

Chat client/server print failed

Source: <http://coding.derkeiler.com/Archive/Perl/comp.lang.perl.misc/2008-01/msg00585.html>

- *From:* deadpickle <deadpickle@xxxxxxxxxx>
 - *Date:* Tue, 15 Jan 2008 16:32:52 -0800 (PST)
-

This is a chat client wrote in perl Gtk2. The problem that I am running into is that when you type and click send I get a "print() on closed filehandle GEN0 at chat-client.pl line 332" error. This error is the print statement in the send_msg_all sub. I cant figure out how the file handle is closed and am wondering if anyone can see why. I'll leave the server running for testing purposes.

the Client:

```
#!/usr/bin/perl
# Flow of the Program:
# *Send message to the server – send_msg_all
# *Connect to the server – sub connect_server
# –unblock the server – nonblock
# –Login to the server – send_login
# –Timer started to wait for messages – wait_for_msg
# >Handler – handle
# $Process the incoming meswsages – process_incoming
# @Recieve messages and display in textview – rcv_msg
```

```
use warnings;
use strict;
use Gtk2 -init;
use Glib qw/TRUE FALSE/;
use IO::Socket::INET;
use Tie::RefHash;
use IO::Select;
```

```
#global variables
my $buffer;
my $host = "Deadpickle-hobo";
my $port = 6666;
my $conn_stat = 'idle';
my %inbuffer = ();
my %outbuffer = ();
my %ready = ();
my $select;
my $conn;
my $user;
```

Chat client/server print failed

```
#the main chat widget
my $main_window = Gtk2::Window->new("toplevel");
$main_window->signal_connect(delete_event => sub { Gtk2->main_quit; });
$main_window->set_default_size(250, 200);

my $table = Gtk2::Table->new(4, 2, FALSE);

$buffer = Gtk2::TextBuffer->new;
my $button = Gtk2::Button->new("Send");
my $entry = Gtk2::Entry->new();

my $label = Gtk2::Label->new("Chat Client Test");

my $textview = Gtk2::TextView->new_with_buffer($buffer);
$textview->set_cursor_visible (FALSE);
my $swindow = Gtk2::ScrolledWindow->new( undef, undef);
$swindow->set_policy( 'automatic', 'automatic');
$swindow->set_shadow_type( 'etched-out');

$swindow->add( $textview);

$table->attach_defaults($label, 0, 1, 0, 1);
$table->attach_defaults($swindow, 0, 2, 1, 3);
$table->attach_defaults($entry, 0, 1, 3, 4);
$table->attach_defaults($button, 1, 2, 3, 4);
$main_window->add($table);

$main_window->show_all();

$button->signal_connect("clicked" => sub { send_msg_all($entry->
    get_text); $entry->set_text(""); } );

#run the login dialog
dialog($buffer);

Gtk2->main;

#-----Login Dialog-----
sub dialog{
my $buffer = shift;

my $dialog_window = Gtk2::Window->new('toplevel');
$dialog_window->signal_connect(delete_event => sub { Gtk2->
    main_quit});

my $dialog_table = Gtk2::Table->new(2, 2, FALSE);
```

Chat client/server print failed

```
my $dialog_label1 = Gtk2::Label->new('Chat Login:');
my $dialog_label2 = Gtk2::Label->new('User:');
my $dialog_label3 = Gtk2::Label->new('Host:');
my $chat_user = Gtk2::Entry->new();
$chat_user->set_text("");
my $dialog_button1 = Gtk2::Button->new('Connect');

$dialog_table->attach_defaults($dialog_label1, 0, 1, 0, 1);
$dialog_table->attach_defaults($chat_user, 1, 2, 0, 1);
$dialog_table->attach_defaults($dialog_button1, 1, 2, 1, 2);

$dialog_button1->signal_connect("clicked" => sub {$user = $chat_user->
    get_text; $dialog_window->destroy; $buffer->insert(($buffer->
    get_end_iter), "Username: $user...\n"); connect_server()});

$dialog_window->add($dialog_table);

$dialog_window->show_all;

return 1;
}
#-----Connect to server-----
#establishes connection to the server
sub connect_server{
if ($conn_stat ne 'connected') {
$buffer->insert(($buffer->get_end_iter), "Connecting to Server
$host:$port...\n");

$conn = IO::Socket::INET->new(PeerAddr => $host, PeerPort =>
$port, Proto => 'tcp') or popup_err(1);

if ($conn) {
%inbuffer = ();
%outbuffer = ();
%ready = ();
tie %ready, 'Tie::RefHash';
nonblock($conn);
$select = IO::Select->new($conn);
$conn_stat = 'connected';
$buffer->insert(($buffer->get_end_iter), "Connected!\n");

#send login to server
send_login();

#start the timer that monitors incoming messages
my $timer_waiting = Glib::Timeout->add(100, \&wait_for_msg);

print "$conn\n";
}
}
```

Chat client/server print failed

Chat client/server print failed

```
}
}
#-----Error popup-----
# pops up an error message
sub popup_err{
my ($error_code) = @_ ;
my $error;

if ($error_code == 1) {$error = "Cannot create Socket!"}
elsif ($error_code == 2) {$error = "Username too Short!"}
elsif ($error_code == 3) {$error = "No connection Established!"}
elsif ($error_code == 4) {$error = "Already Logged on with This User
Name!"}
elsif ($error_code == 5) {$error = "Not Connected!"}
elsif ($error_code == 6) {$error = "User Successfully Added!"}
elsif ($error_code == 7) {$error = "Error Registering User!"}
elsif ($error_code == 8) {$error = "Already Logged Out!"}
else {$error = "Unknown Error!"}

$buffer->insert(($buffer->get_end_iter), "$error\n");

my $error_dialog = Gtk2::MessageDialog->new($main_window, 'destroy-
with-parent', 'error', 'ok', "$error");

$error_dialog->run;
$error_dialog->destroy;
}
#-----blocking-----
# nonblock($socket) puts socket into nonblocking mode
sub nonblock {
my $socket = shift;

$socket->blocking(0);
}
#-----Message Waiting-----
# Wait for incoming messages from the server relayed from clients
sub wait_for_msg {

print "waiting\n";

if ($conn_stat eq 'connected') {
my ($list_size, $msg);
my $server;
my $rv;
my $data;

# check for new information on the connections we have
# anything to read or accept?
foreach $server ($select->can_read(1)) {
# read data
$data = ";
```

Chat client/server print failed

```
$rv = $server->recv($data, 'POSIX::BUFSIZ', 0);

unless (defined($rv) && length $data) {
# This would be the end of file, so close the client
delete $inbuffer{$server};
delete $outbuffer{$server};
delete $ready{$server};

$select->remove($server);
close $server;
next;
}

$inbuffer{$server} .= $data;

# test whether the data in the buffer or the data we
# just read means there is a complete request waiting
# to be fulfilled. If there is, set $ready{$client}
# to the requests waiting to be fulfilled.
while ($inbuffer{$server} =~ s/(.*\n)//) {

push( @{$ready{$server}}, $1 );
}
}

# Any complete requests to process?
foreach $server (keys %ready) {

handle($server);
}
}
}
#-----Handler-----
# handle($socket) deals with all pending requests for $client
sub handle {
# requests are in $ready{$server}
# send output to $outbuffer{$server}
my $server = shift;
my $request;

foreach $request (@{$ready{$server}}) {
# $request is the text of the request
# put text of reply into $outbuffer{$client}
chomp $request;
process_incoming($server, $request);
}
delete $ready{$server};
}
#-----Process Incoming-----
sub process_incoming {
my ($server, $msg) = @_;
```

Chat client/server print failed

Chat client/server print failed

```
my @logged_users;

my @rcvd_msg = split(/::/, $msg);

if ($rcvd_msg[1] eq "1") {
# Login responses
# 12 = already logged on
# 03 = logged in

if($rcvd_msg[2] eq "03") {
print "Successfully Logged in!\n";
} elsif ($rcvd_msg[2] eq "12") {
popup_err(4);
} else {
# Create pop-up for error!
print "Error Logging in ", $msg, "\n";
popup_err(5);
}
} elsif ($rcvd_msg[1] eq "2") {
# register response
if ($rcvd_msg[2] eq "06") {
print "New user successfully registered!\n";
popup_err(6);
} elsif ($rcvd_msg[2] eq "02") {
print "$msg\n";
popup_err(4);
} else {
print "$msg\n";
popup_err(7);
}
} elsif ($rcvd_msg[1] eq "3") {
# quit response
print "$msg\n";
# $exit_cond = 0;
} elsif ($rcvd_msg[1] eq "4") {
# log out response
# 14 = user logged off
# 13 = user not logged in to begin with
print "$msg\n";
if($rcvd_msg[2] == 13) {
popup_err(8); # not logged in
}
} else {
# # clear the buddy list
# $list_size = $buddy_list->size;
# $list_size = $list_size - 1;
# $buddy_list->delete(0,$list_size);
# }
# $menu_file->update;
# } elsif ($rcvd_msg[1] eq "5") {
# # delete existing list of users
```

Chat client/server print failed

Chat client/server print failed

```
# $list_size = $buddy_list->size;
# if($list_size > 0) { $buddy_list->delete(0,$list_size); }
## get users list response
## if server response for proto 5 is 17 then Draw in
$buddy_list
# if ($rcvd_msg[2] == 17) {
# @logged_users = split (/, $rcvd_msg[3]);
# foreach (@logged_users) {
# $buddy_list->insert('end', "$_");
# }
# } elsif ($rcvd_msg[2] eq 18) {
## generate error for login
# print "Please Log in to server first!\n";
# print "$msg\n";
# popup_err(51);
# } else {
# print "Unknown error updating buddy list:\n";
# print "$msg\n";
# popup_err(52);
# }
# $menu_file->update;
} elsif ($rcvd_msg[1] eq "6") {
# receive user message
# 13 – user not logged in
# 23 – buddy (target) not logged in
print "$msg\n";
rcv_msg($rcvd_msg[3], $rcvd_msg[4]);
# } elsif ($rcvd_msg[1] eq "7") {
## receive global message
# print "$msg\n";
# rcv_msg_all($rcvd_msg[3], $rcvd_msg[4]);
# } elsif ($rcvd_msg[1] eq "8") {
# if ($rcvd_msg[2] == 23) {
# popup_err(81);
# } elsif ($rcvd_msg[2] eq "13") {
# popup_err(82);
# } else {
## receive query information
# print "$msg\n";
# process_query($msg);
# }
# $menu_file->update;
} else {
print "Unrecognized response: $msg\n";
# popup_err(92);
exit(0);
}
# if($err) { print "ERROR: $err\n"; }
}
#-----Send message to all-----
sub send_msg_all {
```

Chat client/server print failed

Chat client/server print failed

```
my ($msg) = @_;  
  
print "$conn\n";  
  
if(defined $conn) {  
# Send a the Message to server  
print "Sending\n";  
print $conn "7\:\:$user\:\:$msg\n";  
} else {  
popup_err(3);  
}  
}  
#-----Send login-----  
#logs the user name on the server  
sub send_login {  
# my ($u) = @_;  
  
if(defined $conn) {  
if(length($user) > 0) {  
#send login to server  
print $conn "1\:\:$user\n";  
# update_info();  
} else {  
popup_err(2);  
}  
} else {  
popup_err(3);  
}  
}  
#-----Display Message-----  
sub rcv_msg {  
my ($from, $msg) = @_;  
  
print "Received message from $from\n";  
if(defined $conn) {  
print "Already Connected: Proceeding with message!\n";  
# $status->insert('end',"[$from]: $msg\n");  
} else {  
print "No connection established!\n";  
popup_err(3);  
}  
}
```

The Server:

```
#!/usr/local/bin/perl  
#  
# SERVER PROTOCOLS  
# 01 – Login ..... 1::username::passwd  
# 02 – Register New User 2::username::passwd
```

Chat client/server print failed

Chat client/server print failed

```
# 03 – Terminate Program ..... 3::username::passwd
# 04 – Logoff Server 4::username::passwd
# 05 – Get Logged Users ..... 5::username::passwd
# 06 – Message another user
6::username::passwd::rcpt::message
# 07 – Global message ..... 7::username::passwd::message
# 08 – Query buddy 8::username::passwd::rcpt
# 09 – Update Buddy Info .....
9::username::passwd::Name::Email::Quote

use POSIX;
use IO::Socket;
use IO::Select;
use Socket;
use Fcntl;
use Tie::RefHash;

$port = 6666;
my %hosts;

# Create Server
$server = IO::Socket::INET->new(LocalPort => $port,
Listen => 100 )
or die "Can't make server socket: $@\n";

print "Server created. Waiting for events...\n";

# begin with empty buffers
%inbuffer = ();
%outbuffer = ();
%ready = ();

tie %ready, 'Tie::RefHash';

nonblock($server);
$select = IO::Select->new($server);

# Main loop: check reads/accepts, check writes, check ready to process
while (1) {
my $client;
my $rv;
my $data;

# check for new information on the connections we have

# anything to read or accept?
foreach $client ($select->can_read(1)) {

if ($client == $server) {
# accept a new connection
$client = $server->accept();
```

Chat client/server print failed

```
$select->add($client);
nonblock($client); #subroutine
} else {
# read data
$data = "";
$rv = $client->recv($data, POSIX::BUFSIZ, 0);

unless (defined($rv) && length $data) { #Runs the
bottom statement unless it is true
# This would be the end of file, so close the client
delete $inbuffer{$client};
delete $outbuffer{$client};
delete $ready{$client};

$select->remove($client);
close $client;
next;
}

$inbuffer{$client} .= $data; #add the recieved data
to the inbuffer

# test whether the data in the buffer or the data we
# just read means there is a complete request waiting
# to be fulfilled. If there is, set $ready{$client}
# to the requests waiting to be fulfilled.
# Disceting the matching variable:
# =~: between variable and and regular expression
# s/PATTERN/REPLACEMENT/: searchs the string for the
pattern then replaces it with the replacement text
# (.*\n):
# . match any character except newline
# * match zero or more times
# \n newline
while ($inbuffer{$client} =~ s/(.*\n)/) { #If there is
data in the inbuffer; searches for a string with a newline at the end
push( @{$ready{$client}}, $1 ); # $1 is the matched
string that is added the the hash/array %ready; this must be the
REFHASH
}
}
}

# Any complete requests to process?
foreach $client (keys %ready) { #calls the rehash keys
handle($client); #subroutine
}

# Buffers to flush?
foreach $client ($select->can_write(1)) { #see what clients are
ready to be wrote to
```

Chat client/server print failed

```
# Skip this client if we have nothing to say
next unless exists $outbuffer{$client}; #run the next
iteration unless the outbuffer exists

$rv = $client->send($outbuffer{$client}, 0);
unless (defined $rv) { #run the statement unless $rv is
defined
# Whine, but move on.
warn "I was told I could write, but I can't.\n";
next;
}
if ($rv == length $outbuffer{$client} ||
$! == POSIX::EWOULDBLOCK) {
substr($outbuffer{$client}, 0, $rv) = "";
delete $outbuffer{$client} unless length
$outbuffer{$client};
} else {
# Couldn't write all the data, and it wasn't because
# it would have blocked. Shutdown and move on.
delete $inbuffer{$client};
delete $outbuffer{$client};
delete $ready{$client};

$select->remove($client);
close($client);
next;
}
}

# Out of band data?
foreach $client ($select->has_exception(0)) { # arg is timeout
# Deal with out-of-band data here, if you want to.
print "DEBUG ME!\n";
}
}

#-----Handler-----
# handle($socket) deals with all pending requests for $client
sub handle {
# requests are in $ready{$client}
# send output to $outbuffer{$client}
my $client = shift;
my $request;

foreach $request (@{$ready{$client}}) { #cycle through the
clients
# $request is the text of the request
# put text of reply into $outbuffer{$client}
chomp $request;
rcvd_msg_from_client($client, $request); #subroutine

# $outbuffer{$client} .= "$request";
```

Chat client/server print failed

Chat client/server print failed

```
}
delete $ready{$sclient}; #remove the client from the processes
}
#-----Blocking-----
# nonblock($socket) puts socket into nonblocking mode
sub nonblock {
my $socket = shift;

$socket->blocking(0);
}
#-----Receive Message from Client-----
sub rcvd_msg_from_client {
# This sub receives the message from the client and processes
# it.
my ($sclient, $request) = @_; #drop the passed variables

if (length($request) ne 0) { #if there is a request
chomp $request;
print "CLIENT QUERY: '$request\n"; #this is used for all
requests into the server
# CLIENT QUERY in the form of ID_NUMBER::USERNAME:: ...
my @msg = split(/::/, $request); #assigned by the client;
runs the requested subs
if($msg[0] eq '1') {logon($sclient, @msg)}
# } elsif ($msg[0] eq 2) {
# register($sclient, @msg);
# } elsif ($msg[0] eq 3) {
# quit($sclient, @msg);
# } elsif ($msg[0] eq 4) {
# logoff($sclient, @msg);
# } elsif ($msg[0] eq 5) {
# return_logged_users($sclient, @msg);
# } elsif ($msg[0] eq 6) {
# msg_user($sclient, @msg);
elsif ($msg[0] eq 7) {msg_all_users($sclient, @msg)}
# } elsif ($msg[0] eq 8) {
# query_buddy($sclient, @msg);
# } elsif ($msg[0] eq 9) {
# update_info($sclient, @msg);
else {
print "Unrecognized ID $msg[0]\n";
$outbuffer{$sclient} .= "Unrecognized ID $msg[0]\n";
}
}
}
#-----login-----
sub logon {
# This sub checks the length of the data passed to it,
# if the length is ne 0, then it checks to make sure the
# data authenticates (i.e. login name and password authenticate)
# If this happens then chk_for_login is called else user is
```

Chat client/server print failed

```
# logged in.
my ($client, @msg) = @_;

my $client_ip = get_hostaddr($client);

if (length($msg[1]) ne 0) {
    print "User $msg[1] attempting login...\n";
    # if (authorize($msg[1], $msg[2])) {
    # check if user is already logged in
    if (chk_for_login($msg[1])) { # check if username is already
    in use
    # user is already logged in
    print "SERVER::",time_stamp(),"::03::User $msg[1] $client_ip
    logged in.\n";
    $outbuffer{$client} .= "SERVER::$msg[0]::03::$msg[1] logged in!
    \n";
    } else {
    # user is not logged in, add to %hosts
    my $current_time = time_stamp();
    $hosts{$msg[1]}->{'ip'} = $client_ip;
    $hosts{$msg[1]}->{'status'} = 'connected';
    $hosts{$msg[1]}->{'logged_in'} = 'yes';
    $hosts{$msg[1]}->{'user_name'} = $msg[1];
    $hosts{$msg[1]}->{'con_time'} = $current_time;
    $hosts{$msg[1]}->{'connection'} = $client;
    print "SERVER::",time_stamp(),"::03::User $msg[1] $client_ip
    logged in.\n";
    $outbuffer{$client} .= "SERVER::$msg[0]::03::$msg[1] logged in!
    \n";
    }
    # } else {
    # print "SERVER::",time_stamp(),"::00::User $msg[1] NOT logged in.
    \n";
    # $outbuffer{$client} .= "SERVER::$msg[0]::00::$msg[1] NOT logged
    in!\n";
    # }
    } else {
    print "SERVER::",time_stamp(),"::00::Null user not logged in.\n";
    print "ERROR::",time_stamp(),"::09::Invalid logon attempt.\n";
    $outbuffer{$client} .= "SERVER::$msg[0]::00::Could not login with
    that name.\n";
    }
    }
    #-----Get host address-----
    sub get_hostaddr {
    my ($client) = @_;
    my $sock = $client;
    return $sock->peerhost();
    }
    #-----Check current users-----
    sub chk_for_login {
```

Chat client/server print failed

```
# chk_for_login() checks the hash of logins for a matching
# key. if one is found then 1 is returned, else 0 is returned
my ($user_name) = @_ ;
foreach (keys %hosts) {
if($_ eq $user_name) {
return 1;
} else {
return 0;
}
}
}
}
}
#-----Timestamp-----
sub time_stamp {
# This sub returns the time using gmtime();
my ($s, $m, $h, $dy, $mo, $yr, $wd, $dst);
($s, $m, $h, $dy, $mo, $yr, $wd, $dst) = gmtime(); # get the date
$mo++;
$yr = $yr - 100;
if ($mo < 10) { $mo = "0".$mo; }
if ($dy < 10) { $dy = "0".$dy; }
if ($yr < 10) { $yr = "0".$yr; }
if ($h < 10) { $h = "0".$h; }
if ($m < 10) { $m = "0".$m; }
if ($s < 10) { $s = "0".$s; }
return "$mo-$dy-$yr $h:$m:$s";
}
#-----Message all users-----
sub msg_all_users {
# sends a global message to every user logged in
my ($client, @msg) = @_ ;
my $rcpt;

print "SERVER::",time_stamp(),"::21::Global message attempt:
$msg[1]\n";
if (chk_for_login($msg[1])) { # check if user is logged in
print "SERVER::",time_stamp(),"::21::$msg[1] cleared for GM\n";
foreach (keys %hosts) { # for each logged in user
$rcpt = get_link($_); # get rcpts connection
$outbuffer{$rcpt} .= "SERVER::$msg[0]::21::$msg[1]::$msg[3]\n";
}
}
}
}
#-----Get the links to the connected
users-----
sub get_link {
# This sub gets a connection link to a user
my ($user_name) = @_ ;
foreach (keys %hosts) {
if($_ eq $user_name) {
return $hosts{$_}->{'connection'};
}
}
}
```

Chat client/server print failed

```
}  
}  
.
```