CSE451 Fall 2008
Section 1

Roxana Geambasu
(roxana@cs)

About Me

- Fourth year Ph.D. student
- Research in:
  - Large-scale storage systems (Hank, Steve)
  - Security (Yoshi, Hank)
  - Personal data management (Magda, Hank, Steve)
- First time teaching sections in the U.S.
  - So, I’ll be learning along with you ☺

Reminders

- Sign-up for the mailing list
- Start reading the book
  - Homework 1 due on Monday
- Read and start Project 0 (due next Wed.)
- Make sure you can access forkbomb.cs.washigtnon.edu after Friday
  - If not, email support@cs or me

Office Hours

- Kristin: Tuesday 2-3pm and Wednesday 4-5pm

- Nick: Monday and Wednesday 12 – 1pm

- Roxana: Wednesday 9-10am and Friday (10:30--11:30am)

451 Projects

- 5 interesting but demanding projects (mostly C):
  - Practice C
  - Shell & process control
  - User-level threads
  - Virtual memory
  - File systems
- First one: work individually
- The other four: work in groups
- Likely to be similar to projects in the past
- Start early on each project!

Project Rules

- Collaboration ok (except for the first project)
  - Let us know with whom you collaborate
- Copying is not ok
- Use only ‘forkbomb.cs.washington.edu’!
  - Debug/run your programs in a sandbox (see forkbomb info link on the web site)
  - Do NOT use attu for projects

Project Grading

- What do you think we grade about your code?

Project Grading

- Correctness: algorithm (protocol), implementation
- Code structure and clarity
- Comments
- Memory management (for C, beware of mem. leaks and buffer overflows)
- Error handling (file ops, mem. allocation, all system calls)
- Input handling (unless specified otherwise)
- No warnings (compile with gcc -Wall)
- Performance (complexity) only when specified

Project 0
C programming warm-up

Due date: Oct. 1, 11:59pm

Project 0

- Part 1: Debug and extend a queue implementation
- Part 2: Implement a hash table

- Goal of project 0:
  - Dust up your knowledge of C, UNIX tools (303) and data structures (326)
  - Prepare you for next projects (e.g., function pointers)

C & UNIX Tools Background

- How many of you have:
  - written a C program?
  - seen a Makefile?
  - used gcc?
  - used gdb?

Remember from Previous Classes?

What are those and when are they used:

- Pointers and pointer arithmetic
- Static vs. dynamic memory allocation
- Call-by-value vs. call-by-reference
- Structures, typedef

- Good reminder and resources at:

http://www.cs.washington.edu/education/courses/451/07au/section/rec1.htm

Common C Pitfalls (1)

- What’s wrong and how to fix it?

```
char* get_city_name(double latitude,
                    double longitude) {
  char city_name[100];
  …
  return city_name;
}
```

Common C Pitfalls (1)

- Problem: return pointer to statically allocated mem.
- Solution: allocate on heap

```
char* get_city_name(double latitude,
                    double longitude) {
  char* city_name = (char*)malloc(100);
  …
  return city_name;
}
```

- Slightly more subtle example:

```
typedef struct _city_info_t {
     char* name;
     … …
} city_info_t;

city_info get_city_name(double latitude,
     double longitude) {
  city_info_ city_info;
  char city_name[100];
  … …
  city_info.name = city_name;
  return city_info;
}
```

Common C Pitfalls (2)

- What’s wrong and how to fix it?

```
 char* buf = (char*)malloc(32);
strcpy(buf, argv[1]);
```

Common C Pitfalls (2)

- Problem: Buffer overflow
- Solution:

```
   int buf_size = 32;
  char* buf = (char*)malloc(buf_size);
 strncpy(buf, argv[1], buf_size);
```

- Are buffer overflow bugs important?

Common C Pitfalls (3)

- What’s wrong and how to fix it?

```
char* buf = (char*)malloc(32);
strncpy(buf, “hello”, 32);
printf(“%s\n”, buf);

buf = (char*)malloc(64);
strncpy(buf, “bye”, 64);
printf(“%s\n”, buf);

free(buf);
```

Common C Pitfalls (3)

- Problem: Memory leak
- Solution:

```
char* buf = (char*)malloc(32);
strncpy(buf, “hello”, 32);
printf(“%s\n”, buf);
free(buf);
buf = (char*)malloc(64);
…
```

- Are memory leaks important?
  - OS, web server, web browser, your projects?

Bug in all previous examples

- We didn’t handle memory allocation failures:

```
char *buf = (char*)malloc(32);
if (buf == NULL) return;
```

- You should do that in your code

Debugging C

Poll – What do you use to debug?

- Printf, printf, printf
- Stare at code
- Ask friends
- GDB

GDB

- Most frequent GDB commands
  - Execution: ***run, continue*** *(c)**, **step*** *(s)**, **next*** *(n)*
  - Break point: ***break, clear, condition***
  - Browsing the stack: ***up, down, backtrace***
  - Investigate data: ***display, print***
  - Browsing source: ***list***
- Compile with **'-g'** to have access to symbol table and line numbers.
- More info at:
  - http://sourceware.org/gdb
  - http://www.cs.washington.edu/education/cou