Recitation Nov 21

Malloclab summary, testing,
tools for proxylab

Topics

- Summary of malloclab
- The importance of testing
- Tools for proxylab
  - telnet
  - netcat
  - diff
  - thttp
  - tiny
  - wireshark

Malloclab

- Questions, comments about malloclab?
- Lessons learned?
- Many students tried to implement first, then debug
  - Start with simplest implementation
  - Make small changes, and test every change
  - If something breaks, you know what caused it
- Think about this process in design
  - E.g. "how can I design my explicit list so that I can easily transition to seglists?"

Topics

- Summary of malloclab
- The importance of testing
- Tools for proxylab
  - telnet
  - netcat
  - diff
  - thttp
  - tiny
  - wireshark

Testing in proxylab

- **We provide no tests for proxylab**
  - Designing tests is a critical part of designing a program
- Two major types of tests
  - Unit tests - test small parts of program
  - System tests - test operation of whole program
- Both types of tests are important

- A good test suite can make or break a real program
- Sometimes the bulk of the code is tests!

"SQLite is a test suite that ships with an embedded database"

Unit tests

- Test smallest possible "units" of code
  - E.g. "does this function do what it's supposed to"
- Test interaction between small groups of functions
  - Technically *integration testing*
- Usually written with a *testing framework* library
  - Automates setup, teardown, and reporting
  - You may want to use one of these

Writing tests

1. Determine what test cases to test, and correct output
    - May be known cases, e.g. "if I input X, it should output Y"
    - May be alternative implementation
    - Can use random cases + two implementations
2. Write test function
    - Compares actual output to desired output
    - Logs results
3. Write code to run all tests automatically

Example: divpwr2 from datalab

Unit test example - bad divpwr2

```
/*
 * divpwr2 - Compute x/(2^n), for 0 <= n <= 30
 *  Round toward zero
 *   Examples: divpwr2(15,1) = 7,
 *             divpwr2(-33,4) = -2
 *   Legal ops: ! ~ & ^ | + << >>
 *   Max ops: 15
 *   Rating: 2
 */
int32_t divpwr2(int32_t x, int32_t n)
{
   return x >> n;
}
```

Unit test example - alternative implementation

```
int32_t divpwr2_easy(int32_t x, int32_t n)
{
    assert(n>=0 and n<=30);
    while (n > 0) {
x /= 2;
        n --;
    }
    return x;
}
```

Unit test example - testing function

```
void test_divpwr2_impl(int32_t x, int32_t n)
{
    int a, b;
    a = divpwr2(x, n);
    b = divpwr2_easy(x, n);
    if (a != b) {
printf("divpwr2 failed test %i %i\n",
                x, n);
printf("\treturned %i, should be %i\n",
                a, b);
    }
}
```

Unit test example - output

```
$ ./unittest
divpwr2 failed test -1 1
returned -1, should be 0
divpwr2 failed test -3 1
returned -2, should be -1
```

It's failing on odd negative numbers!

Testing notes

- All code may have bugs in it ... including the tests
  - When a test fails, it could be the program code, or the test code
- Develop tests incrementally as well
  - Start with common case, expected corner cases
    - E.g. negative numbers, 0, 1
    - NULL, "", over-length string, non-terminated string
  - When you discover a bug, make a test!
- Make sure that you can run all tests automatically
  - Simple tests can be hand-coded like example
  - For complicated tests use a testing framework

Topics

- Summary of malloclab
- The importance of testing
- Tools for proxylab
  - telnet
  - netcat
  - diff
  - thttp
  - tiny
  - wireshark

Tools for proxylab

- telnet - simple text-based network connection
- netcat (nc) - "network swiss army knife"
- diff - compare two text files
- thttpd - simple http server
- tiny - another simple http server
- wireshark/tshark - watch network traffic

Telnet

telnet <host> [port]

- Creates simple, plain-text network connections
- Anything you type is sent over wire
- Anything received is printed to screen

Telnet example - http get request

```
$ telnet www.google.com 80
Trying 72.14.204.99...
Connected to www.l.google.com.
Escape character is '^]'.
GET / HTTP/1.1

HTTP/1.1 200 OK
Date: Sun, 20 Nov 2011 21:22:49 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie:
...
```

Netcat

nc <host> <port>

- Like telnet, opens plain-text connection
- Doesn't print any "cruft"

nc -l <port> [-k]

- Starts a server (listen) on given port
- -k: keep server open after client disconnect

- Netcat has many more options, "man nc"
- There are multiple versions with slight differences
  - Consult your man page if any of this doesn't work
- Try starting server in one window, connect to it from another
  - Anything you type should be mirrored between the two

thttpd

thttpd -p <port> -D

- Serves local directory over http (many more options)
- -D keeps it as a foreground process
- Example thttpd and netcat:
  - First window:

```
$ thttpd -p 17171 -D
```

Second window:

```
$ nc localhost 17171
GET /thttpd.log HTTP/1.0

HTTP/1.0 200 OK
Server: thttpd/2.25b 29dec2003
Content-Type: text/plain; charset=iso-8859-1
...
```

tiny - Dave O's tiny http server

- Similar to thttpd
- Source is even tinier
  - May be easier to understand and modify

Wireshark, tshark, tcpdump - packet sniffing

- Gathers all data seen by ethernet
- Can be used to sanity check your program
  - Does the data "over the wire" match what you think you're sending?
- Wireshark is a GUI program
  - Command-line version is tshark
- tcpdump gets data from tcp connections
  - "man tcpdump" has example of sniffing http connections

tcpdump example

```
$ sudo tcpdump tcp port 17171 -i lo -A
...
GET /hello.txt HTTP/1.0
...
HTTP/1.0 200 OK
Server: thttpd/2.25b 29dec2003
Content-Type: text/plain; charset=iso-8859-1
Date: Mon, 21 Nov 2011 03:14:13 GMT
Last-Modified: Sun, 20 Nov 2011 21:35:39 GMT
Accept-Ranges: bytes
Connection: close
Content-Length: 13

Hello World!
```

A few extra things about proxylab

- Revision control (e.g. git)
  - Now that you're working with a partner this is important!
- Get started early on proxylab
- High-level design is a big part of this project
  - Even more so than malloclab