Week 10 1

CSC209 Fall 2023

Dr. Demetres (dee) Kostas

November 21, 2023

Announcements

- A4 is out!
  - uses filters from A3!
  - more transferring of bitmaps!
  - start reading
    - you'll have all the tools
    - by the end of the week
- It is due Dec. 6th

Multi-plexing IPC

- with *one* pipe (or any file-descriptor)
  - we can sit around
  - waiting for data to read or write
- how do we wait for multiple pipes?
  - while wanting to write to
    - a file on a slow USB drive?
  - while waiting for network/internet bytes
- we *could* just do this one-by-one

But we don't have to...

- rather than manually managing
  - all of the `fd`'s associated
  - with each of these read/write targets
    - previous slide
- there is a system call
  - designed to manage this

`select()`

- cycles through *sets* of `fd`'s
- checking (the following sets) for
  - **read**
    - (is there data to be read?)
  - **write**
    - (is there space to write?)
  - or ***exceptional*** conditions
    - (rare, won't be tested)

Recall that `fd`'s are numbers

- effectively they are indexes
  - in the processes table
  - mapping id to pointers
    - in a master table of "files"
    - that include devices and sockets!
- `fd_set` is just a set of bits
  - enough of them to represent
    - indexes of all the `fd`'s
    - as individual bits

system call and API

```
struct timeval{
    time_t tv_sec; //seconds
    suseconds_t tv_usec; // microseconds
};

int select(int maxfd, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
           struct timeval *timeout);
```

- Helper macros:
  - `FD_ZERO(fd_set \*fdset)`, `FD_SET(int fd, fd_set \*fdset)`, `FD_CLR(int fd, fd_set \*fdset)`, `FD_ISSET(int fd, fd_set \*fdset)`

General approach

- create a set (each, r/w/e)
  - memory for `fd_set`, then `FD_ZERO` it
    - then `FD_SET` each fd of interest
- call `select` (once for all sets)
  - check for errors
- then use `FD_ISSET` to check
  - if `select` has said
    - that any fd is ready for use
  - from each set used

A note on `maxfd`

- ultimately, `select` is just a for loop
  - through all the indexes
  - and stops at this one
- it's not too smart…
  - check out `poll` or `epoll`

    - (not part of 209, but better)

- there are many hacks to improve
  - e.g. using low fd numbers

So what's the advantage of `select`?

- instead of just looping yourself
  - and using other system calls
    - which switch back to the system
- you relinquish control
  - to the system
  - your process doesn't need scheduling
    - to then just ask the system stuff
      - and wait again

At this point

- we want to be comfortable
- with just seeing new system APIs
- so let's dive into a worksheet
  - that is *like* select
    - but requires readings docs

WORKSHEET

`select.pdf`

Sockets intro

- many of you are probably familiar
  - with the idea of IP addresses
    - "unique" addresses
    - for individual computers

- the real question is...
  - how do we, as system-level
    - coders, designers, etc.
  - connect processes over the internet?

The first piece to the puzzle

- is understanding the abstractions
  - the scheme imposed by
  - the design of network *protocols*
- there are a few *layers*
  - analogy: consider the *layers*
    - between python libraries
      - and the underlying system calls
      - that are eventually made...

Network system layers (broadly)

- the first format is the connection scheme
- IP format
  - we will use IP v4
    - which has already run out
      - of unique addresses
      - see IP v6
    - it is held together with hacks
      - (network class for more info)

Sending packets over IP

- this layer uses the IP connection
  - to send data in... packets
    - with automatic/abstracted error checking

    - we will use TCP
    - which is guaranteed to
        - inform us if there was an error!
- UDP is the other most common protocol
    - which does not guarantee this

Network layer

- each machine has an address
    - expressed as 4 8-bit numbers
        - e.g. `192.168.1.1`
- at that address, there are 16bits worth
    - of port values
    - numbered: `0 -> ~66000`

sockets

- Send data in *packets*
    - using a port on **each end**
        - of a connection
- through a networking protocol
- which requires an address
    - and sometimes a specific port
    - sometimes any random port
        - will do (and happens automatically)