Thought question
Announcements

• Some clarifications to project 1 spec posted to Piazza

• Today’s plan:
  – Socket options
  – Signals and timeouts
  – Q&A related to project 1
Part 1: Socket options
Socket options: motivation

• Basic “out of the box” socket functionality fine for most purposes
  – But what if you need to tweak the behavior?

• Can set/get ‘options’ on sockets

• These options apply to different layers of the network stack:
  – IP
  – TCP
  – Socket
Interesting options

• Send and receive buffer sizes
  – What is the default?

  [gmporter@seed-f60-100 ~]$ cat /proc/sys/net/ipv4/tcp_rmem
  4096 87380 6291456

  [gmporter@seed-f60-100 ~]$ cat /proc/sys/net/ipv4/tcp_wmem
  4096 16384 4194304

Minimum  Default  Maximum

• Can we change that value?
  – Yes!
Setting/getting socket options

GETSOCKOPT(2)

NAME

getsockopt, setsockopt – get and set options on sockets

SYNOPSIS

#include <sys/types.h>
#include <sys/socket.h>

/* See NOTES */

int getsockopt(int sockfd, int level, int optname,
void *optval, socklen_t *optlen);

int setsockopt(int sockfd, int level, int optname,
const void *optval, socklen_t optlen);

<table>
<thead>
<tr>
<th>Level</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOL_SOCKET</td>
<td>SO_SNDBUF</td>
<td>Send buffer size</td>
</tr>
<tr>
<td></td>
<td>SO_REUSEADDR</td>
<td>Allow TCP port to be reused immediately</td>
</tr>
<tr>
<td></td>
<td>SO_RCVTIMEO</td>
<td>Set a recv() timeout</td>
</tr>
<tr>
<td></td>
<td>SO_SNDTIMEO</td>
<td>Set a send() timeout</td>
</tr>
</tbody>
</table>
Demo 1

Enabling a server socket to immediately reuse a TCP port
Part 2: Signals and timeouts
Signals

• OS mechanism to asynchronously interrupt a program

• Why is this useful?
  – Kill a runaway/hung process
  – Notify program that there is activity on the keyboard
  – Disk read operation has completed
  – The dreaded SIGSEGV
Signals in action

• $ sleep 9999999
• How to stop this program?
Signals and networking

• Signals can be used to implement *timeouts*
• Examples:
  – Close connection after 3 minutes of inactivity
  – HTTP server: is the client going to send another request? Set timeout for e.g., 5 seconds
• Useful any time you need to stop blocking
  – recv()
  – send()
  – ...
Signals

• SIGALRM
  – Issued after a set period of time goes by
  – Like an alarm clock for your program

• Others in D&C Chapter 6.2
Setting up event handlers

SIGACTION(2) Linux Programmer's Manual SIGACTION(2)

NAME

sigaction – examine and change a signal action

SYNOPSIS

#include <signal.h>

int sigaction(int signum, const struct sigaction *act,
               struct sigaction *oldact);

struct sigaction {
    void   (*sa_handler)(int);
    void   (*sa_sigaction)(int, siginfo_t *, void *);
    sigset_t   sa_mask;
    int   sa_flags;
    void   (*sa_restorer)(void);
};

Function to handle event
How to handle other events during the handling of this event
Rest of fields can be ignored
So how do we use this?

• Define the event handling function
  – void myfun(int signal);

• Associate that function with the signal you want to handle
  – sigaction() call
What does ‘mask’ mean?

• Signals arrive unpredictably and asynchronously
  – Get a SIGINT or SIGTERM for example

• What happens if, if your handler for SIGINT, another SIGINT comes?

• Can simplify our handler by ‘masking’ signals during our event handler
  – Helper functions provided (e.g., sigfillset(…))
Signals and networking APIs

- What happens to a blocking call when an event comes in?
  - Control transferred to event handler
  - When control returned, the blocking stops, and an error code is returned

- `Recv()`
  - Might return fewer bytes than requested, or EINTR return code if no bytes received

- `Send()`
  - Might send fewer bytes than requested or EINTR if no bytes sent
Demo 2

Setting a receiver timeout on our echo server using an event handler, closing the connection after 3 seconds of client inactivity
Demo 2 overview

1. Define our event handler
2. Setup the event handler with sigaction()
3. Change our recv() code to check for EINTR return code
   1. If so, close the connection
Alternative timeout mechanism

• Instead of a SIGALRM event handler, can set timeouts using a socket option

• Why might this be a better option in some cases?
Demo 3

Setting a receiver timeout on our echo server with socket options, closing the connection after 3 seconds of client inactivity
Demo 3 overview

1. Set the timeout value with setsockopt()
2. Change recv() calls to check for the timeout
   – EWOULDBLOCK instead of EINTR
   – Otherwise the same as the alarm example
Project 1 Q&A