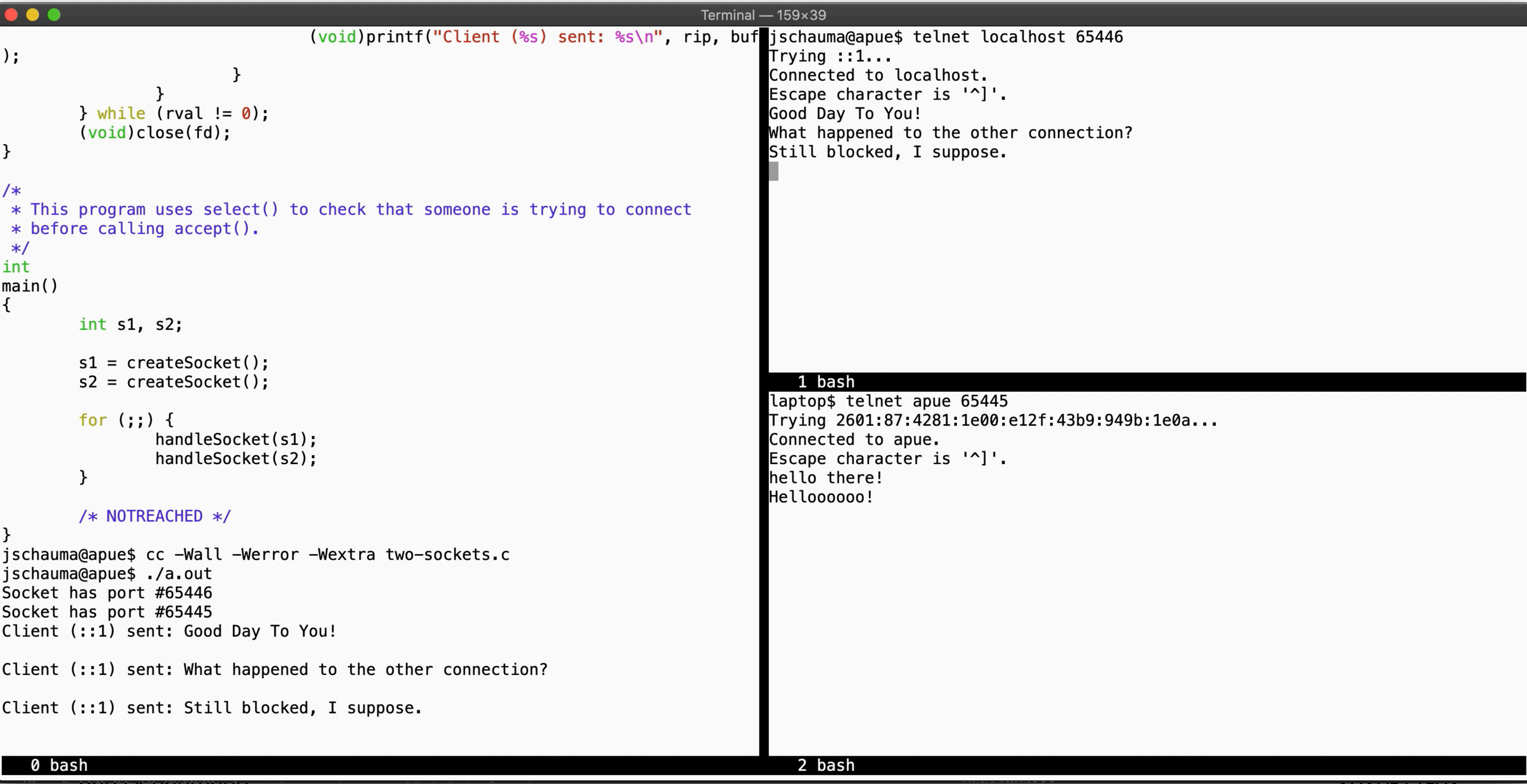
Advanced Programming in the UNIX Environment

Week 09, Segment 5:
I/O Multiplexing

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/


Terminal — 159×39
(void)printf("Client (%s) sent: %s\n", rip, buf
);
}
} while (rval != 0);
(void)close(fd);
}
/*
* This program uses select() to check that someone is trying to connect
* before calling accept().
*/
int
main()
{
int s1, s2;
s1 = createSocket();
s2 = createSocket();
for (;;) {
handleSocket(s1);
handleSocket(s2);
}
/* NOTREACHED */
}
jschauma@apue$ cc -Wall -Werror -Wextra two-sockets.c
jschauma@apue$ ./a.out
Socket has port #65446
Socket has port #65445
Client (::1) sent: Good Day To You!
Client (::1) sent: What happened to the other connection?
Client (::1) sent: Still blocked, I suppose.
0 bash
jschauma@apue$ telnet localhost 65446
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Good Day To You!
What happened to the other connection?
Still blocked, I suppose.
1 bash
laptop$ telnet apue 65445
Trying 2601:87:4281:1e00:e12f:43b9:949b:1e0a...
Connected to apue.
Escape character is '^]'.
hello there!
Hellooooooo!
2 bash

I/O Multiplexing

When handling I/O on multiple file descriptors, we have the following options:

- *blocking mode*: open one fd, block (possibly forever), then test the next fd
- *fork* and use one process for each, communicate using signals or other IPC
- *non-blocking mode*: open one fd, immediately get results, open next fd, immediately get results, sleep for some time
- *asynchronous I/O*: get notified by the kernel when either fd is ready for I/O

I/O Multiplexing

Instead of *blocking forever* (undesirable), using *non-blocking mode* (busy-polling is inefficient) or using *asynchronous I/O* (somewhat limited), we can:

- build a set of file descriptors we're interested in
- call a function that will return if any of the file descriptors are ready for I/O (or a timeout has elapsed)

select(2)

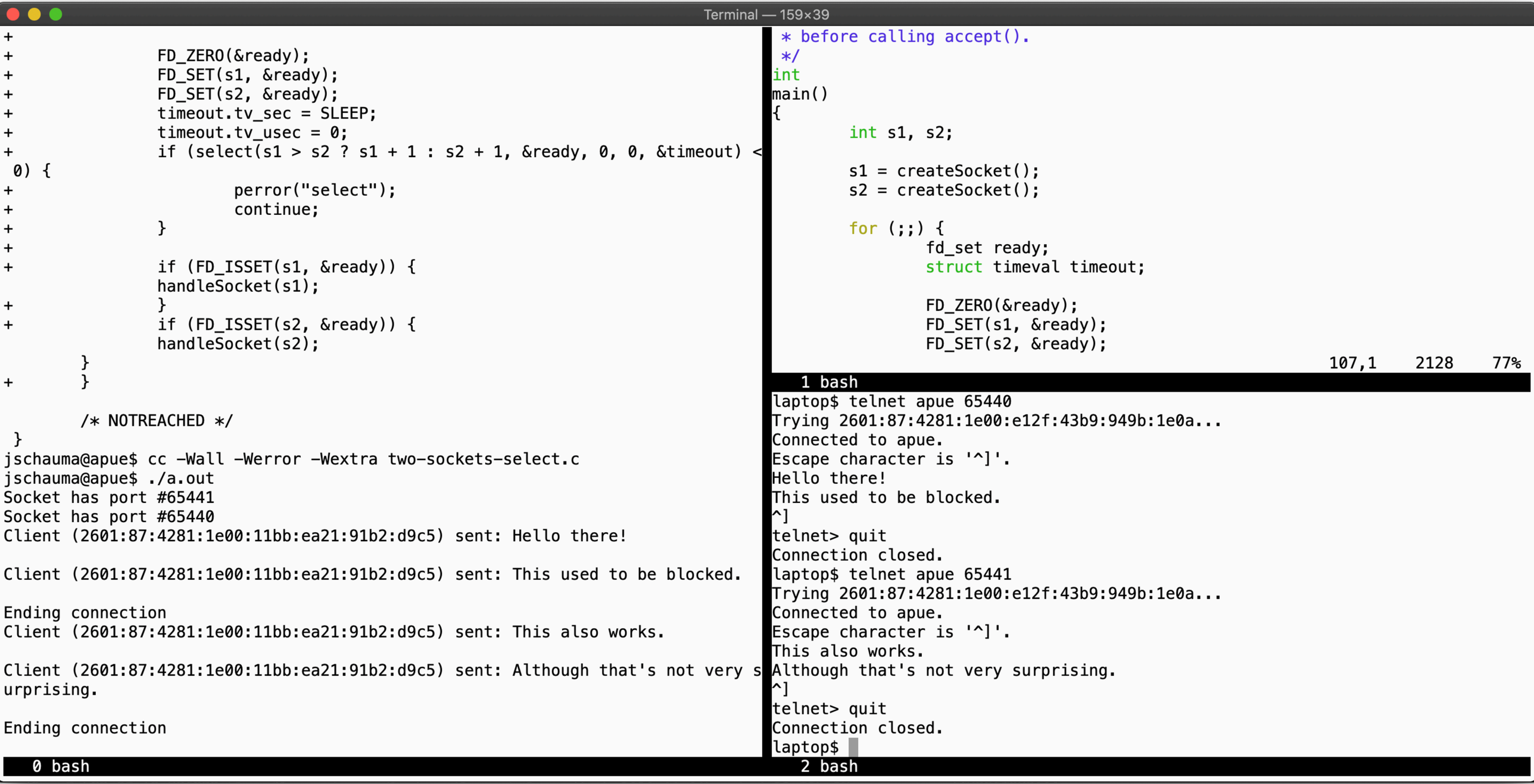
```
#include <sys/select.h>
int select(int nfds, fd_set * restrict readfds, fd_set * restrict writefds,
                     fd_set * restrict exceptfds, struct timeval * restrict timeout);
                                                  Returns: # of descriptors if ok, -1 otherwise
```

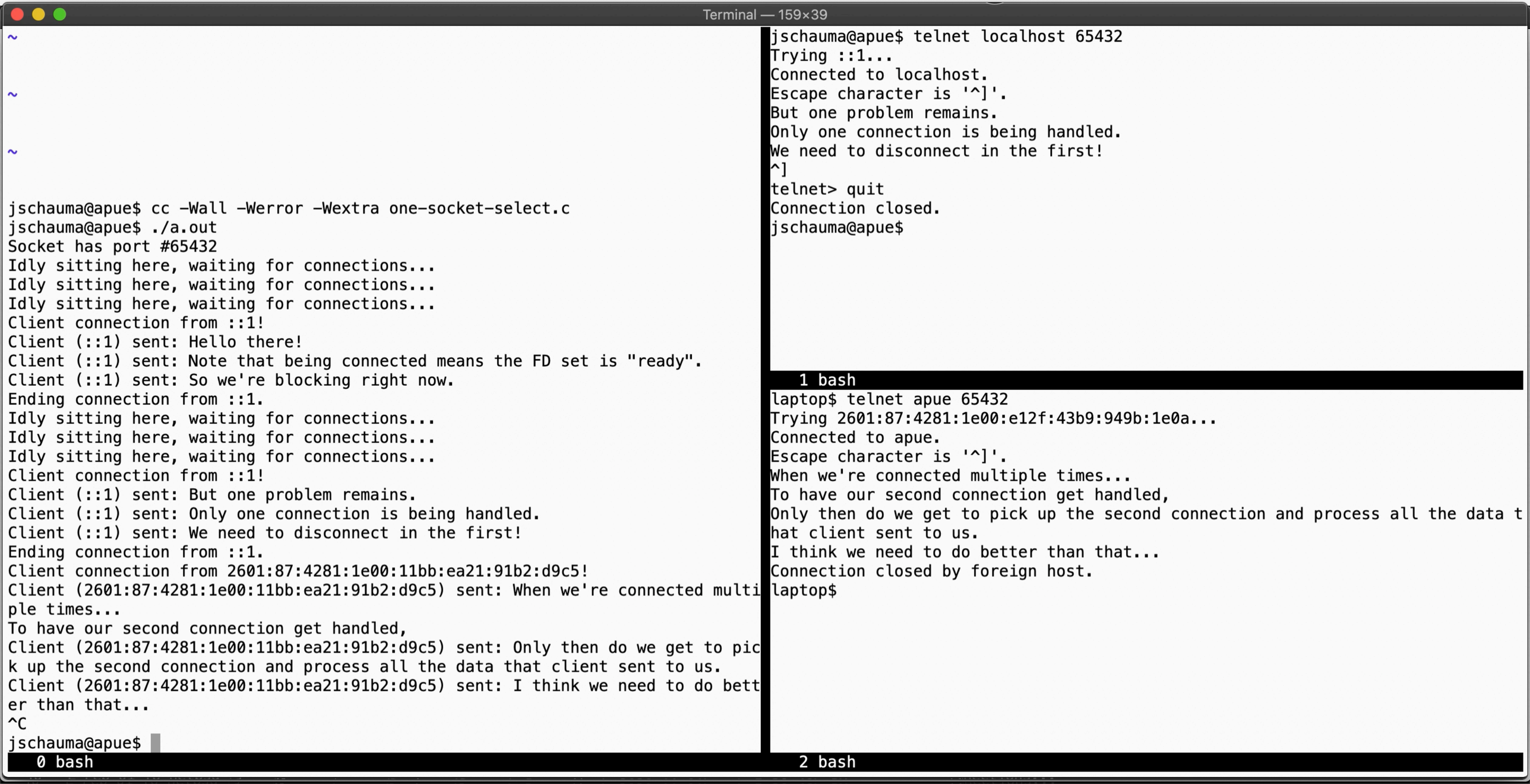
select(2) tells us both the total count of descriptors that are ready as well as which ones are ready:

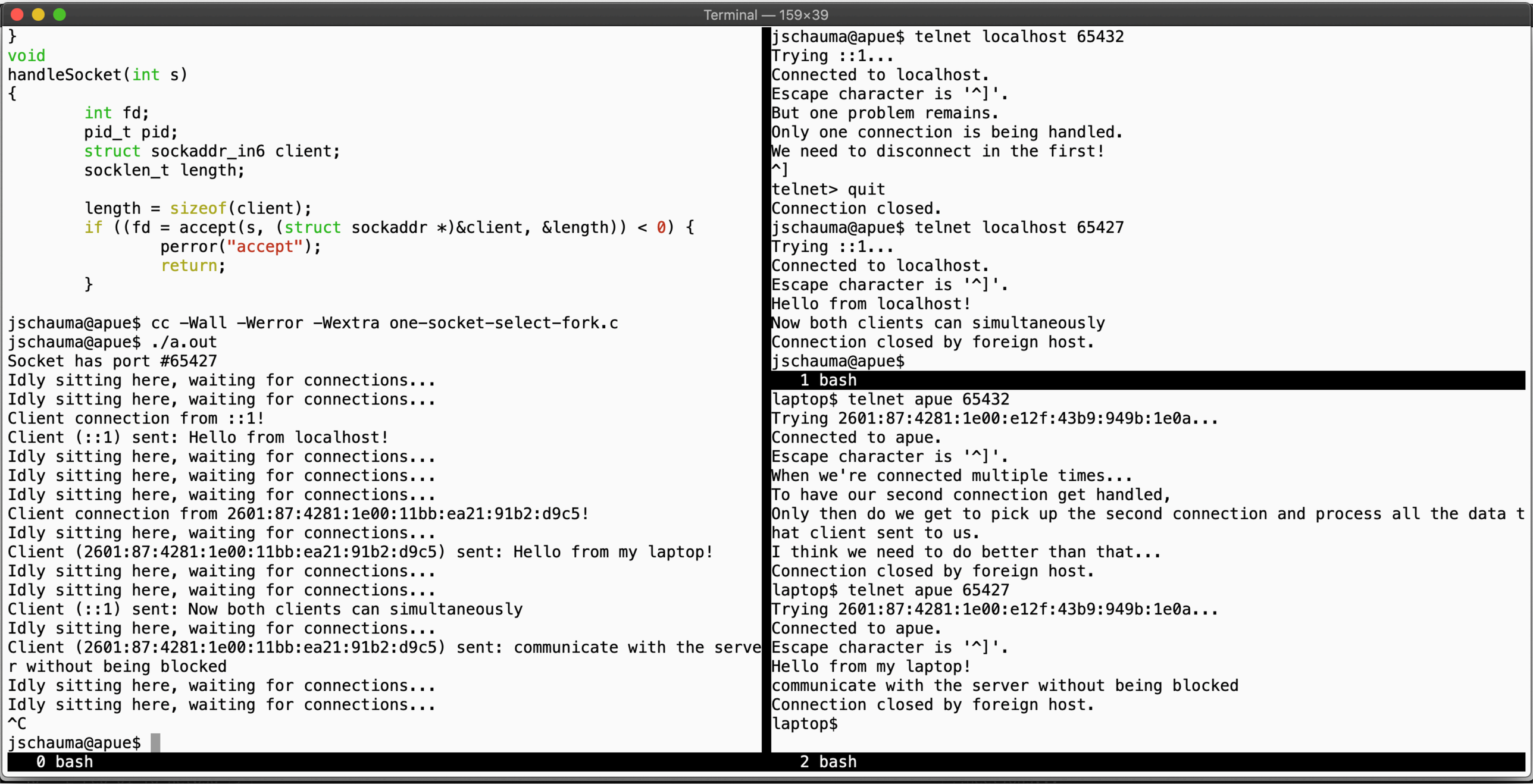
- nfds: which descriptors we're interested in
- readfds, writefds, exceptfds: what conditions we're interested in
- timeout: how long we want to wait
  - tvptr == NULL: wait forever
  - tvptr->tvsec == tvptr->tvusec == 0: don't wait at all

I/O Multiplexing

- file descriptor sets are manipulated using the `FD*` functions/macros
- `readfds`/`writefds` sets indicate readiness for read/write; `exceptfds` indicates an exception condition (for example OOB data, certain terminal events)
- `EOF` means ready for read - `read(2)` will just return 0 (as usual)
- `pselect(2)` provides finer-grained timeout control; allows you to specify a signal mask (original signal mask is restored upon return)


Terminal — 159×39
+
+ FD_ZERO(&ready);
+ FD_SET(s1, &ready);
+ FD_SET(s2, &ready);
+ timeout.tv_sec = SLEEP;
+ timeout.tv_usec = 0;
+ if (select(s1 > s2 ? s1 + 1 : s2 + 1, &ready, 0, 0, &timeout) <
0) {
+ perror("select");
+ continue;
+ }
+
+ if (FD_ISSET(s1, &ready)) {
handleSocket(s1);
+ }
+ if (FD_ISSET(s2, &ready)) {
handleSocket(s2);
}
+ }
/* NOTREACHED */
}
jschauma@apue$ cc -Wall -Werror -Wextra two-sockets-select.c
jschauma@apue$ ./a.out
Socket has port #65441
Socket has port #65440
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: Hello there!
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: This used to be blocked.
Ending connection
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: This also works.
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: Although that's not very s
urprising.
Ending connection
0 bash
* before calling accept().
*/
int
main()
{
int s1, s2;
s1 = createSocket();
s2 = createSocket();
for (;;) {
fd_set ready;
struct timeval timeout;
FD_ZERO(&ready);
FD_SET(s1, &ready);
FD_SET(s2, &ready);
107,1 2128 77%
1 bash
laptop$ telnet apue 65440
Trying 2601:87:4281:1e00:e12f:43b9:949b:1e0a...
Connected to apue.
Escape character is '^]'.
Hello there!
This used to be blocked.
^]
telnet> quit
Connection closed.
laptop$ telnet apue 65441
Trying 2601:87:4281:1e00:e12f:43b9:949b:1e0a...
Connected to apue.
Escape character is '^]'.
This also works.
Although that's not very surprising.
^]
telnet> quit
Connection closed.
laptop$
2 bash


Terminal — 159×39
~
~
~
jschauma@apue$ cc -Wall -Werror -Wextra one-socket-select.c
jschauma@apue$ ./a.out
Socket has port #65432
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Client connection from ::1!
Client (::1) sent: Hello there!
Client (::1) sent: Note that being connected means the FD set is "ready".
Client (::1) sent: So we're blocking right now.
Ending connection from ::1.
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Client connection from ::1!
Client (::1) sent: But one problem remains.
Client (::1) sent: Only one connection is being handled.
Client (::1) sent: We need to disconnect in the first!
Ending connection from ::1.
Client connection from 2601:87:4281:1e00:11bb:ea21:91b2:d9c5!
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: When we're connected multi
ple times...
To have our second connection get handled,
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: Only then do we get to pic
k up the second connection and process all the data that client sent to us.
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: I think we need to do bett
er than that...
^C
jschauma@apue$
0 bash
jschauma@apue$ telnet localhost 65432
Trying ::1...
Connected to localhost.
Escape character is '^]'.
But one problem remains.
Only one connection is being handled.
We need to disconnect in the first!
^]
telnet> quit
Connection closed.
jschauma@apue$
1 bash
laptop$ telnet apue 65432
Trying 2601:87:4281:1e00:e12f:43b9:949b:1e0a...
Connected to apue.
Escape character is '^]'.
When we're connected multiple times...
To have our second connection get handled,
Only then do we get to pick up the second connection and process all the data t
hat client sent to us.
I think we need to do better than that...
Connection closed by foreign host.
laptop$
2 bash


Terminal — 159×39
}
void
handleSocket(int s)
{
int fd;
pid_t pid;
struct sockaddr_in6 client;
socklen_t length;
length = sizeof(client);
if ((fd = accept(s, (struct sockaddr *)&client, &length)) < 0) {
perror("accept");
return;
}
jschauma@apue$ cc -Wall -Werror -Wextra one-socket-select-fork.c
jschauma@apue$ ./a.out
Socket has port #65427
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Client connection from ::1!
Client (::1) sent: Hello from localhost!
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Client connection from 2601:87:4281:1e00:11bb:ea21:91b2:d9c5!
Idly sitting here, waiting for connections...
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: Hello from my laptop!
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
Client (::1) sent: Now both clients can simultaneously
Idly sitting here, waiting for connections...
Client (2601:87:4281:1e00:11bb:ea21:91b2:d9c5) sent: communicate with the serve
r without being blocked
Idly sitting here, waiting for connections...
Idly sitting here, waiting for connections...
^C
jschauma@apue$
0 bash
jschauma@apue$ telnet localhost 65432
Trying ::1...
Connected to localhost.
Escape character is '^]'.
But one problem remains.
Only one connection is being handled.
We need to disconnect in the first!
^]
telnet> quit
Connection closed.
jschauma@apue$ telnet localhost 65427
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Hello from localhost!
Now both clients can simultaneously
Connection closed by foreign host.
jschauma@apue$
1 bash
laptop$ telnet apue 65432
Trying 2601:87:4281:1e00:e12f:43b9:949b:1e0a...
Connected to apue.
Escape character is '^]'.
When we're connected multiple times...
To have our second connection get handled,
Only then do we get to pick up the second connection and process all the data t
hat client sent to us.
I think we need to do better than that...
Connection closed by foreign host.
laptop$ telnet apue 65427
Trying 2601:87:4281:1e00:e12f:43b9:949b:1e0a...
Connected to apue.
Escape character is '^]'.
Hello from my laptop!
communicate with the server without being blocked
Connection closed by foreign host.
laptop$
2 bash

I/O Multiplexing

- use `select(2)` to check multiple file descriptors for I/O readiness
- avoid blocking individual file descriptors by handling I/O in a separate process or thread
- alternative or similar interfaces:
  - `poll(2)`
  - `epoll(2)` (Linux)
  - `kqueue(2)` (*BSD)
  - https://libevent.org/ / http://software.schmorp.de/pkg/libev.html
- See also: http://www.kegel.com/c10k.html