



COLLEGE OF ENGINEERING  
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES  
UNIVERSITY OF MICHIGAN

# Lecture 24

## Miscellaneous Topics

Prof. Brendan Kochunas  
12/2/2019

NERS 590-004



# Outline

- Formatting Guidelines
- Documentation
- Copyright and Licensing Resources
- Other Resources
  - Allocations
  - Training & Community



## Today's Learning Objectives

- Understand importance of formatting and documentation
- Understand basics of software licenses and where to get more info
- Gain awareness about resources available to you beyond this course



## Why should I care about formatting & documentation?

- My formatting is best because it is the best, the other developers should write code like me.
- I don't need to document this, I wrote it, I understand how it works and my memory is really good (I'm a doctor).
  - But what if someone else needs to learn the code?
  - Will you remember how it works in 5 years? 10 years?
  - Do you remember code you wrote as a freshman?
- ***All of you will spend more time in your careers reading code than writing code.***
- ***NO ONE will use your code if they cannot understand it.***



# Licenses and Copyrights

- You spend a lot (*A LOT*) of time writing software.
  - If its valuable (and we should hope that it is), we should receive compensation for it.
- Copyrights, Licensing, and Patents make sure you get credit (legally) for your ideas
- When you use other software in your code (e.g. third party libraries) the license defines rules for how the software can be used.



# Formatting



## What do we desire from “good” formatting guidelines?

- Should be easily readable
- Should be easily understandable
- Does not make documenting code more difficult than it needs to be
- Makes searching the source easy
- Re-enforces good code behavior
  - Predictable
  - no memory leaks
  - efficient
- **Bottom Line**: Should enable developers to read and write code more efficiently



# Common things to address in formatting

- Whitespacing
- Naming conventions
  - Files
  - Routines
  - variables
  - classes
- Comments
- Metrics for evaluating guidelines and practices for formatting
  - Difficult problem with no clear single answer.





# Whitespacing

- Line endings
  - Unix/Linux line endings (LF) work with almost anything (except Windows notepad)
  - Windows Line Endings (CRLF) often have trouble outside Windows
  - Most editors support automatic conversion (and there's also the dos2unix program)
- Tabs v. Spaces
  - Editors may show tabs as different field lengths, spaces are more explicit
  - Watch this for a humorous analysis: <https://www.youtube.com/watch?v=SsoOG6ZeyUI>
- Indentation
  - Purpose is to show logical structure. This has been studied and results suggest 2 to 4 is optimal.
- Line length & carry-over
  - Prevents side-scrolling and automatic word wrap
  - Choose a length that is appropriate for how the source will be viewed
    - e.g. most terminals default to 80 columns



## Fundamental Theorem of Formatting

***“Good visual layout shows the logical structure of a program”***

- Accurately represent logical structure of code
- Consistently represent logical structure of code
- Example: This is “baked into” Python.



# Naming Conventions

- When to use:
  - multiple programmers on a project
  - when you plan to hand off source code to another person
  - when source is to be reviewed
  - when source is too large to hold in your head all at once (~40-50K lines)
  - No real downside, so just do this always.
- We saw that several libraries have naming conventions (MPI, HDF5, PETSc, others)
  - Naming conventions can speed up comprehension and allow for “self-documenting” code
  - This can lead to less comments elsewhere.



## Naming Conventions cont.

- Names should be readable, memorable, and appropriate
- For mathematical equations name variables following notation
- Name variables after “what” not “how”
  - “what” refers to the computing solution, “how” refers to the problem domain
  - e.g. Employee record: inputRec or employeeData
- Length of name is recommended at 10-16 characters
  - short names imply scratch variables or limited scope (e.g. loop index)
- Do not reuse local variables for multiple things.
  - Each variable should have its own purpose (loop indices may be exception)



## Good names and Bad names

```
subroutine calcb(id,b,a,x)
```

```
    b=b-x
```

```
    y=a+stax(a)
```

```
    x=x+penalty(id,b)+y
```

```
    b=b+int(id,b)
```

```
end
```

```
subroutine ComputeCustomerBill(
```

```
    customerID,balance,
```

```
    newPurchases,
```

```
    lastPayment)
```

```
    balance=balance-lastPayment
```

```
    monthlyTotal=newPurchases+
```

```
        SalesTax(newPurchases)
```

```
    balance=balance+
```

```
        LateFee(customerID,balance)+
```

```
        monthlyTotal
```

```
    balance=balance+
```

```
        Interest(customerID,balance)
```

```
end
```



## Some Tips and Common Conventions

- Qualifiers are added at the end:
  - e.g. SolveDirect and Solvelterative (not IterativeSolve and DirectSolve)
- Avoid names that conflict with language keywords or intrinsics
- CamelCase improves readability
- Variables start with lower case
- Routines start with upper case
- Compile-time constants appear in ALL CAPS
- “\_t” is appended to C structs
- Trailing “\_” for private attributes on classes



## Example: Python PEP-8 Style Guide

- <https://www.python.org/dev/peps/pep-0008/>



# Documentation

“Can’t someone else do it?”

*- Homer Simpson*





# Taxonomy for Types of Comments

- Repeat of code
  - Useless, Restatement of source code without additional info
- Explanation of code
  - Typically used for complex code. Often useful, but better to make code less complicated
- Marker in code
  - Must be standardized to be usable
- Summary of code
  - Distills a few lines or block of code into 1 or 2 sentences. Often very helpful
- Description of code's intent
  - Make sure these are focused on “problem” not “solution”
- Information not possible to express by the code
  - copyright notices, references to requirements or external documentation, markup needed by Doxygen or Sphinx or other tools.



# Comment Examples

## Summary

```
// swap the roots  
oldRoot = root[0];  
root[0] = root[1];  
root[1] = oldRoot;
```

## Explanation

```
blockSize = optimalBlockSize (numItems, sizePerItem );
```

```
/* The following code is necessary to work around an error in  
WriteData() that appears only when the third parameter  
equals 500. '500' has been replaced with a named constant  
for clarity */
```

```
if ( blockSize == WRITEDATA_BROKEN_SIZE ) {  
    blockSize = WRITEDATA_WORKAROUND_SIZE;  
}  
WriteData ( file, data, blockSize );
```

## Repeat

```
memoryToInitialize = MemoryAvailable();    // get amount of memory available  
pointer = GetMemory( memoryToInitialize ); // get a ptr to the available mem.  
ZeroMemory( pointer, memoryToInitialize ); // set memory to 0
```

## Marker

```
return NULL; // ***** NOT DONE! FIX BEFORE RELEASE!
```



# Tools for automating generation of documentation

- Doxygen (<http://www.stack.nl/~dimitri/doxygen/>)
  - Used by PAPI, LAPACK, HDF5, Metis/ParMetis, Trilinos, and many others.
  - Supports C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL, Tcl, and to some extent D
- Sphinx (<http://www.sphinx-doc.org>)
  - Primarily Python oriented
- Basic idea:
  - Program processes source code and outputs marked-up comments into more readable format
    - e.g. HTML, LaTeX, PDF
- Tools are very powerful
  - e.g. Change a user input option, change comment in code. Generate new user manual immediately



## Examples

- [https://support.hdfgroup.org/HDF5/doc/cppplus\\_RM/](https://support.hdfgroup.org/HDF5/doc/cppplus_RM/)
- <http://www.netlib.org/lapack/explore-html/>



# Copyright & Licensing



# Copyrights and Licensing

- First resource: UM's Office of Tech Transfer <http://techtransfer.umich.edu>
- Copyright
  - *Copyright is a form of protection provided by the laws of the United States to the authors of "original works of authorship." This includes literary, dramatic, musical, artistic, and certain other intellectual works as well as computer software. This protection is available to both published and unpublished works. The Copyright Act generally gives the owner of copyright the exclusive right to conduct and authorize various acts, including reproduction, public performance and making derivative works.*
- Licensing
  - *License Agreements grant third parties the rights under U-M patents or copyrights. University license agreements typically stipulate that the licensee should diligently seek to bring the intellectual property into commercial use for the public good, provide a reasonable return for U-M, and specify limitations to U-M's liability.*



## Example From MPACT

- In MPACT the following appears at the top of every source file

```
!+++++!  
!                                     Copyright (C) 2012                               !  
!                               The Regents of the University of Michigan                !  
!                               MPACT Development Group and Prof. Thomas J. Downar      !  
!                               All rights reserved.                                   !  
!                                                                                       !  
! Copyright is reserved to the University of Michigan for purposes of                 !  
! controlled dissemination, commercialization through formal licensing, or           !  
! other disposition. The University of Michigan nor any of their employees,          !  
! makes any warranty, express or implied, or assumes any liability or                !  
! responsibility for the accuracy, completeness, or usefulness of any                !  
! information, apparatus, product, or process disclosed, or represents that          !  
! its use would not infringe privately owned rights. Reference herein to any         !  
! specific commercial products, process, or service by trade name, trademark,       !  
! manufacturer, or otherwise, does not necessarily constitute or imply its          !  
! endorsement, recommendation, or favoring by the University of Michigan.           !  
!+++++!
```





# Getting Information on types of Licenses






- Too Long Didn't Read (TLDR) Legal: <https://tldrlegal.com/>
- Licenses define a lot of ground rules for distribution, modifications, commercial use, liability, trademark, and ability to modify.
  - Important to understand how these license can work together if your software uses other software.
- Common Open Source Licenses:
  - MIT License
  - BSD
  - FreeBSD
  - GPL
  - LGPL
  - Apache






# MIT License

- Developed to encourage faculty to “walk their inventions out the front door, rather than sneak them out the back door”.
- A short, permissive software license. Basically, you can do whatever you want as long as you include the original copyright and license notice in any copy of the software/source. There are many variations of this license in use.

Can	
▶ Commercial Use	
▶ Modify	
▶ Distribute	
▶ Sublicense	
▶ Private Use	

Cannot	
▶ Hold Liable	

Must	
▶ Include Copyright	
▶ Include License	



# MIT License: Full Text

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



# Berkeley Software Distribution (BSD)

- Developed at UC Berkeley for Unix operating system
  - Several versions: 1 clause, 2 clause (FreeBSD), 3 clause
- FreeBSD (Most common)
  - The BSD 2-clause license allows you almost unlimited freedom with the software so long as you include the BSD copyright notice in it (found in Fulltext). Many other licenses are influenced by this one.

## Can

▶ Commercial Use



▶ Modify



▶ Distribute



▶ Place Warranty



## Cannot

▶ Hold Liable



## Must

▶ Include Copyright



▶ Include License











# GNU General Public License (GPL)

- Developed as part GNU project and free software foundation.
- Several versions: v1.0, v2.0, v3.0
- GPL v2.0 (most common)
  - You may copy, distribute and modify the software as long as you track changes/dates in source files. Any modifications to or software including (via compiler) GPL-licensed code must also be made available under the GPL along with build & install instructions.

## Can

- ▶ Commercial Use 
- ▶ Modify 
- ▶ Distribute 
- ▶ Place Warranty 

## Cannot

- ▶ Sublicense 
- ▶ Hold Liable 














## Must

- ▶ Include Original 
- ▶ Disclose Source 
- ▶ Include Copyright 
- ▶ State Changes 
- ▶ Include License 



# Apache License

- Developed as a part of Apache Webserver software
  - Several version: 1.0, 2.0
- You can do what you like with the software, as long as you include the required notices. This permissive license contains a patent license from the contributors of the code.

Can	Cannot	Must
<ul style="list-style-type: none"><li>▶ Commercial Use </li><li>▶ Modify </li><li>▶ Distribute </li><li>▶ Sublicense </li><li>▶ Private Use </li><li>▶ Use Patent Claims </li><li>▶ Place Warranty </li></ul>	<ul style="list-style-type: none"><li>▶ Hold Liable </li><li>▶ Use Trademark </li></ul>	<ul style="list-style-type: none"><li>▶ Include Copyright </li><li>▶ Include License </li><li>▶ State Changes </li><li>▶ Include Notice </li></ul>



# Contributor License Agreement

- Important if you intend to have people outside your institution make contributions.
  - Some examples: OpenMPI, Trilinos (based on OpenMPI)
  - Set's rules for ownership of IP from contributors
- <https://www.open-mpi.org/community/contribute/open-mpi-individual-contributor-agreement.pdf>
- Seem to be losing popularity. Most permissive licenses include clauses for contribution (e.g. Apache v2.0)



## Licenses & Contribution Templates

- Github now lets you initialize a project with a license
- Licensing resource curated by Github: <https://choosealicense.com>
- Contributing guidelines
  - <https://help.github.com/articles/setting-guidelines-for-repository-contributors/>



# Other Resources





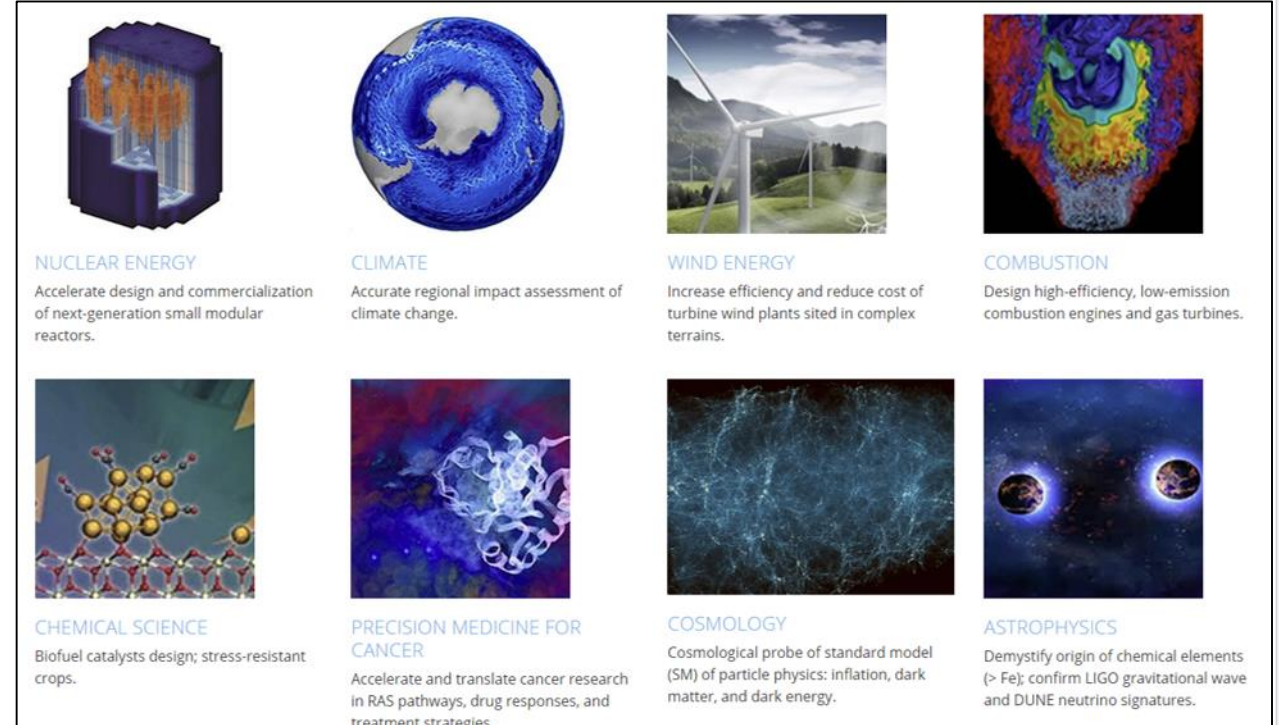
## XSEDE: <https://www.xsede.org>

- First point of contact is UM's Campus Champion: Brock Palen @ ARC-TS
- Resources
  - Comet (San Diego Supercomputing Center)
  - Stampede2 (Texas Advanced Computing Center)
  - SuperMIC (LSU) – Another Xeon Phi machine
  - XSTreme (Stanford)
  - Bridges (Pittsburg Supercomputing Center)
  - Jetstream (IU/TACC)
- Allocations (usually have least requirements)
  - Education & Startup – pretty small
  - Research – Large
  - Campus Champion – similar to trial
  - Trial - small

# Exascale Computing Project: <https://www.exascaleproject.org/>



- Sponsored by US DOE Office of Science and National Nuclear Security Administration
- Focus Areas
  - Application Development
  - Software Technology
  - Hardware & Integration





## IDEAs Project: <https://ideas-productivity.org/>

- Mission:
  - “The IDEAS Project has specific use case requirements, an ambitious effort to improve compatibility and usability of important DOE libraries, improve the practices, processes, and tools for scientific software development, and improve community knowledge.”
- Focus Areas:
  - eXtreme Scale Development Kit (xSDK)
    - Portable, TriBITS based build environment for common software stack
  - HowTo
    - Guides for Software Engineering in Scientific Software
  - Outreach
    - Developing and executing training for DOE/ASCR computing facilities





## Better Scientific Software ([bssw.io](https://bssw.io))

- Intro to CSE and HPC
- Numerous Webinars
  - Practical Introduction to Debugging
  - An Introduction to Software Licensing
  - Best Practices for HPC Software Developers Webinar Series
- Largely aggregates material they have sponsored



OLCF: <https://www.olfc.ornl.gov>

- Resources

- Titan – Largest open science computer in the US (Cray XK7)
  - Compute nodes have 16-core AMD chips and GPU's, 2GB RAM/core
  - 299,008 cores, need to effectively use GPUs to get allocations
- Eos – Conventional HPC cluster (Cray XC30)
  - Compute nodes have 16 physical cores, Intel chips w/ hyperthreading up to 32, 64 GB RAM per node
  - 11,776 cores
- Rhea – Visualization cluster
  - 8192 cores, 8 GB/core, fast connections to lustre file system

- Allocations

- Director's Discretionary – small award (< 1M core hours), available all the time, like a “start-up” award, up to 1 yr
- ALCC – medium scale (~10M core hours), available once per year, for some dev but mostly production
- INCITE – very large award (> 100M core hours), available once per year, production only. For grand challenge type problems.





## ALCF: <https://www.alfc.anl.gov>

- Resources

- Theta – Newest machine (Cray/Intel)
  - 3,240 nodes w/ 64 core processors (207,360 core). Uses second generation Xeon Phi
- Mira – Large production machine (BlueGene/Q)
  - 786,432 processors, 1GB/core, IBM PowerPC
- Vesta – test and Development platform (BlueGene/Q)
  - 32,768 cores, 1GB/core, IBM PowerPC
- Cetus – Debug cluster for Mira (BlueGene/Q)
  - 65,536 cores, 1GB/core, IBM PowerPC
- Cooley – Visualization Cluster (Cray)
  - 12 cores per node, 2 GPUs (K80 Tesla's) per node, 384 GB per node

- Allocations

- Director's Discretionary – small award (< 1M core hours), available all the time, like a “start-up” award, up to 1 yr
- ALCC – medium scale (~10M core hours), available once per year, for some dev but mostly production
- INCITE – very large award (> 100M core hours), available once per year, production only. For grand challenge type problems.
- Early Science Program (ESP) – Very small, access to new machines during installation and prior to production



NERSC: <https://www.nersc.gov>

- Resources

- Edison – Big Conventional Cluster (Cray XC30)
  - 24 core Intel Haswell @ 2.4 GHz, 2.6 GB/core
  - 133,824 cores
- Cori – Newest machine w/ 2<sup>nd</sup> General Xeon Phi's (Cray XC40)
  - 63,616 conventional cores, 4 GB/core
  - 9,304 Xeon Phi compute nodes
- PDSF, Genepool – Special application clusters

- Allocations

- DOE Production – CSGF, SBIR, SciDAC, DOE supported
- ALCC - medium scale (~10M core hours), available once per year, for some dev but mostly production
- Director's award – appears to be an internal award
- Education & Startup – small award (< 50,000 cpu hours), ongoing, 18 month award



# Additional Training

- All of these computing centers do education and outreach
  - Continually holding webinars and training sessions, past sessions are available
  - Some come with certifications you can put on your CV
  - ***Highly recommend*** you utilize these resources to gain more depth on particular topics covered in this course.
    - <https://www.olcf.ornl.gov/support/training-events/>
    - <https://www.alcf.anl.gov/training>
    - <http://www.nersc.gov/users/training/>
    - <https://portal.xsede.org/training/overview>
- Summer Schools
  - ATPESC (July 26 – Aug 7 2020, Chicago area) – <https://extremecomputingtraining.anl.gov/>
    - 2-week annual summer school, premier training, must apply, all expenses paid, held in Illinois
    - Videos from past years are online!
    - very intense and most comprehensive
  - IHPCSS (July 12 – 17 2020, Toronto) - <http://www.ihpcss.org/>
    - 1-week annual summer school, also prestigious, must apply, a lot of expenses (all?) are covered, rotates around the world
    - Applications are open!





IHPCSS – 2011, South Lake Tahoe



ATPESC – 2013

