Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Факультет вечернего, заочного и дистанционного обучения
Кафедра ЭВМ

Контрольная работа
по дисциплине «Системное программное обеспечение локальных вычислительных сетей»
студента 6 курса 500502 учебной группы
Авсеева С.П.

Минск 2010

Содержание

1. Лабораторная работа №1. Архитектура клиент-сервер

1.1. Задание. Вариант 7

TCP, параллельный тип сервера (процессы). Выполнение команды ОС. Клиент должен иметь возможность работы с несколькими серверами одновременно.

1.2. Реализация

Сервер (server.rb)

```
#!/usr/local/bin/ruby1.9.2

require 'optparse'

port = 1234
OptionParser.new do |opts|
  opts.banner = "Usage: server.rb [options]"

  opts.on("-p", "--port N", Numeric, "Run verbosely") do |v|
    port = v
  end
end.parse!

require 'socket'

acceptor = Socket.new(Socket::AF_INET, Socket::SOCK_STREAM, 0)
address = Socket.pack_sockaddr_in(port, 'localhost')
acceptor.bind(address)
acceptor.listen(10)

puts "listening on localhost:#{port}"

children = []
trap('EXIT') { acceptor.close }
trap('INT') do
  puts "\nkilling children: #{children.inspect}"
  children.each do |child|
    Process.kill 'INT', child
  end
end

3.times do

  child = fork do
```

```
            trap('INT') { exit }

            loop do
              socket, addr = acceptor.accept
              puts "[child␣#$$]␣accepted:␣#{addr.inspect_sockaddr}"
              loop do
                command = socket.gets
                break unless command
                data = `#{command} 2>&1`
                socket.puts data.length
                socket.flush
                socket.write data
                socket.flush
                puts "[child␣#$$]␣executed:␣#{command.strip}"
              end
              puts "[child␣#$$]␣closed:␣#{addr.inspect_sockaddr}"
              socket.close
            end

            exit
          end
  children << child

end

puts "3␣children␣started:␣#{children.inspect}"
Process.waitall
```

Клиент (client.rb)

```
#!/usr/local/bin/ruby1.9.2
require 'rubygems'
require 'ruby-debug'
require 'socket'
require 'readline'

trap('INT') { teardown }

class Socket
  def pretty_address
    remote_address.inspect_sockaddr
  end
end

def teardown
  @connections.keys.each {|address| close(address)}
end

def ask(socket)
  @connections ||= {}
  prompt = socket ? "[#{socket.pretty_address}]␣" : ""
  prompt << "conn:#{@connections.size}␣>␣"
  answer = Readline.readline(prompt, true)
  teardown && exit unless answer
  answer.chomp.strip
end

def use(address)
  host, port = address.split(':')
  if host.nil? || port.nil?
    puts "invalid␣address␣#{address}"
    return
  end

  @connections ||= {}
  socket = @connections[address]
  if socket
    puts "using␣previous␣connection␣to␣#{address}"
  else
    puts "establishing␣new␣connection␣to␣#{address}"
    socket = Socket.new(Socket::AF_INET, Socket::SOCK_STREAM, 0)
    sockaddr = Socket.pack_sockaddr_in(port, host)
    socket.connect(sockaddr)
    @connections[address] = socket
  end

  return socket
```

```
end

def close(address)
  socket = @connections[address]
  if socket
    puts "closing␣connection␣to␣#{address}"
    socket.close
    @connections[address] = nil
  else
    puts "there␣is␣no␣connection␣to␣#{address}.␣do␣nothing"
  end
  address
end

def list
  connections = @connections.map do |address, socket|
    "#{address}␣=>␣#{socket.pretty_address}"
  end
  puts connections
end

current_socket = nil

loop do
  command = ask(current_socket)
  next if command.nil? || command.empty?
  action, address = command.split('␣')
  case action
  when 'list'
    list
  when 'use'
    current_socket = use(address)
  when 'close'
    current_address = current_socket.try(:pretty_address)
    if close(address || current_address) == current_address
      current_socket = nil
    end
  else
    if current_socket
      current_socket.puts(command)
      current_socket.flush
      length = current_socket.gets.to_i
      puts current_socket.read(length)
    else
      puts "offline.␣connect␣to␣server␣with␣'use␣host:port'"
    end
  end
end
```

2. Лабораторная работа №2. Реализация протокола высокого уровня на примере FTP

2.1. Задание. Вариант 7

Реализовать FTP-клиент с интерпретатором либо FTP-сервер с поддержкой многопоточности. Реализовать минимальный набор FTP-команд (USER, PASS, RETR, STOR, REST, QUIT) и три любые дополнительные команды из заданной группы. Реализовать механизм тайм-аута. Пакеты должны иметь фиксированный размер.

- Компонент: клиент
- Режим обмена: пассивный
- Файловое представление: Image
- Группа команд: Service (PWD, CWD, DELE, STAT)

2.2. Реализация

```
#!/usr/local/bin/ruby1.9.2
require 'rubygems'
require 'socket'
require 'ripl'
require 'ripl/multi_line'
require 'ripl/irb'

class Ftp
  class Code
    def initialize(code)
      @code = code
    end

    def positive_preliminary?
      @code =~ /^1/
    end

    def positive_completion?
      @code =~ /^2/
    end

    def positive_intermediate?
      @code =~ /^3/
    end
```

```
  def positive?
    @code =~ /^[123]/
  end

  def negative?
    @code =~ /^[45]/
  end
end

DEFAULT_BLOCKSIZE = 1024
FTP_PORT = 21
CRLF = "\r\n"

def initialize(host = nil, port = nil)
  @host = host
  @port = port
  connect(host, port) if host
end

def connect(host, port = FTP_PORT)
  puts "connect␣to␣#{host}..."
  @socket = open_socket(host, port)
  getresp
end

def login(user = "anonymous", passwd = nil, acct = nil)
  return unless @socket
  if user == "anonymous" and passwd == nil
    passwd = "anonymous@"
  end

  code, resp = sendcmd('USER␣' + user)
  code, resp = sendcmd('PASS␣' + passwd) if code.positive_intermediate?
  code, resp = sendcmd('ACCT␣' + acct) if code.positive_intermediate?
  puts "Access␣denied" and return unless code.positive_completion?
  sendcmd("TYPE␣I")
  @logged_in = true
end

def quit
  if @logged_in
    sendcmd("QUIT")
    @logged_in = false
  end
end

def ls(*args, &block)
  cmd = "LIST␣#{args.join('␣')}"
```

```
  if block
    retr(cmd, &block)
  else
    retr(cmd) { |line| puts line }
  end
end

def pwd
  code, resp = sendcmd("PWD")
  puts resp
end

def cd(dirname)
  sendcmd("CWD␣#{dirname}")
end

def rm(filename)
  sendcmd("DELE␣#{filename}")
end

def stat
  sendcmd("STAT")
end

def get(remotefile, localfile = File.basename(remotefile),
        blocksize = DEFAULT_BLOCKSIZE, resume = nil)
  result = nil
  if localfile
    if resume
      offset = File.size?(localfile)
      f = open(localfile, "a")
    else
      offset = nil
      f = open(localfile, "w")
    end
  else
    result = ""
  end

  begin
    retr("RETR␣" + remotefile, blocksize, resume) do |data|
      if localfile
        f.write(data)
      else
        result.concat(data)
      end
    end
    return result
```

```
  ensure
    f.close if localfile
  end
end

def put(localfile, remotefile = File.basename(localfile),
        blocksize = DEFAULT_BLOCKSIZE, resume = nil)
  return if remotefile.nil? || remotefile.strip.empty?
  offset = File.size?(remotefile) if resume
  f = File.open(localfile)
  begin
    if offset
      stor("APPE␣" + remotefile, f, blocksize, offset)
    else
      stor("STOR␣" + remotefile, f, blocksize, offset)
    end
  ensure
    f.close
  end
end

private

def retr(cmd, blocksize = DEFAULT_BLOCKSIZE, offset = nil)
  conn = transfercmd(cmd, offset)
  return unless conn
  loop do
    data = conn.read(blocksize)
    break if data == nil
    yield(data)
  end
  conn.close
  getresp
end

def stor(cmd, file, blocksize = DEFAULT_BLOCKSIZE, offset = nil)
  file.seek(rest_offset, IO::SEEK_SET) if offset
  conn = transfercmd(cmd)
  return unless conn
  loop do
    buf = file.read(blocksize)
    break if buf.nil?
    conn.write(buf)
  end
  conn.close
  getresp
end
```

```
def pasv
  code, resp = sendcmd("PASV")
  numbers = resp[/\((.+)\)/, 1].split(',').map(&:to_i)
  host = numbers[0, 4].join('.')
  port = (numbers[4] << 8) + numbers[5]
  return host, port
end

def sendcmd(cmd)
  puts "[send]␣#{cmd}"
  @socket.write(cmd + CRLF)
  getresp
end

def transfercmd(cmd, rest_offset = nil)
  host, port = pasv
  conn = open_socket(host, port)
  if rest_offset
    code, resp = sendcmd("REST␣" + rest_offset.to_s)
    return unless code.positive_intermediate?
  end
  code, resp = sendcmd(cmd)
  resp = getresp if code.positive_completion?
  return unless code.positive_preliminary?
  return conn
end

def getline
  line = @socket.readline
  line.sub!(/(\r\n|\n|\r)\z/n, "")
  puts "[recv]␣#{line}"
  line
end

def getresp
  buff = line = getline
  if line[3] == "-"
    code = line[0, 3]
    begin
      line = getline
      buff << "\n" << line
    end until line[0, 3] == code and line[3] != "-"
  end
  buff << "\n"
  [Code.new(buff[0, 3]), buff]
end

def open_socket(host, port)
```

```
    socket = Socket.new(Socket::AF_INET, Socket::SOCK_STREAM, 0)
    socket.connect(Socket.pack_sockaddr_in(port, host))
    return socket
  end

end

class Ripl::Shell
  def print_result(not_used); end
  def print_eval_error(exception); warn(exception); end
  EXIT_WORDS.delete('quit')
end

ftp = Ftp.new
at_exit { puts "quit"; ftp.quit }
Ripl.start(:prompt => 'ftp>␣', :binding => ftp.instance_eval{ binding })
```

Пример сеанса взаимодействия с сервером

```
$ ./ftpclient.rb
ftp> connect('localhost') && login
connect to localhost...
[recv] 220 ProFTPD 1.3.3a Server (avsej.local) [::ffff:127.0.0.1]
[send] USER anonymous
[recv] 331 Anonymous login ok, send your complete email address as your password
[send] PASS anonymous@
[recv] 230-Welcome, archive user anonymous@localhost !
[recv] 230-
[recv] 230-The local time is: Wed Dec 29 17:29:53 2010
[recv] 230-
[recv] 230-This is an experimental FTP server.  If you have any unusual problems,
[recv] 230-please report them via e-mail to <root@avsej.home>.
[recv] 230-
[recv] 230 Anonymous access granted, restrictions apply
[send] TYPE I
[recv] 200 Type set to I
ftp> stat
[send] STAT
[recv] 211-Status of 'avsej.local'
[recv] 211-Connected from localhost (::ffff:127.0.0.1)
[recv] 211-Logged in as ftp
[recv] 211-TYPE: BINARY, STRUcture: File, Mode: Stream
[recv] 211-No data connection
[recv] 211 End of status
ftp> pwd
[send] PWD
[recv] 257 "/" is the current directory
257 "/" is the current directory
ftp> ls
[send] PASV
[recv] 227 Entering Passive Mode (127,0,0,1,157,67).
[send] LIST
[recv] 150 Opening BINARY mode data connection for file list
-rw-r--r--   1 ftp      ftp          4296 Dec 29 12:44 ftpclient.rb
-rw-r--r--   1 ftp      ftp        199988 Dec 29 14:52 qt-ruby.pdf
drwxrwxrwx   2 ftp      ftp          4096 Dec 29 12:18 samples
-rw-r--r--   1 ftp      ftp          2111 Dec 29 14:51 session.txt
-rw-rw-rw-   1 ftp      ftp           170 Nov  3 15:03 welcome.msg
[recv] 226 Transfer complete
ftp> get 'qt-ruby.pdf'
[send] PASV
[recv] 227 Entering Passive Mode (127,0,0,1,144,166).
[send] RETR qt-ruby.pdf
[recv] 150 Opening BINARY mode data connection for qt-ruby.pdf (199988 bytes)
[recv] 226 Transfer complete
ftp> cd 'samples'
```

```
[send] CWD samples
[recv] 250 CWD command successful
ftp> pwd
[send] PWD
[recv] 257 ”/samples” is the current directory
257 ”/samples” is the current directory
ftp> ls
[send] PASV
[recv] 227 Entering Passive Mode (127,0,0,1,172,152).
[send] LIST
[recv] 150 Opening BINARY mode data connection for file list
[recv] 226 Transfer complete
ftp> put ’qt-ruby.pdf’
[send] PASV
[recv] 227 Entering Passive Mode (127,0,0,1,137,171).
[send] STOR qt-ruby.pdf
[recv] 150 Opening BINARY mode data connection for qt-ruby.pdf
[recv] 226 Transfer complete
ftp> ls
[send] PASV
[recv] 227 Entering Passive Mode (127,0,0,1,208,31).
[send] LIST
[recv] 150 Opening BINARY mode data connection for file list
-rw-r--r--   1 ftp      ftp        199988 Dec 29 15:30 qt-ruby.pdf
[recv] 226 Transfer complete
ftp> quit
[send] QUIT
[recv] 221 Goodbye.
```