CSC 357 Lecture Notes Week 1

Introduction to the Course
Introduction to C and UNIX
I. Relevant reading.

A. K&R chapters 1 - 4; chapter 7, excluding 7.3, 7.4

B. Selected parts of Stevens and selected man pages, as cited in writeups.
II. Go over syllabus.
II. Go over syllabus.
   - Instructor
II. Go over syllabus.
   • Instructor
   • Course Objectives
II. Go over syllabus.
   • Instructor
   • Course Objectives
   • Prerequisites
II. Go over syllabus.
   • Instructor
   • Course Objectives
   • Prerequisites
   • Text and Other Class Materials
II. Go over syllabus.
   • Instructor
   • Course Objectives
   • Prerequisites
   • Text and Other Class Materials
   • Graded Work
II. Go over syllabus.
   • Instructor
   • Course Objectives
   • Prerequisites
   • Text and Other Class Materials
   • Graded Work
   • Where and When to Turn in Assignments
II. Go over syllabus.
  • Instructor
  • Course Objectives
  • Prerequisites
  • Text and Other Class Materials
  • Graded Work
  • Where and When to Turn in Assignments
  • Late Policy
II. Go over syllabus.
   • Instructor
   • Course Objectives
   • Prerequisites
   • Text and Other Class Materials
   • Graded Work
   • Where and When to Turn in Assignments
   • Late Policy
   • Collaboration and Cheating
II. Go over syllabus.

- Instructor
- Course Objectives
- Prerequisites
- Text and Other Class Materials
- Graded Work
- Where and When to Turn in Assignments
- Late Policy
- Collaboration and Cheating
- Computer Accounts
II. Go over syllabus.
   • Instructor
   • Course Objectives
   • Prerequisites
   • Text and Other Class Materials
   • Graded Work
   • Where and When to Turn in Assignments
   • Late Policy
   • Collaboration and Cheating
   • Computer Accounts
   • Course Topic Due-Date Schedule
III. Go over Lab 1.
III. Go over Lab 1.

• Tasks
III. Go over Lab 1.
   • Tasks
   • Deliverables
III. Go over Lab 1.
   • Tasks
   • Deliverables
   • Scoring Details
III. Go over Lab 1.

- Tasks
- Deliverables
- Scoring Details
- Collaboration Allowed
III. Go over Lab 1.

- Tasks
- Deliverables
- Scoring Details
- Collaboration Allowed
- How to Submit the Deliverables
IV. Go over Programming Assignment 1.
IV. Go over Programming Assignment 1.
   • Specification
IV. Go over Programming Assignment 1.

- Specification
- Sample Inputs and Outputs
IV. Go over Programming Assignment 1.

- Specification
- Sample Inputs and Outputs
- Implementation Suggestions
IV. Go over Programming Assignment 1.
- Specification
- Sample Inputs and Outputs
- Implementation Suggestions
- Deliverables
IV. Go over Programming Assignment 1.

- Specification
- Sample Inputs and Outputs
- Implementation Suggestions
- Deliverables
- Scoring Details
IV. Go over Programming Assignment 1.

• Specification
• Sample Inputs and Outputs
• Implementation Suggestions
• Deliverables
• Scoring Details
• Collaboration
IV. Go over Programming Assignment 1.
   • Specification
   • Sample Inputs and Outputs
   • Implementation Suggestions
   • Deliverables
   • Scoring Details
   • Collaboration
   • How to Submit the Deliverable
IV. Go over Programming Assignment 1.
   • Specification
   • Sample Inputs and Outputs
   • Implementation Suggestions
   • Deliverables
   • Scoring Details
   • Collaboration
   • How to Submit the Deliverable
   • Test Plan
V. Go over "UNIX Basics" handout.

- Bare-essential UNIX commands.
- Chapter 1 of the Stevens has some discussion.
- Links to additional UNIX resources in the 357 documentation dir.
VI. Go over Coding Conventions.
VI. Go over Coding Conventions.
   • Program Organization and Style
VI. Go over Coding Conventions.

- Program Organization and Style
- Functionality
VI. Go over Coding Conventions.
   • Program Organization and Style
   • Functionality
   • Compilation
VI. Go over Coding Conventions.
   • Program Organization and Style
   • Functionality
   • Compilation
   • Documentation
VI. Go over Coding Conventions.

- Program Organization and Style
- Functionality
- Compilation
- Documentation
- Scoring
VII. A brief history of C.

A. First edition in 1978; known as "K&R".

B. In 1983, work began to standardize under ANSI.

   • In 39th printing.
   • Typeset using \texttt{troff}??
Brief history of C, cont’d

D. Since its inception, C has been used to write gazillions of lines of code, at the systems level, as well as for higher-level application programs.
VIII. Brief history and variants of UNIX.

A. Bell Labs and AT&T (late 1960s).

B. BSD (early 1980s to present).

C. SUN OS and Solaris (mid 1980s to present).

D. Various other vendors (1980s to present).

E. CMU’s Mach, other universities (mid 1980s).
Brief history of UNIX., cont’d

F. Apple, A/UX and Mac OS X (1980s and present).

G. Linux (early 1990s to present).

H. CYGWIN (mid 1990s to present).

I. POSIX (late 1980s to present).
IX. Key features (and non-features) of C.

A. A low-level language.

B. A simple language.

C. No classes or other OO constructs, and no garbage collection.
X. Key similarities and differences between Java and C.

A. Most of you have used Java.

B. Fundamental constructs very similar.
   1. if-else and switch.
   2. for and while loops.
   3. function (aka method) decls and invocations.
between Java and C, cont’d

C. Features in chs 1-4 of K&R very similar to Java.

D. Functions in C, compared to methods in Java.
   1. "Function" and "method" largely synonymous.
   2. C functions defined as a top-level constructs.
between Java and C, cont’d

3. C and Java pass arguments the same; both use *call-by-value* semantics.

between Java and C, cont’d

E. Key differences between Java and C are:

1. C has no classes, no exceptions, no language-coupled GUI library.

2. C uses explicit pointers, compared to the more implicit references of Java.

3. C has features that provide low-level memory access that are missing from Java.
XI. Introductory C example.

```c
#include <stdio.h>

main() {
    printf("hello, world\n");
}
```
Introductory C example, cont’d

A. Compile using
   gcc hello.c

B. Run using
   a.out
XII. Comparison of Java and C "Hello world".

```java
import java.io.*;

public class hello {
    public static void main(String[] args) {
        System.out.println("hello, world");
    }
}
```
Comparison of Java and C, cont’d

A. Compile with
   javac hello.java

B. Run with
   java hello
XIII. **Standard I/O in C (K&R Chs 1 and 7)**

A. Streams *stdin* and *stdout* are basic.

B. Formated output is with `printf`.

1. Numerous examples throughout K&R.
2. Section 1.2 has good intro.
3. Section 7.4 has details
I/O, cont’d

C. Character i/o with `getchar` and `putchar`.
   1. We’ll use these regularly in 357.
   2. Section 1.5 has details and examples.
I/O, cont’d

D. Formated input with scanf.

1. We won’t use scanf a lot 357.

2. Various examples in K&R.

3. Section 7.4 has details.
XIV. Strings as char arrays in C.

A. In C, strings are arrays of characters.

B. Next week we’ll cover full details on C arrays; for starters, ...
Strings as char arrays, cont’d

1. Declare string variable,

   char string_var[100];

2. Assign values char-by-char, e.g.,

   string_var[0] = 'a';
   string_var[1] = 'b';
   string_var[2] = 'c';
   string_var[3] = '\0';
Strings as char arrays, cont’d

a. Strings must be *null-terminated*.

b. C library functions deal with null-terminated strings.
Strings as char arrays, cont’d

3. C functions that take strings as arguments,

```c
void f(char string_arg[]) { ... }
```

a. Empty brackets mean the `string_arg` can accept string values of any size.
Strings as char arrays, cont’d

b. Equivalent notation,

    void f(char *string_arg) { /*...*/}
Strings as char arrays, cont’d

4. Double-quoted string constants, e.g.,

```c
f("xyz");
```
Strings as char arrays, cont’d

5. Initialize string-constant variable:

```
char* string_const = "xyz";
```

a. Can’t assign to such a string constant.

b. We’ll discuss why next week.
Strings as char arrays, cont’d

6. Section 1.9 of K&R has useful example.

7. We’ll cover the full details next week.
XV. C Types, operators, expressions (K&R Ch 2).

A. Primitive data types.

<table>
<thead>
<tr>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>a single byte, capable of holding one character in the local character set</td>
</tr>
</tbody>
</table>
## Primitive data types, cont’d

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>an integer, typically reflecting the natural size of integers on the host machine</td>
</tr>
<tr>
<td>float</td>
<td>single-precision floating point</td>
</tr>
<tr>
<td>double</td>
<td>double-precision floating point</td>
</tr>
</tbody>
</table>
Primitive data types, cont’d

with qualifiers short, long, signed, and unsigned.
Types, operators, and expressions, cont’d

B. Constant declarations.

1. Use ‘#define’.

2. E.g.,

   #define LINE_LENGTH 72

3. Also the enum declaration.
Types, operators, and expressions, cont’d

C. Arith and rel operators.

1. Very much the same as Java.

2. See the cited K&R sections for details.
Types, operators, and expressions, cont’d

D. Type conversions.

1. C does fewer auto conversions than Java.
2. Many things convert to int.
3. Details in upcoming lectures.
XVI. C Control flow (K&R Ch 3).

A. Very much the same as Java.

B. Notably missing from Java is `goto`, which you should not use in C.
XVII. Functions, program structure (K&R Ch 4).

A. Function declaration and invocation in C is much like methods in Java.

B. A C program is a collection of .c files, exactly one of which contains a main function.
Functions and program structure, cont’d

C. C programs typically have .h files.

1. Pair of .c and .h files are rough equivalent of Java class.

2. We’ll discuss further in upcoming lectures.
XVIII. File access (K&R Ch 7).

A. Files opened with `fopen` and `fclose`.

B. `getc` and `putc` perform character i/o.

C. `fprintf` and `fscanf` do formatted i/o.

D. `fgets` and `fputs` do whole-line i/o.

E. Sections 7.5 and 7.7 have details.
XIX. Key diffs between UNIX and Windows.

A. UNIX users spend more time in terminal shells.

B. With the latest advances, UNIX user experience much like Windows.

C. In 357, we’ll be dealing with UNIX primarily at the system level.
Key diffs between UNIX and Windows., cont’d

1. We’ll not spend much time in higher-level apps.

2. Universal interface to compilation and execution using command-line interface.

3. You are free to use a higher-level IDE.
XX. Basic C tools for CSC 357.

A. The GNU C compiler -- gcc.

B. The GNU C debugger -- gdb.
XXI. C development environments.

A. Plain text editor, UNIX terminal, basic tools.

B. Emacs and the basic tools (Fisher’s choice).

C. Open-source IDEs:
C development environments, cont’d

1. Eclipse CDT
2. Gnu DDD
3. jGrasp

D. A slew of commercial IDEs.
XXII. IMPORTANT NOTE about C execution environment for CSC 357.

A. All programs must run correctly on falcon/hornet.

B. It’s fine to use an IDE, but you must confirm that program compiles and executes properly on falcon/hornet.