COS 217: Introduction to Programming Systems

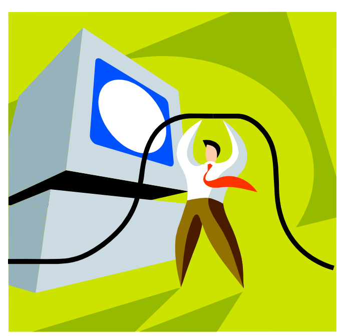
Professor Jennifer Rexford

http://www.cs.princeton.edu/~jrex

1

Goals for Today's Class

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



- Getting started with C
  - C programming language overview

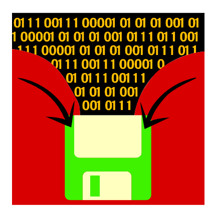
2

Introductions

- Jennifer Rexford (professor)
  - Room 306 in Computer Science Building
  - jrex@cs.princeton.edu
- Robert Dondero (lead preceptor)
  - Room 206 in Computer Science Building
  - rdondero@cs.princeton.edu
- Jailu Huang (preceptor)
  - Room 213 in Computer Science Building
  - jailuh@cs.princeton.edu
- Donna O'Leary (administrator)
  - Room 210 in Computer Science Building
  - doleary@cs.princeton.edu

3

Course Goal 1: "Programming in the Large"



- Goal 1: "Programming in the large"
  - Help you learn how to write large computer programs
- Specifically, help you learn how to:
  - Write modular code
    - Hide information
    - Manage resources
    - Handle errors
  - Write portable code
  - Test and debug your code
  - Improve your code's performance (and when to do so)
  - Use tools to support those activities

4

Course Goal 2: “Under the Hood”

- Goal 2: “Look under the hood”
  - Help you learn what happens “under the hood” of computer systems
- Specifically, two downward tours
  - Language levels tour
    - High-level language (C) → assembly language (AT&T) → machine language (IA-32)
  - Service levels tour
    - High-level language (C) → standard libraries → operating system (Linux)
- Goal 2 supports Goal 1
  - Reveals many examples of effective abstractions

5

Course Goals: Why C?

- Q: Why C instead of Java?
- A: C supports Goal 1 better
  - C is a lower-level language
    - C provides more opportunities to create abstractions
  - C has some flaws
    - C’s flaws motivate discussions of software engineering principles
- A: C supports Goal 2 better
  - C facilitates language levels tour
    - C is closely related to assembly language
  - C facilitates service levels tour
    - Linux is written in C

6

Course Goals: Why Linux?

- Q: Why Linux instead of Microsoft Windows?
- A: Linux is good for education and research
  - Linux is open-source and well-specified
- A: Linux is good for programming
  - Linux is a variant of Unix
  - Unix has GNU, a rich open-source programming environment

7

Course Goals: Summary

- Help you to become a...

Power Programmer!!!

8

Resources: Lectures and Precepts

- Lectures
  - Describe concepts at a high level
  - Slides available online at course Web site
- Precepts
  - Support lectures by describing concepts at a lower level
  - Support your work on assignments

Resources: Website and Listserv

- Website
  - Access from http://www.cs.princeton.edu
    - Course Information → COS 217
- Listserv
  - cos217@lists.cs.princeton.edu
  - Subscription is required
  - Instructions provided in first precept

Resources: Books

- Required book
  - *C Programming: A Modern Approach (Second Edition),* King, 2008.
    - Covers the C programming language and standard libraries
    - First edition is not quite as good, but is sufficient
- Highly recommended books
  - *The Practice of Programming,* Kernighan and Pike, 1999.
    - Covers “programming in the large”
    - (Required for COS 333)
  - *Computer Systems: A Programmer's Perspective*, Bryant and O'Hallaron, 2003.
    - Covers “under the hood”
    - Some key sections will be on electronic reserve
  - *Programming with GNU Software*, Loukides and Oram, 1997.
    - Covers tools
- *All books are on reserve in Engineering Library*

11

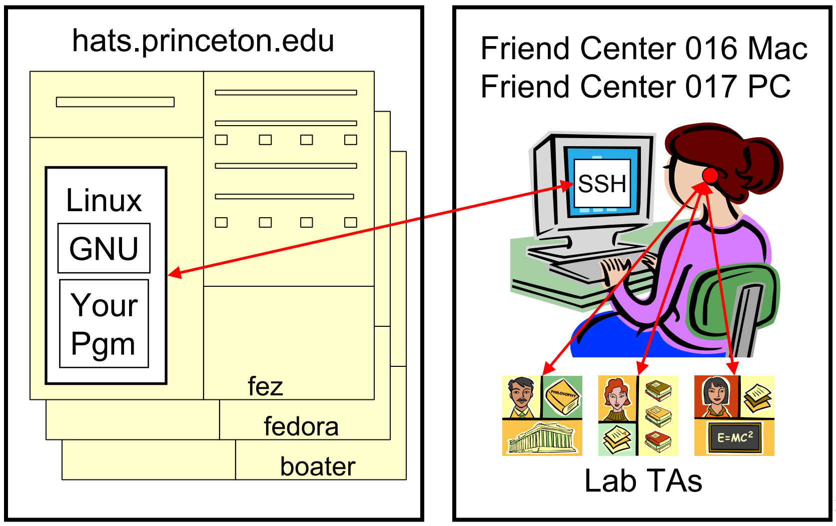
Resources: Manuals

- Manuals (for reference only, available online)
  - *IA32 Intel Architecture Software Developer's Manual, Volumes 1-3*
  - *Tool Interface Standard & Executable and Linking Format*
  - *Using as, the GNU Assembler*
- See also
  - Linux “man” command
  - man – short for manual
    - For more help, type “man man”

12

Resources: Programming Environment

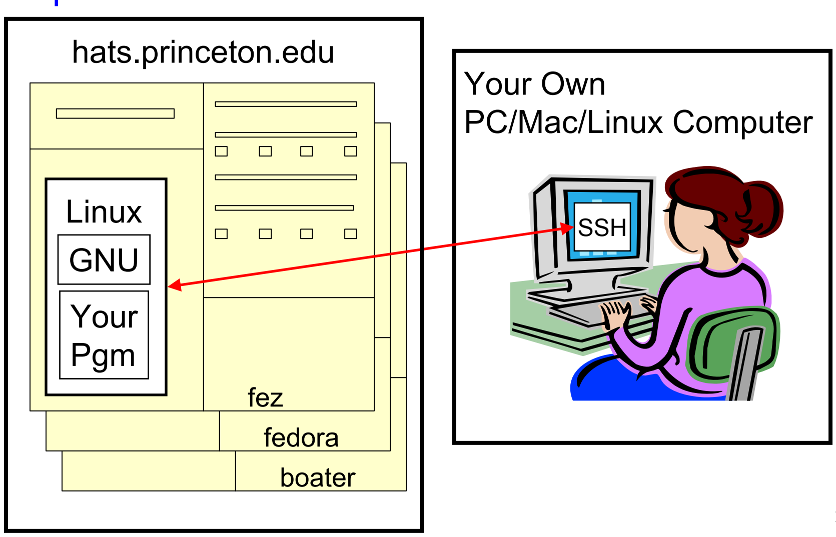
- Option 1



13

Resources: Programming Environment

- Option 2



14

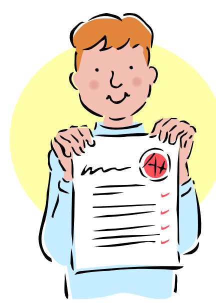
Resources: Programming Environment

- Other options
  - Use your own PC/Mac/Linux computer; run GNU tools locally; run your programs locally
  - Use your own PC/Mac/Linux computer; run a non-GNU development environment locally; run your programs locally
  - Etc.
- Notes
  - Other options cannot be used for some assignments (esp. timing studies)
  - Instructors cannot promise support of other options
  - Strong recommendation: Use Option 1 or 2 for all assignments
  - First precept provides detailed setup instructions

15

Grading

- Seven programming assignments (50%)
  - Working code
  - Clean, readable, maintainable code
  - On time (penalties for late submission)
  - Final assignment counts double (12.5%)
- Exams (40%)
  - Midterm (15%)
  - Final (25%)
- Class participation (10%)
  - Lecture and precept attendance is ***mandatory***



16

Programming Assignments

- Programming assignments
  1. A “decomment” program
  2. A string module
  3. A symbol table module
  4. A heap manager module
  5. IA-32 assembly language programs
  6. A buffer overrun attack
  7. A UNIX shell
- Key aspect of the course
- Due (typically) Sundays at 9:00PM
- First assignment is available now
- Advice: Start early to allow time for debugging (especially in the background while you are doing other things!)…

17

Why Debugging is Necessary…

18

Policies

- From course Policies web page:

  "Programming in an individual creative process much like composition. You must reach your own understanding of the problem and discover a path to its solution. During this time, discussions with friends are encouraged. However, when the time comes to write code that solves the problem, such discussions are no longer appropriate - the program must be your own work. If you have a question about how to use some feature of C, UNIX, etc., you can certainly ask your friends or the teaching assistants, but **do not, under any circumstances, copy another person's program.** Letting someone copy your program or using someone else's code in any form is a **violation of academic regulations**. "Using someone else's code" includes using solutions or partial solutions to assignments provided by commercial web sites, instructors, preceptors, teaching assistants, friends, or students from any previous offering of this course or any other course."

19

Course Schedule

- Very generally…

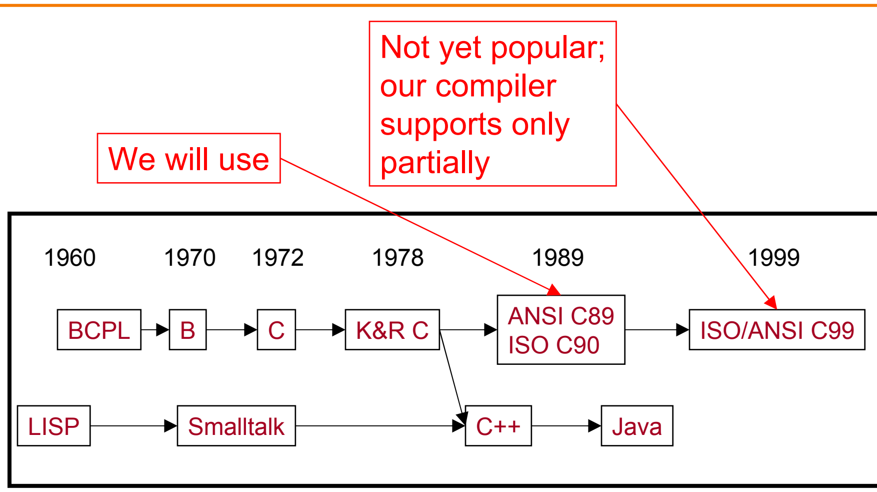
Weeks	Lectures	Precepts
1-2	Intro to C	Intro to Linux/GNU
3-6	"Pgmming in the large"	Advanced C
6	Midterm exam	
7	Recess	
8-13	"Under the hood"	Assembly language
	Reading period	
	Final exam	

- See course "Schedule" web page for details

20

Any questions before we start?

C vs. Java: History



C vs. Java: Design Goals

- Java design goals
  - Support object-oriented programming
  - Allow same program to be executed on multiple operating systems
  - Support using computer networks
  - Execute code from remote sources securely
  - Adopt the good parts of other languages (esp. C and C++)
- Implications for Java
  - Good for application-level programming
  - High-level
    - Virtual machine insulates programmer from underlying assembly language, machine language, hardware
  - Portability over efficiency
  - Security over efficiency
  - Security over flexibility

23

C vs. Java: Design Goals

- C design goals
  - Support structured programming
  - Support development of the Unix OS and Unix tools
    - As Unix became popular, so did C
- Implications for C
  - Good for system-level programming
    - But often used for application-level programming – sometimes inappropriately
  - Low-level
    - Close to assembly language; close to machine language; close to hardware
  - Efficiency over portability
  - Efficiency over security
  - Flexibility over security

24

C vs. Java: Design Goals

- Differences in design goals explain many differences between the languages
- C's design goal explains many of its eccentricities
  - We'll see examples throughout the course

25

C vs. Java: Overview

- Dennis Ritchie on the nature of C:
  - "C has always been a language that never attempts to tie a programmer down."
  - "C has always appealed to systems programmers who like the terse, concise manner in which powerful expressions can be coded."
  - "C allowed programmers to (while sacrificing portability) have direct access to many machine-level features that would otherwise require the use of assembly language."
  - "C is quirky, flawed, and an enormous success. While accidents of history surely helped, it evidently satisfied a need for a system implementation language efficient enough to displace assembly language, yet sufficiently abstract and fluent to describe algorithms and interactions in a wide variety of environments."

26

C vs. Java: Overview (cont.)

- Bad things you **can** do in C that you **can't** do in Java
  - Shoot yourself in the foot (safety)
  - Shoot others in the foot (security)
  - Ignore wounds (error handling)
- Dangerous things you **must** do in C that you **don't** in Java
  - Explicitly manage memory via `malloc()` and `free()`
- Good things you **can** do in C, but (more or less) **must** do in Java
  - Program using the objected-oriented style
- Good things that you **can't** do in C but **can** do in Java
  - Write completely portable code

27

C vs. Java: Details

- Remaining slides provide some details
  - Suggestion: Use for future reference
- Slides covered briefly now (as time allows)…

28

C vs. Java: Details (cont.)

	Java	C

Overall Program Structure	`Hello.java:`
	`public class Hello {`
	`  public static void`
	`    main(String[] args) {`
	`      System.out.println(`
	`        "Hello, world");`
	`  }`
	`}`
		`hello.c:`
		`#include <stdio.h>`
		`int main(void) {`
		`  printf("Hello, world\n");`
		`  return 0;`
		`}`

Building	`% javac Hello.java`
	`% ls`
	`Hello.class`
	`Hello.java`
	`%`
		`% gcc217 hello.c`
		`% ls`
		`a.out`
		`hello.c`
		`%`

Running	`% java Hello`
	`Hello, world`
	`%`
		`% a.out`
		`Hello, world`
		`%`

29

C vs. Java: Details (cont.)

	Java	C
Character type	`char // 16-bit unicode`	`char /* 8 bits */`

Integral types	`byte // 8 bits`
	`short // 16 bits`
	`int // 32 bits`
	`long // 64 bits`
		`(unsigned) char`
		`(unsigned) short`
		`(unsigned) int`
		`(unsigned) long`

Floating point types	`float // 32 bits`
	`double // 64 bits`
		`float`
		`double`
		`long double`

Logical type	`boolean`	`/* no equivalent */` `/* use integral type */`
Generic pointer type	`// no equivalent`	`void*`

Constants	`final int MAX = 1000;`	`#define MAX 1000`
		`const int MAX = 1000;`
		`enum {MAX = 1000};`

30

C vs. Java: Details (cont.)

	Java	C

Arrays	`int [] a = new int [10];`
	`float [][] b =`
	`    new float [5][20];`
		`int a[10];`
		`float b[5][20];`

Array bound checking	`// run-time check`	`/* no run-time check */`
Pointer type	`// Object reference is an` `// implicit pointer`	`int *p;`

Record type	`class Mine {`
	`    int x;`
	`    float y;`
	`}`
		`struct Mine {`
		`    int x;`
		`    float y;`
		`}`

31

C vs. Java: Details (cont.)

	Java	C

Strings	`String s1 = "Hello";`
	`String s2 = new`
	`    String("hello");`
		`char *s1 = "Hello";`
		`char s2[6];`
		`strcpy(s2, "hello");`

String concatenation	`s1 + s2` `s1 += s2`	`#include <string.h>` `strcat(s1, s2);`
Logical ops	`&&, ||, !`	`&&, ||, !`
Relational ops	`=, !=, >, <, >=, <=`	`=, !=, >, <, >=, <=`
Arithmetic ops	`+, -, *, /, %, unary -`	`+, -, *, /, %, unary -`
Bitwise ops	`>>, <<, >>>, &, |, ^`	`>>, <<, &, |, ^`
Assignment ops	`=, *=, /=, +=, -=, <<=,` `>>=, >>>=, =, ^=, |=, %=`	`=, *=, /=, +=, -=, <<=,` `>>=, =, ^=, |=, %=`

32

C vs. Java: Details (cont.)

	Java	C

if stmt	**if** (i < 0)
	    statement1;
	else
	    statement2;
		if (i < 0)
		    statement1;
		else
		    statement2;

switch stmt	**switch** (i) {
	  case 1:
	    ...
	    break;
	  case 2:
	    ...
	    break;
	  default:
	    ...
	}
		switch (i) {
		  case 1:
		    ...
		    break;
		  case 2:
		    ...
		    break;
		  default:
		    ...
		}

goto stmt	// no equivalent	**goto** SomeLabel;

33

C vs. Java: Details (cont.)

	Java	C

for stmt	**for** (int i=0; i<10; i++)
	    statement;
		int i;
		for (i=0; i<10; i++)
		    statement;

while stmt	**while** (i < 0)
	    statement;
		while (i < 0)
		    statement;

do-while stmt	**do** {
	    statement;
	    …
	} **while** (i < 0)
		do {
		    statement;
		    …
		} **while** (i < 0)

continue stmt	**continue**;	**continue**;
labeled continue stmt	**continue** SomeLabel;	/* no equivalent */
break stmt	**break**;	**break**;
labeled break stmt	**break** SomeLabel;	/* no equivalent */

4

C vs. Java: Details (cont.)

	Java	C
return stmt	`return 5;` `return;`	`return 5;` `return;`

Compound stmt (alias block)	`{`
	`    statement1;`
	`    statement2;`
	`}`
		`{`
		`    statement1;`
		`    statement2;`
		`}`

Exceptions	`throw, try-catch-finally`	`/* no equivalent */`
Comments	`/* comment */` `// another kind`	`/* comment */`

Method / function call	`f(x, y, z);`
	`someObject.f(x, y, z);`
	`SomeClass.f(x, y, z);`
		`f(x, y, z);`

35

Example C Program

```
#include <stdio.h>
#include <stdlib.h>

const double KMETERS_PER_MILE = 1.609;

int main(void) {
   int miles;
   double kmeters;
   printf("miles: ");
   if (scanf("%d", &miles) != 1) {
      fprintf(stderr, "Error: Expect a number.\n");
      exit(EXIT_FAILURE);
   }
   kmeters = miles * KMETERS_PER_MILE;
   printf("%d miles is %f kilometers.\n",
      miles, kmeters);
   return 0;
}
```

36

Summary

- Course overview
  - Goals
    - Goal 1: Learn "programming in the large"
    - Goal 2: Look "under the hood"
    - Goal 2 supports Goal 1
    - Use of C and Linux supports both goals
  - Learning resources
    - Lectures, precepts, programming environment, course listserv, textbooks
    - Course Web site: access via http://www.cs.princeton.edu

37

Summary

- Getting started with C
  - C was designed for system programming
    - Differences in design goals of Java and C explain many differences between the languages
    - Knowing C design goals explains many of its eccentricities
  - Knowing Java gives you a head start at learning C
    - C is not object-oriented, but many aspects are similar

38

Getting Started

- Check out course Web site **soon**
  - First assignment is available
- Establish a reasonable computing environment **soon**
  - Instructions given in first precept
- Reading
  - Required: *C Programming* (King) 1, 2, 3
  - Required: *Computer Systems* (Bryant & O'Hallaron) 1
  - Recommended: *Programming with GNU Software* (Loukides & Oram) 1, 2, 4

39