ```
5. Execute the resulting program:

   ```
   ./hello
   ```


**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Then what?

- Whatever you like:
  - Look at, compile, execute other example programs
  - Explore the release -- see the bottom of the README for pointers
  - Try coding up an algorithm of interest to you
  - Work through some of the exercises we've prepared
- Please ask us questions if you have any difficulties
  - (or simply questions)
- Reminders:
  - break at 3pm
  - we'll reconvene at 4:30pm for a final Q&A and to get your feedback

Outline

- Overview of the release structure
- Getting started with the hands-on session
- Chapel environment settings
  - main settings
  - cross-compilation settings
  - other settings

Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

Main Chapel Environment Settings

CHPL_HOME: points to location of `chapel/` directory
- **default:** none
- **typical values:** `~/chapel`, `/cygdrive/c/chapel`, or any path

CHPL_HOST_PLATFORM: architecture on which compiler is built, run
- **default:** a best guess is made using uname -a
- **typical values:** cygwin, darwin, linux, linux64, sunos

PATH: the Chapel compiler's path should be added to yours
- **default:** none
- **typical value**: `$CHPL_HOME/bin/$CHPL_HOST_PLATFORM`

MANPATH: Chapel's man page path should be added to yours
- **default:** none
- **typical value**: `$CHPL_HOME/man`

(See `$CHPL_HOME/doc/README.chplenv` for more detail)

Cross-Compilation Environment Variables

CHPL_TARGET_PLATFORM: architecture for which Chapel is compiled
- **default:** $CHPL_HOST_PLATFORM
- **typical values:** mta, x1, x2, xmt, xt-cnl

CHPL_HOST_COMPILER: compiler to use for the host platform

CHPL_TARGET_COMPILER: compiler to use for the target platform
- **default:** a best guess is made using the corresponding PLATFORM variable
- **typical values:** gnu, intel, pathscale, pgi, cray-mta, cray-vec, cray-xt-gnu, cray-xt-pathscale, cray-xt-pgi

CHPL_MAKE: the GNU-compatible make utility to use for the target
- **default:** a best guess is made using the PLATFORM variables
- **typical values**: gmake, make

(See `$CHPL_HOME/doc/README.chplenv` for more detail)

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Other Environment Variables

CHPL_THREADS: threading layer to use for the generated code
- **default:** a best guess is made using $CHPL_TARGET_PLATFORM
- **typical values:** none, pthreads, mta

CHPL_COMM: communication layer to use for the generated code
- **default:** none
- **typical values:** none, gasnet, armci

CHPL_*: most compiler options can be set using an environment variable
- see `chpl --help-env` and `--help` for details

(See `$CHPL_HOME/doc/README.chplenv` for more detail)

Questions?

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**