$ cat welcome.c
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("COS 217\n");
    printf("Introduction to Programming Systems\n\n");
    printf("Fall, 2018\n");
    return 0;
}

$ gcc217 welcome.c -o welcome
$ ./welcome
COS 217
Introduction to Programming Systems
Fall, 2018
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
Introductions

Lead Instructor
• Andrew Appel  appel@princeton.edu

Lead Preceptor
• Xiaoyan Li  xiaoyan@cs.princeton.edu

Faculty Preceptor
• Donna Gabai  dgabai@cs.princeton.edu

Preceptors
• Seo Young Kyung  skyung@princeton.edu
• Austin Le  austinle@princeton.edu
• Logan Stafman  stafman@princeton.edu
• Alberto Mizrahi Benmaman  albertob@princeton.edu
• Jiashuo Zhang  jiashuoz@princeton.edu
Agenda

Course overview
• Introductions
• **Course goals**
• Resources
• Grading
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
Goal 1: “Programming in the large”
  • Help you learn how to compose large computer programs

Topics
  • Modularity/abstraction, information hiding, resource management, error handling, testing, debugging, performance improvement, tool support
Goal 2: Under the Hood

Learn what happens “under the hood” of computer systems

Learn “how to be a client of an operating system”

Downward tours

C Language

Assembly Language

Machine Language

language levels tour

Application Program

Operating System

Hardware

service levels tour
Modularity!
Goals: Summary

Help you to become a...

Power Programmer!!!
Question: Why C instead of Java?

Answer 1: Primary language for “under the hood” programming

Answer 2: Knowing a variety of approaches helps you “program in the large”
Goals: Why Linux?

Question: Why use the Linux operating system?

Answer 1: Linux is the industry standard for servers and embedded devices

Answer 2: Linux (with GNU tools) is good for programming (which helps explain answer 1)
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
Lectures

- Describe material at conceptual (high) level
- Slides available via course website

Lecture etiquette

- Let’s start on time, please
- Please don’t use electronic devices during lectures
- If you must phiddle with your phone or laptop, sit in the back row where you won’t distract other students
The Pen Is Mightier Than the Keyboard
Advantages of Longhand Over Laptop Note Taking

Pam A. Mueller
Daniel M. Oppenheimer

1Princeton University
2University of California, Los Angeles

Pam A. Mueller, Princeton University, Psychology Department, Princeton, NJ 08544 E-mail: pamuelle@princeton.edu

Abstract
Taking notes on laptops rather than in longhand is increasingly common. Many researchers have suggested that laptop note taking is less effective than longhand note taking for learning. Prior studies have primarily focused on students’ capacity for multitasking and distraction when using laptops. The present research suggests that even when laptops are used solely to take notes, they may still be impairing learning because their use results in shallower processing. In three studies, we found that students who took notes on laptops performed worse on conceptual questions than students who took notes longhand. We show that whereas taking more notes can be beneficial when learners do not need to process information, when they do, taking more notes impairs performance.

FEEDBACK
Lectures

iClicker

- Please obtain one and register in Blackboard (not with iClicker – they’ll charge you)
- Occasional questions in class, graded on participation (with a generous allowance for not being able to attend)
Q: Do you have an iClicker with you today?

- A. Yes
- B. No, but I’ve been practicing my mental electrotelekenisis and the response is being registered anyway
- C. I’m not here, but someone is iClicking for me (don’t do this! it’s academic fraud)
Precepts

Precepts
• Describe material at the “practical” (low) level
• Support your work on assignments
• Hard copy handouts distributed during precepts
• Handouts available via course website

Precept etiquette
• Attend your precept – attendance will be taken
• Use SCORE to move to another precept
  • Trouble ⇒ See Colleen Kenny (CS Bldg 210)
  • But Colleen can’t move you into a full precept
• Must miss your precept? ⇒ inform preceptors & attend another

Precepts begin next week! (No precept this week)
Website

  - Princeton CS → Courses → Course Schedule → COS 217
  - Home page, schedule page, assignment page, policies page
Piazza

- http://piazza.com/class#spr2018/cos217/
- Instructions provided in first precept

Piazza etiquette

- Study provided material before posting question
  - Lecture slides, precept handouts, required readings
  - Read all (recent) Piazza threads before posting question
- Don’t show your code!!!
  - See course policies
Books

**The Practice of Programming** *(recommended)*
- Kernighan & Pike
- “Programming in the large”

**Computer Systems: A Programmer’s Perspective (Third Edition)** *(recommended)*
- Bryant & O'Hallaron
- “Under the hood”
  *out of stock until oct 16th, a few used copies at Labyrinth, readings available in Blackboard*

**C Programming: A Modern Approach** *(Second Edition)* *(required)*
- King
- C programming language and standard libraries
Manuals

Manuals (for reference only, available online)

- Intel 64 and IA-32 Architectures Software Developer’s Manual, Volumes 1-3
- Intel 64 and IA-32 Architectures Optimization Reference Manual
- Using as, the GNU Assembler

See also

- Linux man command
Programming Environment

Server

CourseLab Cluster

Linux
GNU
Your Pgm

courselab01
courselab02

Client

Your Computer

On-campus or off-campus

SSH
Agenda

Course overview

• Introductions
• Course goals
• Resources
• **Grading**
• Policies
• Schedule

Getting started with C

• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
Grading

<table>
<thead>
<tr>
<th>Course Component</th>
<th>Percentage of Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments *</td>
<td>50</td>
</tr>
<tr>
<td>Midterm Exam **</td>
<td>15</td>
</tr>
<tr>
<td>Final Exam **</td>
<td>25</td>
</tr>
<tr>
<td>Participation ***</td>
<td>10</td>
</tr>
</tbody>
</table>

* Final assignment counts double; penalties for lateness

** Closed book, closed notes, no electronic devices

*** Did your involvement benefit the course as a whole?
  - Lecture/precept attendance and participation counts

These percentages are approximate
Programming Assignments

Programming assignments
(some individual, some done with a partner from your precept)

0. Introductory survey
1. “De-comment” program
2. String module
3. Symbol table module
4. Assembly language programs
5. Buffer overrun attack
6. Heap manager module
7. Game referee

Assignments 0 and 1 are available now

Start early!
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
Study the course “Policies” web page!

Especially the assignment collaboration policies

- Violations often involve trial by Committee on Discipline
- Typical course-level penalty is F for course
- Typical University-level penalty is suspension from University for 1 academic year
Assignment Related Policies

Some highlights:

• You may not reveal any of your assignment solutions (products, descriptions of products, design decisions) on Piazza.

• **Getting help**: To help you compose an assignment solution you may use only authorized sources of information, may consult with other people only via the course's Piazza account or via interactions that might legitimately appear on the course's Piazza account, and must declare your sources in your readme file for the assignment.

• **Giving help**: You may help other students with assignments only via the course's Piazza account or interactions that might legitimately appear on the course's Piazza account, and you may not share your assignment solutions with anyone, ever, in any form.

Ask the instructor for clarifications

• Permission to deviate from policies must be obtained in writing.
Agenda

Course overview
• Introductions
• Course goals
• Resources
• Grading
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
## Course Schedule

<table>
<thead>
<tr>
<th>Weeks</th>
<th>Lectures</th>
<th>Precepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>Number Systems C (conceptual)</td>
<td>Linux/GNU C (pragmatic)</td>
</tr>
<tr>
<td>3-6</td>
<td>Programming in the Large</td>
<td>Advanced C</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Midterm Exam</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Midterm break!</td>
</tr>
<tr>
<td>8-13</td>
<td>“Under the Hood” (conceptual)</td>
<td>“Under the Hood” (assignment how-to)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reading Period</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Final Exam</td>
</tr>
</tbody>
</table>
Questions?
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
The C Programming Language

Who?  Dennis Ritchie
When?  ~1972
Where?  Bell Labs
Why?  Build the Unix OS
Java vs. C: History
## C vs. Java: Design Goals

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the Unix OS</td>
<td>Language of the Internet</td>
</tr>
<tr>
<td>Low-level; close to HW and OS</td>
<td>High-level; insulated from hardware and OS</td>
</tr>
<tr>
<td>Good for system-level programming</td>
<td>Good for application-level programming</td>
</tr>
<tr>
<td>Support structured programming</td>
<td>Support object-oriented programming</td>
</tr>
<tr>
<td><strong>Unsafe: don’t get in the programmer’s way</strong></td>
<td><strong>Safe: can’t step “outside the sandbox”</strong></td>
</tr>
<tr>
<td></td>
<td>Look like C!</td>
</tr>
</tbody>
</table>
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
Building Java Programs

$ javac MyPgm.java

Java compiler (machine lang code)
$ java MyPgm

Java interpreter (Java virtual machine) (machine lang code)
Building C Programs

$ gcc217 mypgm.c –o mypgm

C “compiler driver”
(machine lang code)
Running C Programs

$ ./mypgm

mypgm (machine lang code)
Agenda

Course overview
- Introductions
- Course goals
- Resources
- Grading
- Policies
- Schedule

Getting started with C
- History of C
- Building and running C programs
- Characteristics of C
- C details (if time)
### Java vs. C: Portability

<table>
<thead>
<tr>
<th>Program</th>
<th>Code Type</th>
<th>Portable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyPgm.java</td>
<td>Java source code</td>
<td>Yes</td>
</tr>
<tr>
<td>mypgm.c</td>
<td>C source code</td>
<td>Mostly</td>
</tr>
<tr>
<td>MyPgm.class</td>
<td>Bytecode</td>
<td>Yes</td>
</tr>
<tr>
<td>mypgm</td>
<td>Machine lang code</td>
<td>No</td>
</tr>
</tbody>
</table>

**Conclusion:** Java programs are more portable.
Java vs. C: Safety & Efficiency

Java
- Automatic array-bounds checking,
- NULL pointer checking,
- Automatic memory management (garbage collection)
- Other safety features

C
- Manual bounds checking
- NULL pointer checking,
- Manual memory management

Conclusion 1: Java is often safer than C
Conclusion 2: Java is often slower than C
Java vs. C: Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Efficiency</td>
<td>~</td>
<td>+</td>
</tr>
<tr>
<td>Safety</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>
Q: Which corresponds to the C programming language?

- A.
- B.
- C.
Agenda

Course overview
• Introductions
• Course goals
• Resources
• Grading
• Policies
• Schedule

Getting started with C
• History of C
• Building and running C programs
• Characteristics of C
• C details (if time)
Java vs. C: Details

Remaining slides provide some details

Use for future reference

Slides covered now, as time allows…
### Java vs. C: Details

<table>
<thead>
<tr>
<th>Overall Program Structure</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello.java:</td>
<td>Hello.java:</td>
<td>hello.c:</td>
</tr>
<tr>
<td>public class Hello</td>
<td>public class Hello</td>
<td>#include &lt;stdio.h&gt;</td>
</tr>
<tr>
<td>{ public static void main</td>
<td>{ int main(void)</td>
<td></td>
</tr>
<tr>
<td>(String[] args)</td>
<td>printf(&quot;hello, world\n&quot;);</td>
<td></td>
</tr>
<tr>
<td>{ System.out.println(&quot;hello, world&quot;);</td>
<td>return 0;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Building</th>
<th>$ javac Hello.java</th>
<th>$ gcc217 hello.c –o hello</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>$ java Hello</td>
<td>$ ./hello</td>
</tr>
<tr>
<td>hello, world</td>
<td>hello, world</td>
<td>hello, world</td>
</tr>
<tr>
<td>$</td>
<td>$</td>
<td>$</td>
</tr>
</tbody>
</table>
# Java vs. C: Details

<table>
<thead>
<tr>
<th>Character type</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td><code>byte</code> // 8 bits</td>
<td><code>(unsigned) char</code></td>
</tr>
<tr>
<td>Short</td>
<td><code>short</code> // 16 bits</td>
<td><code>(unsigned) short</code></td>
</tr>
<tr>
<td>Integer</td>
<td><code>int</code> // 32 bits</td>
<td><code>(unsigned) int</code></td>
</tr>
<tr>
<td>Long</td>
<td><code>long</code> // 64 bits</td>
<td><code>(unsigned) long</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Integral types</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character type</td>
<td><code>char</code> // 16-bit Unicode</td>
<td><code>char</code> /* 8 bits */</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Floating point types</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float</td>
<td><code>float</code> // 32 bits</td>
<td><code>float</code></td>
</tr>
<tr>
<td>Double</td>
<td><code>double</code> // 64 bits</td>
<td><code>double</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logical type</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td><code>boolean</code></td>
<td>/* no equivalent */</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generic pointer type</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td><code>Object</code></td>
<td><code>void*</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constants</th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX</td>
<td><code>final int MAX = 1000;</code></td>
<td><code>define MAX 1000</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>const int MAX = 1000;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>enum {MAX = 1000};</code></td>
</tr>
</tbody>
</table>
# Java vs. C: Details

<table>
<thead>
<tr>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arrays</strong></td>
<td>int [] a = new int [10]; float [][] b = new float [5][20];</td>
</tr>
<tr>
<td>Array bound checking</td>
<td>// run-time check</td>
</tr>
<tr>
<td>Pointer type</td>
<td>// Object reference is an implicit pointer</td>
</tr>
<tr>
<td>Nullpointer checking</td>
<td>// Check for NULL, throw exception</td>
</tr>
<tr>
<td>Record type</td>
<td>class Mine { int x; float y; }</td>
</tr>
<tr>
<td></td>
<td>struct Mine { int x; float y; };</td>
</tr>
</tbody>
</table>
# Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strings</td>
<td>String s1 = &quot;Hello&quot;;</td>
<td>char *s1 = &quot;Hello&quot;;</td>
</tr>
<tr>
<td></td>
<td>String s2 = new String(&quot;hello&quot;);</td>
<td>char s2[6];</td>
</tr>
<tr>
<td></td>
<td></td>
<td>strcpy(s2, &quot;hello&quot;);</td>
</tr>
<tr>
<td>String</td>
<td>s1 + s2</td>
<td>#include &lt;string.h&gt;</td>
</tr>
<tr>
<td>concatenation</td>
<td>s1 += s2</td>
<td>strcat(s1, s2);</td>
</tr>
<tr>
<td>Logical ops</td>
<td>&amp;&amp;,</td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
<td>=, !=, &gt;, &lt;, &gt;=, &lt;=</td>
</tr>
<tr>
<td>ops *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arithmetic</td>
<td>+, -, *, /, %, unary -</td>
<td>+, -, *, /, %, unary -</td>
</tr>
<tr>
<td>ops *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bitwise</td>
<td>&gt;&gt;&gt;, &lt;&lt;, &gt;&gt;&gt;&gt;, &amp;,</td>
<td>, ^</td>
</tr>
<tr>
<td>ops</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assignment</td>
<td>=, *=, /=, +=, -=, &lt;&lt;=, &gt;&gt;=, &gt;&gt;&gt;=, &gt;&gt;&gt;&gt;&gt;, =, &amp;,, ^=,</td>
<td>=, %=</td>
</tr>
<tr>
<td>ops</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Essentially the same in the two languages*
## Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>if stmt</strong></td>
<td>if (i &lt; 0)</td>
<td>if (i &lt; 0)</td>
</tr>
<tr>
<td></td>
<td>statement1;</td>
<td>statement1;</td>
</tr>
<tr>
<td></td>
<td>else</td>
<td>else</td>
</tr>
<tr>
<td></td>
<td>statement2;</td>
<td>statement2;</td>
</tr>
<tr>
<td><strong>switch stmt</strong></td>
<td>switch (i)</td>
<td>switch (i)</td>
</tr>
<tr>
<td></td>
<td>{ case 1:</td>
<td>{ case 1:</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>break;</td>
<td>break;</td>
</tr>
<tr>
<td></td>
<td>case 2:</td>
<td>case 2:</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>break;</td>
<td>break;</td>
</tr>
<tr>
<td></td>
<td>default:</td>
<td>default:</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td><strong>goto stmt</strong></td>
<td>// no equivalent</td>
<td>goto someLabel;</td>
</tr>
</tbody>
</table>

* Essentially the same in the two languages*
Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>for stmt</td>
<td>for (int i=0; i&lt;10; i++) statement;</td>
<td>int i; for (i=0; i&lt;10; i++) statement;</td>
</tr>
<tr>
<td>while stmt</td>
<td>while (i &lt; 0) statement;</td>
<td>while (i &lt; 0) statement;</td>
</tr>
<tr>
<td>do-while stmt</td>
<td>do statement; while (i &lt; 0)</td>
<td>do statement; while (i &lt; 0);</td>
</tr>
<tr>
<td>continue stmt</td>
<td>continue;</td>
<td>continue;</td>
</tr>
<tr>
<td>labeled continue stmt</td>
<td>continue someLabel;</td>
<td>/* no equivalent */</td>
</tr>
<tr>
<td>break stmt</td>
<td>break;</td>
<td>break;</td>
</tr>
<tr>
<td>labeled break stmt</td>
<td>break someLabel;</td>
<td>/* no equivalent */</td>
</tr>
</tbody>
</table>

* Essentially the same in the two languages
## Java vs. C: Details

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>return stmt *</td>
<td><code>return 5;</code>&lt;br&gt;<code>return;</code></td>
<td><code>return 5;</code>&lt;br&gt;<code>return;</code></td>
</tr>
<tr>
<td>Compound stmt (alias block) *</td>
<td><code>{&lt;br&gt;</code>statement1;&lt;br&gt;<code>statement2;</code>&lt;br&gt;<code>}</code></td>
<td><code>{&lt;br&gt;</code>statement1;&lt;br&gt;<code>statement2;</code>&lt;br&gt;<code>}</code></td>
</tr>
<tr>
<td>Exceptions</td>
<td><code>throw, try-catch-finally</code></td>
<td><code>/* no equivalent */</code></td>
</tr>
<tr>
<td>Comments</td>
<td><code>/* comment */</code>&lt;br&gt;<code>// another kind</code></td>
<td><code>/* comment */</code></td>
</tr>
<tr>
<td>Method / function call</td>
<td><code>f(x, y, z);</code>&lt;br&gt;<code>someObject.f(x, y, z);</code>&lt;br&gt;<code>SomeClass.f(x, y, z);</code>&lt;br&gt;<code>f(x, y, z);</code></td>
<td></td>
</tr>
</tbody>
</table>

* Essentially the same in the two languages*
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    const double KMETERS_PER_MILE = 1.609;
    int miles;
    double kMeters;

    printf("miles: ");
    if (scanf("%d", &miles) != 1)
    {
        fprintf(stderr, "Error: Expected a number.\n");
        exit(EXIT_FAILURE);
    }

    kMeters = (double)miles * KMETERS_PER_MILE;
    printf("%d miles is %f kilometers.\n", miles, kMeters);
    return 0;
}
Course overview

• Introductions
• Course goals
  • Goal 1: Learn “programming in the large”
  • Goal 2: Look “under the hood” and learn low-level programming
  • Use of C and Linux supports both goals
• Resources
  • Lectures, precepts, programming environment, Piazza, textbooks
  • Course website: access via http://www.cs.princeton.edu
• Grading
• Policies
• Schedule
Summary

Getting started with C

• History of C
• Building and running C programs
• Characteristics of C
• Details of C
  • Java and C are similar
  • Knowing Java gives you a head start at learning C
Getting Started

Check out course website soon
  • Study “Policies” page
  • First assignment is available

Establish a reasonable computing environment soon
  • Instructions given in first precept