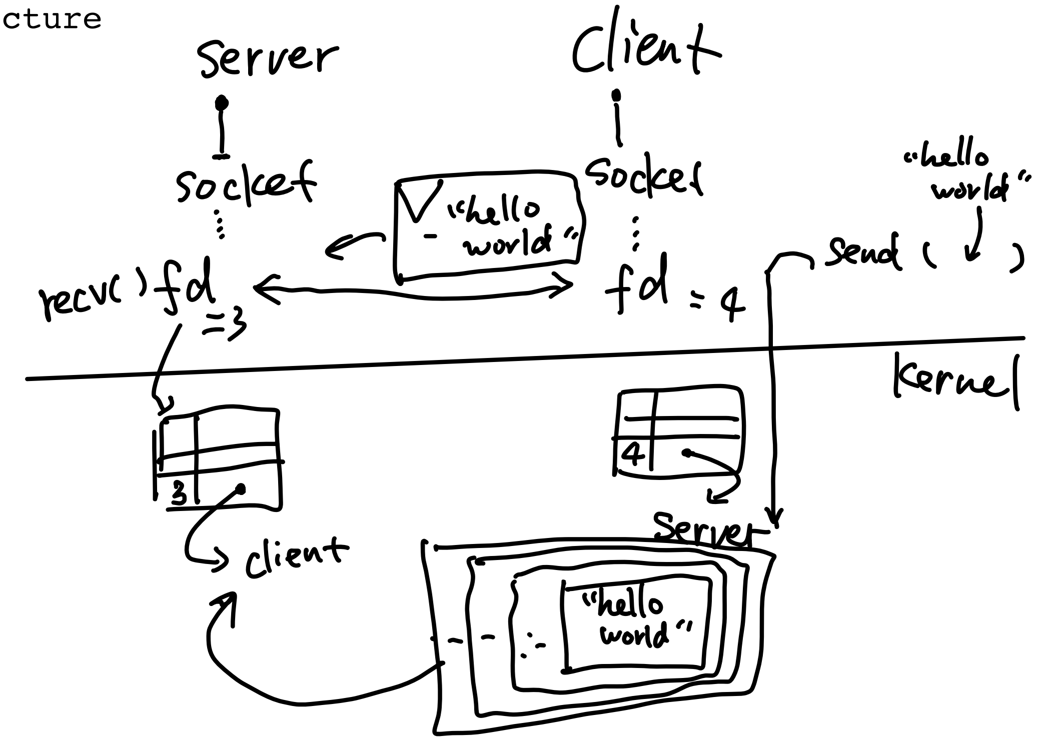
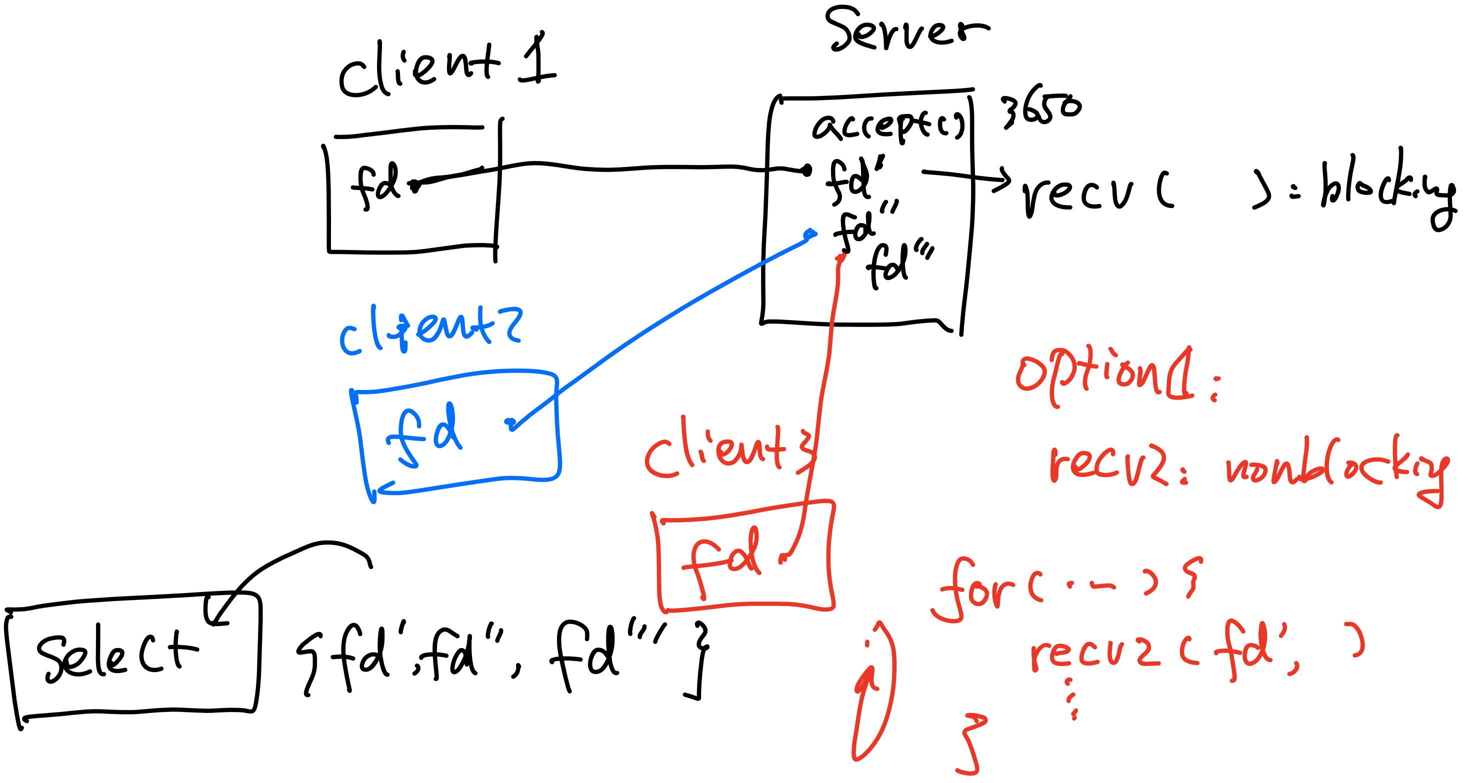
Week 13.a
CS3650
04/01 2024
https://naizhengtan.github.io/24spring/

1. socket programming
2. select syscall
3. Device drivers
4. I/O architecture



multiplexing IO



```
CS3650: socket programming

// 1. This is a simple example of a client sending "hello world!"
//    to a server.

     [server]                   [client]
        |                          |
     fd = socket(...)           fd = socket(...)
     bind(fd,...)                  |
     listen(fd,...)                |
        |                          |
 new_fd = accept(fd,...)      +--connect(fd,...)
        |                    /     |
        |<-------------------+     |
        |------------------------->|
        |                          |
     new_fd <====================> fd


// 2. Server code

  // assuming the following helper function will fill in the "struct sockaddr"
  void init_sockaddr(struct sockaddr *in_addr, const char *ip, int port);

  // return a file descriptor
  int listen_socket() {
    int fd = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr addr;
    init_sockaddr(&addr, NULL, 3650 /*port number*/);

    bind(fd, &addr, sizeof(addr);

    listen(fd, 128);

    struct sockaddr tmp;
    socklen_t addr_size = sizeof(tmp);
    int new_fd = accept(fd, &tmp, &addr_size);

    close(fd); // stop accepting more connections

    return new_fd;
  }

  int main() {
    int new_fd = listen_socket();

    char buf[1024] = {0};
    recv(new_fd, &buf, 1024, 0); // receiving data
    printf("%s\n", buf);

    close(new_fd);
  }

```

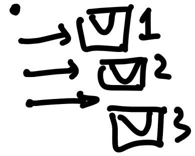
IP
TCP
ANY:3650
IP,TCP
ANY,3650
fd: IP,TCP, ANY,3650,
listen
wait
"hello world"

172.28.83.21

10.107.166.238

UDP

client Server



```

// 3. Client code

int connect_socket() {
    int fd = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr serv_addr;
    init_sockaddr(&serv_addr, "127.0.0.1" /* ip */, 3650 /* port */);

    connect(fd, &serv_addr, sizeof(serv_addr));

    return fd;
}

int main() {
    int fd = connect_socket();

    char *hello = “hello world!";
    send(fd, hello, strlen(hello), 0); // sending data

    close(fd);
}


```

IP
TCP (★)
Not a well-known Function
localhost
≈ (IP.TCP, localhost:3650)

4. Socket programming interfaces:

a) socket, send, and recv

- int **socket**(int domain, int type, int protocol);

  socket() creates an endpoint for communication and returns a descriptor.

- ssize_t **send**(int socket, const void *buffer, size_t length, int flags);

  send a message from a socket

- ssize_t **recv**(int socket, void *buffer, size_t length, int flags);

  receive a message from a socket

b) bind, listen, and accept (server side)

- int **bind**(int socket, const struct sockaddr *address, socklen_t address_len);

  bind() assigns a name to an unnamed socket.

- int **listen**(int socket, int backlog);

  listen for connections on a socket

fd

- int **accept**(int socket, struct sockaddr *restrict address,
  socklen_t *restrict address_len);

  accept a connection on a socket

c) connect (client side)

fd ?

- int **connect**(int socket, const struct sockaddr *address, socklen_t address_len);

  initiate a connection on a socket

CS3650: select, synchronous I/O multiplexing

1. select interfaces

fd∈[0, 5-1]
   =
   4

{1, 5, 7}

a) select

≤5

```
* int select(int nfds, fd_set *restrict readfds,
             fd_set *restrict writefds, fd_set *restrict errorfds,
             struct timeval *restrict timeout);
```

select() examines the I/O descriptor sets whose addresses are passed in readfds, writefds, and errorfds to see if some of their descriptors are ready for reading, are ready for writing, or have an exceptional condition pending, respectively.

b) fd_set manipulation

```
* FD_ZERO(fd_set *set);               Clear all entries from the set.
* FD_SET(int fd, fd_set *set);        Add fd to the set.
* FD_CLR(int fd, fd_set *set);        Remove fd from the set.
* FD_ISSET(int fd, fd_set *set);      Return true if fd is in the set.
```

2. An example – a chat server

```
// Below is a code snippet using select()

int fds[2] = {0, 0};
fds[0] = ...     accept()      // socket connection 1
fds[1] = ...     accept()      // socket connection 2

fd_set readfds; // local var

while(1) {
    FD_ZERO(&readfds);   // ☆   {fds[0], fds[1]}
    for (int i=0; i<2; i++) {
        FD_SET(fds[i], &readfds);
    }
                max(fds[0], fds[1]);
    int maxfd = ...             // Q: what is the maxfd?
             [0, maxfd]
    select(maxfd+1, &readfds, NULL, NULL, NULL);

    for (int i=0; i<2; i++) {
        if (FD_ISSET(fds[i], &readfds)) {
            print(fds[i], ...); // print msg received
        }
    }
}
... // wrap up and exit
```

CS3650: I/O and device driver

1. An examples of I/O instructions:

Setting the cursor position

The code below is also excerpted from WeensyOS's k-hardware.c. It uses I/O instructions to set a blinking cursor in the upper left of the screen.

```
// console_show_cursor(cpos)
//   Move the console cursor to position ‘cpos‘,
//   which should be between 0 and 80 * 25.

void console_show_cursor(int cpos) {
  if (cpos < 0 || cpos > CONSOLE_ROWS * CONSOLE_COLUMNS)
    cpos = 0;

  outb(0x3D4, 14); // Command 14 = upper byte of position
  outb(0x3D5, 0 / 256); // row 0
  outb(0x3D4, 15); // Command 15 = lower byte of position
  outb(0x3D5, 0 % 256); // column 0

}
```