CS241 #38 – System Concepts & Review. UDP sendto/recvfrom tips.

> Warmup: Is the following code threadsafe?

```
void* func(void*ptr) {
  sleep(1);
  puts((char*)ptr);
  return ptr;
}

void run() {
  pthread_t t1;
  char data[8];
  strcpy(data,"1234567");
  pthread_create(&t1, NULL, func, data+4);
  puts("p_created called!");
  pthread_join(t1, NULL);
}
```

> Virtual Memory Concepts:

What is working set and thrashing?

How large is the working set of your mmap binary search 'advert price' process if your data file is 4GB and search requests are random? Always the same?

What is demand paging?

> Brain Teaser

Why does calling calloc(200000) actually take the same time as malloc(200000) !?

> Context Switch

What is a context switch?

Why are context switches “expensive”?

> EPoll

Give a network example why you might call epoll_ctl with EPOLL_CTL_DEL, after epoll_wait returns.

> Traffic Filtering

Name two ways an Internet provider can prevent BitTorrent traffic

> HTTP Protocols

Explain why do web pages display faster if the client and server use HTTP/1.1 instead of HTTP/1.0

Give two performance advantages of HTTP2.0 over HTTP1.1

> Domain Name System

What is DNS? How does it work?

> UDP using sendto and recvfrom.

How do I make a simple UDP client and server?

Client Server

```
ssize_t sendto(int fd,void* buf, size_t len,
                int flags,
                struct sockaddr *dest,
                socklen_t dest_len);


ssize_t recvfrom(int fd,void* buf,size_t len,
        int flags,
        struct sockaddr * address,
        socklen_t * address_len);
```

struct sockaddr
struct sockaddr_in
struct sockaddr_in6
struct sockaddr_storage

> Protip: Use connect and send if you want to send data to the same endpoint (host & port).

```
send(int socket, void *buff, size_t length, int flags);
```

... So what is different using connect compared to using connect with a TCP socket?

> Underhanded C Challenge

```
#define N (20)
int admin, debug;
int histogram[N];

static int hash(char* str) {
  int c, h = 0; // sdbm hash
  while (c = *str++)
     h = c + (h << 6) + (h << 16) - h;
  return h;
}

int main(int argc, char**argv){

  while(argc>1) {
    char*word= argv[ --argc];
    int h = hash(word);
    histogram[ (h<0?-h:h) % N ] ++;
  }

  if(admin || debug) puts("Admin/Debug");
  return;
}
```