CS3157: Advanced Programming

Lecture #5

Oct 9

Shlomo Hershkop
shlomo@cs.columbia.edu

1

Announcements

- please make sure you are making progress on the homework
  - any questions ?
  - again, except md5, no cpan modules in general
  - try to organize your code for efficiency, TAs can help you with this on giving you feedback on your ideas (A or B faster etc)

2

Today

- wrap up Perl
  - packages and modules
  - OOP
  - Random stuff
- Please acquire a C book (Deitel & Deitel)

- reading:
  - perl object and packages

3

IMPORTANT

- if you miss something (might be glossing over some things)
  - please contact me about it
  - try to also find it in the index of your book

- Hard to learn unless you ask

4

Issues

- character encoding issues
- cgi isssues

5

Side Note: Line Endings

- Carriage return \r
- Line Feed \n
- CRLF
- Unix – LF (\n) CR (\r)
- print “Content-type: text/html\n\n”
- Why not \n\r\n\r ????

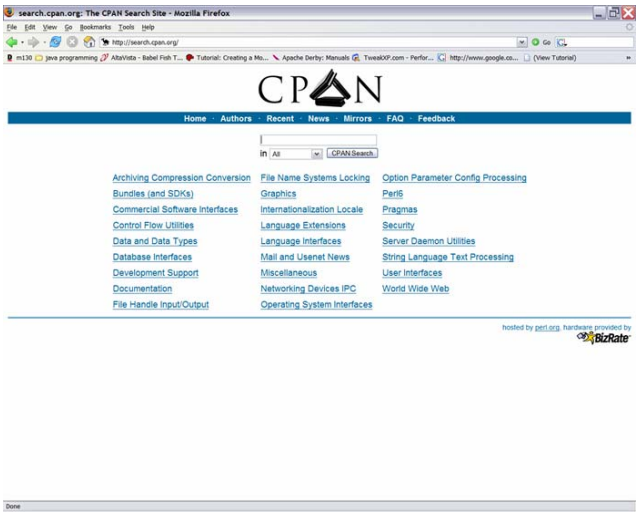
6

Updates

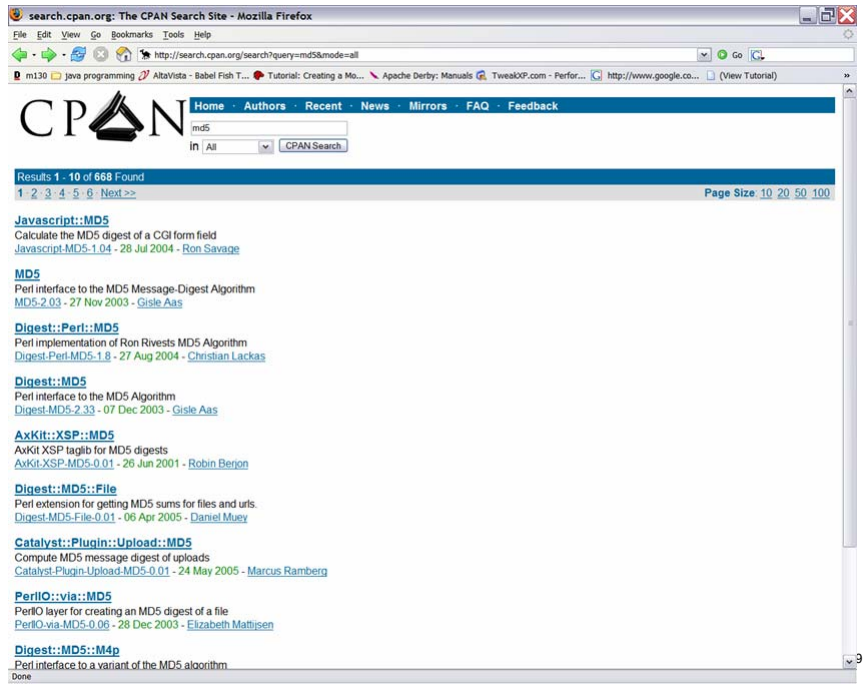
- Ranges are only in the positive direction
- [5 .. 1]
  - null list returned
  - So how to get it ?

7

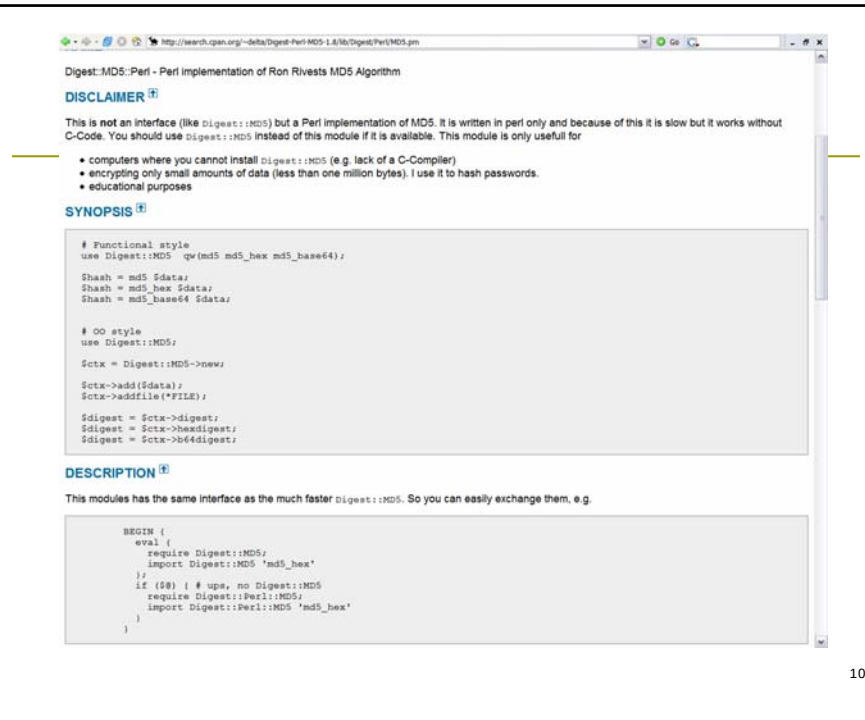
Using Perl Libraries



8


search.cpan.org: The CPAN Search Site - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://search.cpan.org/search?query=md5&mode=all
CPAN
Home · Authors · Recent · News · Mirrors · FAQ · Feedback
md5
in All
CPAN Search
Results 1 - 10 of 668 Found
1 2 3 4 5 6 Next >>
Page Size: 10 20 50 100
Javascript::MD5
Calculate the MD5 digest of a CGI form field
Javascript-MD5-1.04 - 28 Jul 2004 - Ron Savage
MD5
Perl interface to the MD5 Message-Digest Algorithm
MD5-2.03 - 27 Nov 2003 - Gisle Aas
Digest::Perl::MD5
Perl implementation of Ron Rivests MD5 Algorithm
Digest-Perl-MD5-1.8 - 27 Aug 2004 - Christian Lackas
Digest::MD5
Perl interface to the MD5 Algorithm
Digest-MD5-2.33 - 07 Dec 2003 - Gisle Aas
AxKit::XSP::MD5
AxKit XSP taglib for MD5 digests
AxKit-XSP-MD5-0.01 - 26 Jun 2001 - Robin Berjon
Digest::MD5::File
Perl extension for getting MD5 sums for files and urls.
Digest-MD5-File-0.01 - 06 Apr 2005 - Daniel Muey
Catalyst::Plugin::Upload::MD5
Compute MD5 message digest of uploads
Catalyst-Plugin-Upload-MD5-0.01 - 24 May 2005 - Marcus Ramberg
PerlIO::via::MD5
PerlIO layer for creating an MD5 digest of a file
PerlIO-via-MD5-0.06 - 28 Dec 2003 - Elizabeth Mattijsen
Digest::MD5::M4p
Perl interface to a variant of the MD5 algorithm
Done

9


Digest::MD5::Perl - Perl implementation of Ron Rivests MD5 Algorithm
DISCLAIMER
This is not an interface (like Digest::MD5) but a Perl implementation of MD5. It is written in perl only and because of this it is slow but it works without C-Code. You should use Digest::MD5 instead of this module if it is available. This module is only usefull for
computers where you cannot install Digest::MD5 (e.g. lack of a C-Compiler)
encrypting only small amounts of data (less than one million bytes). I use it to hash passwords.
educational purposes
SYNOPSIS
# Functional style
use Digest::MD5 qw(md5 md5_hex md5_base64);
$hash = md5 $data;
$hash = md5_hex $data;
$hash = md5_base64 $data;
OO style
use Digest::MD5;
$ctx = Digest::MD5->new;
$ctx->add($data);
$ctx->addfile(*FILE);
$digest = $ctx->digest;
$digest = $ctx->hexdigest;
$digest = $ctx->b64digest;
DESCRIPTION
This modules has the same interface as the much faster Digest::MD5. So you can easily exchange them, e.g.
BEGIN {
eval {
require Digest::MD5;
import Digest::MD5 'md5_hex'
};
if ($@) { # ups, no Digest::MD5
require Digest::Perl::MD5;
import Digest::Perl::MD5 'md5_hex'
}
}

10

GUI

- There are easy ways to make graphics in perl
- Will not cover in this course
  - But will have enough knowledge to pick this up on your own if you choose
  - Better way: will see later today

11

Graphics

```
#!c:\perl\bin
use Tk;

my $mwin = MainWindow->new;

$mwin->Button(-text => "Hello World!", -
  command => sub{exit})->pack;
MainLoop;
```

12

Graphics

- Good to know about
- Might need to one day debug someone else's code (GASP!)

13

Computer Security

- System and theory of ensuring the confidentiality, integrity, availability, and control of electronic information and systems.
  - Network
  - Host
  - Data

14

For host based security

- Want to ensure permission system
  - X should only be allowed to do A, B, and C
- Want to ensure accountability
  - If Y does something not allowed, should be noted
- Want to be able to track
  - If something has been tampered with, how can we locate it
  - Both preventative and reactionary

15

Homework Project

- Assuming you are a system administrator or just paranoid
- Take chronological snapshots of your system to compare and find changes
  - Many changes by system
  - Many changes by valid user
  - Might locate malicious user/system changes
- Want to search filenames
- Want to organize snapshots of system

16

References

- Reminder: A reference is a way of talking about a variable
- Symbolic Reference: a reference which we need to look up what it means in some other place
- Hard Reference: direct value of other object

17

Creating References

- Backslash (covered in the past)
- `$foo = 200;`
- `$ref_foo = \$foo;`
- `$constref = \312;`
- `$sub_ref = \&somesub;`

18

References to Arrays

- `$ref = [0,1,2,3,];`
  - how would you print the 1 ?
- `$ref = [ [-1,1],45,78,[ 's','r'] ];`
  - how would you print the 's' ?

19

Hashes

- Remember that when you stick an anonymous list into a hash value (why not key ?)
- How to use it ?

20

sub references

- `$subref = sub(print “hello\n”);`
- `&$subref;`

21

Arrow Operator

- `->` infix operator for referencing
  - arrays
  - hashes
  - subroutines
- `$$arrayref[2]` can be `$arrayref->[2]`
- `&$subref()` can be `$subref->()`

22

Modules

- Idea: take a piece of code we are using a lot, and package it up so anyone can reuse it
- Advantages ??
- Disadvantages ?

23

Modules

- File with code:
  - something.pm
- Code which wants to use it:
  - use something;
- really::something::else
  - would be really/something/else.pm

24

Subroutines in Modules

- something.pm which defines foo();
- either
- something::foo();
- or export it correctly in the module file
  - @EXPORT = qw(foo);

25

Warning

- Be careful about exporting!!
- Does anyone know the difference between overloading and overriding ??

26

Bad Example

- say your package defined a sub called isdir ……
- If you export it what will happen ?

27

Versions

- can also define a $VERSION scalar which tells perl what your version is
- `use Function 3.2.1.2;`
- would check the Function.pm for version 3.2.1.2 or later

28

Objects

- Perl can be programmed in an object oriented fashion with objects/classes etc

29

Class declaration

- package Person;

- Will define the scope until the end of the current file

30

Perl OOP

- No way of doing strict encapsulation
- Expectation of good behavior

- Recycles concepts to get OOP

- you end up using references!
- Class is just a package
- Bless!! ties a reference with a class!

31

summary

- Objects = references
- Class = package
- Method = subroutine

32

Note

- Object's sub get an extra initial argument passed in:
  - name of the class
- sub package resolved during compile stage
- method resolved during runtime

33

Method calls

- $name->foo()
  foo $name ()
  - example of instance method being called
- $result = Math->pi();
  - class method being called

34

- $instance = new Math
  - can have problems if local new or Math sub defined
- $instance = new Math::
  $instance = Math::new;

35

Construction

- objects are references
- not all references are objects

- marking a reference with package is called BLESSING

```
$obj = { };
bless($obj);        #current package
bless($obj, Math);
```

36

- Person Example
- any questions from last lab ??

37

- How would we do a Fraction class ?

38

methods

- Accessor and modifier methods are defined in the current package
- Use arrow notation to access methods
- Object reference => name of method()

39

constructor

```
sub new {
  my $self = {

  _firstName => undef;
  _lastName => undef;

  };
  bless $self, ‘Person’;
  return $self;
  }
```

40

```
sub print {
  my ($self) = @_;

  #print info
  print $self->firstName . " ". $self->lastName;
  }
```

41

Instantiating

- `my $shlomo = new Person();`

42

accessing

```
sub firstName {
  my ( $self, $firstName) = @_;

  $self->{_firstName} = $firstName if defined
  ($firstName);

  return $self->{_firstName};
  }
```

43

Default sub

- AUTOLOAD
  - special sub which exists in the undefined package
  - when call an unknown sub will pass args and call AUTOLOAD
  - great for classes, if call something undefined

44

```
sub AUTOLOAD {
our $AUTOLOAD;
warn “attemp to call $AUTOLOAD
  failed\n”;
}
```

- so can access regular @_

45

Inheritance

- Classes can be used to extend the functionality of a class using a process called inheritance
- @ISA
  - perl keyword showing inheritance

46

Inheritance

- Process of relating classes
- if you create an instance of class Z which is derived from X
- Zinstance->foo
  - will look in Z
  - then look in X
  - then ??

47

Multiple inheritance

- We can have X->Y->Z relationship (single inheritance)
- Can have mulitple
- A->(B,C,D)
  - B->E
  - C->(F,G)
- So how to resolve foo when called on A??

48

- Current
- Parent
  - Left->Right
  - Recursively
  - Deep first
- Universal

- Then same thing for AUTOLOAD for all above

49

Garbage Collection

- High level languages do the garbage collection for you
- When leave a scope, all local variables are freed and their memory recycled

- How do you decide who to free ??

50

Reference counting

- An object can keep track of how many references are pointing to it
  - How can we do this ??
  - used on unix file systems

51

Problem

```
{
  my ($a, $b);
    $a = \$b;
    $b = \$a;
}
```

52

Worse

```
{
    my $a;
    $a = \$a;
}
```

solutions ??

53

Switch

- Back to CGI
- review: What is CGI ?
- How do we write CGI applications ?

54

Interacting

- GET
  - HTTP request directly to the cgi script by appending the URL
- POST
  - HTTP request in content of message, i.e it is stdin to your script
- Format of GET (default):
  - Value=key separated by &
  - Space replaced by +
  - URL conversion characters

55

Input Tag

- Each field is in an input tag
- Type
  - Text
  - Radio button
  - Checkbox
  - Pull down menus
  - etc
- Name
  - Symbolic name (so can recognize it)
- Value
  - Default value, or what the user will end up typing

56

Note: Encoding

- Spaces are turned to +
- & separates field
- Special characters are turned into %?? (hex)
  - “(“ is %28
  - So “class is great” = “class+is+great”

57

others

- Submit buttons
  - <input type=“submit”>
- Reset buttons
  - <input type=“reset”>
- Value will change the default name on the button
- try not to trick user….

58

Decoding Form Input

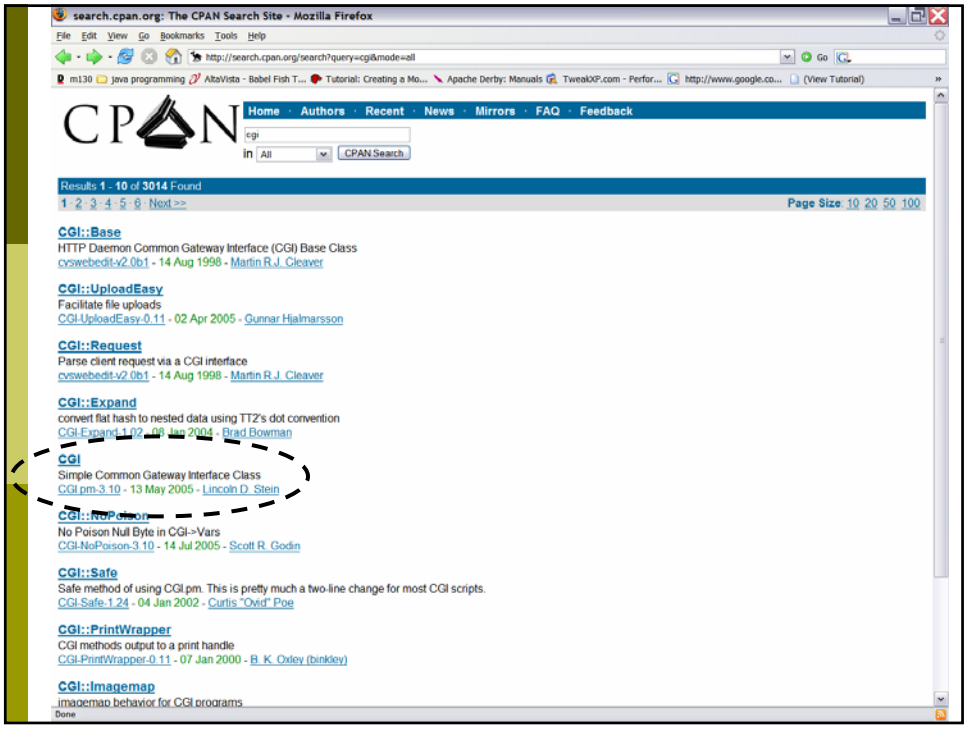
1. $ENV{QUERY_STRING}
2. If($ENV{REQUEST_METHOD} eq POST) { read $ENV{CONTENT_LENGTH}}
3. Split pairs around &
4. Split keys and values
5. Decode URL
6. Remember key,values

59

Drawback

- A lot of work
- Pain if we have multiple values associated with one key
- Must be easier way…..
- CGI.pm
  - Included after 5.003_07+

60



CGI.pm

- Allows you to handle cgi in a standard format
- Can save and load key,value pairs to standard file
- Helps in creating html documents to the server by streamlining certain operations and keeping it in an object oriented design

62

The bad news

- Can't use it in this class for our labs, since I want to teach how its done on the low level
- feel free to write your own test stuff with it
- understand what is happening on object level
- Want you to practice doing it the manual way…better for learning and later CGI + C/CPP

63

Summary: CGI

- Minimum the web server needs to provide to allow an external process to create WebPages.
- Goal: responding to queries and presenting dynamic content via HTTP.

64

File handling

- We covered basic file handling
- How does this change over the web?

65

File Locking

```
use Fcntl ":flock";

open FILE, "?????.txt" or die $!;

#one of these
flock FILE, LOCK_EX;
flock FILE, LOCK_SH;
.....
flock FILE, LOCK_UN;
```

66

Serving mp3 files

```
open(MP3FILE,"....") || die ....

my $buffer;
print "Content-type: audio/mp3\n\n";
binmode STDOUT;
while( read(MP3FILE, $buffer, 16384)){
 print $buffer;
}
```

67

Example

- http://..../cgi-bin/mp3server.cgi/Song.mp3

68

Argument passing

- Say you have a cool program which you can hook to the web…..
  - Give a cell phone
  - Give a message
  - Will send the cell phone a message

69

```
<HTML><HEAD>
<TITLE>Cool</TITLE>
</HEAD>
<BODY>

<form action="cgi-bin/cool.cgi" method="GET">
<p>Enter cell phone to use:
<input type="text" name="cellphone"></p>
<p>Enter Message:
<input type="text" name"message"></p>
<input type="submit">
</form>
</BODY></HTML>
```

70

```
Use CGI;
my $coolp = '/usr/local/bin/cellmsg';
```

```
my $q = new CGI;
my $cell = $q->param("cellphone");
my $msg = $q->param("message");
#error checking here
open PIPE, "$coolp $cell $message |" or die "Can
   not open cellphone program";
print $q->header( "text/plain");
print while <PIPE>
close PIPE;
```

71

What can go wrong?

72

- When executing command can in theory pass in the following arguments

Something ; rm –rf *.*

73

Perl Taint mode

- -T
  - Taints all data references (incoming)
- #!/usr/bin/perl –wT
- Flags data to make sure perl doesn't do anything insecure

74

Tainted?

- STDIN
- CGI

- If variables/values are tainted
- Tainted follows it around with assignments

```
Sub is_tainted {
   my $var = shift;
   my $blank = substr($var ,0,0);
   return not eval { eval “1 || $blank” || 1};
}
```

75

Why

- Why would you want to keep track of tainted data?

76

Getting out of taint

- Match related patterns ($1,$2 ..)
- Idea: would check for security problems and then allow it
- Reminder: only in taint mode if set

77

Other issues

- Remember with each user, your perl script is being instantiated and executed
- In general might want to be able to run alongside yourself (not only in web context).
  - How do we share a variable between instances (to pass information) ?

78

Command shell

- A better way of executing command shell arguments to a program is to divide the work
- Create an instance of the program you want to run
- Pass arguments directly to it, instead of using the command shell (where can combine multiple commands

79

fork/exec

```
my $pid = open PIPE, "-|";
die "problem forking $!" unless defined
  $pid;

unless($pid) {
  exec COOL, $message or die "cant open
  pipe $!";
```

80

Some more background

- When you work with CGI, many times you have to work with specific formats and files
- Need to know how it will be handled on client side
- One such common file, is graphics..

81

Graphics

- Formats:
  - GIF (Graphic Interchange Format)
    - 256 colors
    - LZW compression
    - Animation
    - Transparent bit
  - PNG (Portable Network Graphic)
    - 256 color / 16-bit gray / 48-bit true color
    - NOT LZW
    - Alpha channels
    - Interlacing algorithms

82

- JPEG (Joint Photographic Expert Group)
  - 24-bit color
  - Lossy compression
  - No animation/transparency
- PDF (Portable Document Format)
  - Postscript language for document layout

83

Image manipulation

- Many packages in perl to work with image data
- GD
  - Lightweight package
  - Port of c graphics library
  - Manipulation routines for PNG

84

CGI

- CGI is a common framework
- Perl is not the only player
- We will also be doing CGI + PERL|C|CPP

85

Alternatives

- ASP
  - Created by Microsoft for its servers
  - Mix code into html
  - Visual basic/javascript
- PHP
  - Apache webserver
  - Similar to perl
  - Embed code in html

86

Alt II

- Coldfusion
  - Webserver interprets std coldfusion call embedded in html, and can add code to run custom functions
  - Windows, and linux
- Java servelts
  - Compiled java classes invoked by web client
  - Code creates documents
- FastCGI
  - Threaded instance of perl continuasly running to help cgi perl run faster
- Mod_perl
  - Appache server perl thread to make perl cgi faster

87

Text handling

- One of the exciting developments in the last decade of computer science is data processing/mining/learning
- Many other area in and out of CS need data to be analyzed or presented in some (controlled but arbitrary fashion)

88

Handling data

- Using chiseled stone
- By hand (literally copy paste)
- Early mechanics (typwriters)
- Take 3157 ☺

89

Outputting text

- Many times will have multiple fields per line
- Arbitrary delimiters:
  - Comma
  - Tabs
  - Pipe |
- Make sure whatever you choose
  - Is either not/can't be present in the data
  - What if it is? How to represent these delimiters ??

90

Approach

- Memory vs disk based handling
- Brute force
- Divide and conquer
- Regexp is your friend

91

Ahead!

- Because CGI/Internet involves network based thinking, I will illustrate a quick example now.
- we will be doing this in next lab

92

Socket

- In order to communicate across computer networks (or between processes on the same computer)
- Need to decide on rules of communication
  - Language of protocol
  - Directional vs bi-directional
  - Throwaway or continuous
  - Life time of communication
  - Overhead
  - Priority
  - Location

93

IO:Socket client

see: http://search.cpan.org/~gbarr/IO-1.2301/IO/Socket/INET.pm

```
Use IO::Socket::INET;

$socket = IO::Socket::INET->new(
  PeerAddr => $remote_host,
  PeerHost => $remote_port,
  Proto => “tcp”,
  Type => SOCK_STREAM) or die…
```

94

```
#writing out
print $socket “hello World”;

#notice treatment of handle
$answer = <socket>;

close($socket);
```

95

Server version

```
my $server = IO::Socket::INET->new(
                        LocalPort=> $portnum,
                          LocalAddr => 'localhost',
                          Proto => 'tcp',
                          Reuse => "1",
                          Listen => "10")
   or die "could not start server on port $portnum ….\

while($client = $server->accept()) {
   #...
}
```

96

Next

- understand slides
- start homework
- have fun on the lab

97