CS 2303
Systems Programming
Bob Kinicki
C08
Introduction
Introduction

- Survey and TA/SA Introductions
- Pause to Look Backwards and Forwards
- Course Objectives
- Course Operation/Expectations
- Course Plan and Fluid Syllabus
- Systems Concepts
- ‘Old’ Development Environment
  - C and C++
- Higher Level Language History
Look Backwards and Forwards

- Computers
- WPI CS Curriculum
- Instructor
- Students
  - Expected Background
  - Going Forward
  - Your Future
CS2303 Course Objectives

- To expose students to some of the lower level systems interface 'grunge' only clearly visible via C.
- To learn to program in C++ by learning to program in C first.
- To further develop the ability to design programs with emphasis on the abstract view of data structures.
- To get experience with the low-level implementation of data structures in C.
CS2303 Course Objectives

- To learn the advantages of programming in an object-oriented language such as C++.
- To experience programming in the Large

Systems Programming: Introduction 6
Pointers!!
The course web page is an important asset.

- Required Labs
- 5 Programming Assignments
- 2 Exams
To cover the details of C briskly.

- Assume an understanding of iteration and conditional constructs.
- To use only C I/O {grunge as promised!} at first

To introduce data structures in C by doing at least one program with ‘structs’ and call by value.

To finish up with as much C++ as possible.

{Note - reading of the textbook will require jumping around during the C portion of the course.}
The goal in this programming course is to try to expose the students to places where the software and hardware meet or where the application interfaces with the operating system (OS).

A systems viewpoint includes resource management (CPU and memory), process scheduling, concurrency and performance.

{Note – this is too much for this instance of the course!}
The assignments include simulation and a little computer networking that introduce two system performance concerns - **efficiency** and **fairness**.

The other important approach to appreciate is the computer scientist technique of using **abstraction** to insulate outer interfaces from ‘under-the-hood’ details (e.g., virtual memory and loaders).
Virtual Memory
C Program Development Environment

Standard Steps

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute

© 2007 Pearson Ed -All rights reserved.

Fig. 1.1 | Typical C development environment.
## Higher Level Programming Languages History

<table>
<thead>
<tr>
<th>Language</th>
<th>Year(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran</td>
<td>1957</td>
</tr>
<tr>
<td>Algol</td>
<td>1960, 1968</td>
</tr>
<tr>
<td>PL1</td>
<td>1964</td>
</tr>
<tr>
<td>Pascal</td>
<td>1970</td>
</tr>
<tr>
<td>C</td>
<td>1972</td>
</tr>
<tr>
<td>Basic</td>
<td>1975</td>
</tr>
<tr>
<td>C++</td>
<td>1986</td>
</tr>
<tr>
<td>Java</td>
<td>1995</td>
</tr>
<tr>
<td>COBOL</td>
<td>1959</td>
</tr>
<tr>
<td>Lisp</td>
<td>1959</td>
</tr>
<tr>
<td>APL</td>
<td>1962</td>
</tr>
<tr>
<td>SNOBOL</td>
<td>1967</td>
</tr>
<tr>
<td>Prolog</td>
<td>1972</td>
</tr>
<tr>
<td>Scheme</td>
<td>1975</td>
</tr>
<tr>
<td>ADA</td>
<td>1983</td>
</tr>
</tbody>
</table>