Networking
Networking

• Core concepts
  – 5-layer internet model
  – Protocol
  – IP Address, Ports
  – Packets and layering

• Differences between TCP and UDP

• Sockets
New connection for each client

TCP Server / Client

Server

socket()
bind()
listen()
accept()
recv()
send()
close()

Client

socket()
connect()
send()
recv()
close()

Blocks until new connection

New connection for each client
UDP Server / Client

Server

socket()
bind()
recvfrom()
sendto()
close()

Client

socket()
bind() optional
sendto()
recvfrom()
close()
Threads
Threads

• Goals for doing multithreading
  – Parallelism
  – Concurrency

• Why processes are sometimes not ideal task containers

• What is a thread
  – A “lightweight process”
  – State: bare minimum to allow computation
    • Stack + CPU registers
Thread State

**Per process items**
- Address space (page table)
- Heap space
- Global variables
- Open files
- Child processes
- Signals & handlers

**Per thread items**
- Stack
- Registers($EIP, $ESP, …)
Threads

• Threads Pros / Cons
• Thread safe code
• User-level vs. Kernel-level threading
  – Key differences
  – Pros / Cons
  – Issues with user-level threading and solutions
Pthreads
Pthreads

• What are Pthreads
• Pthreads API
  – pthread_create
    • Creates a new thread
    • Prototype for start routine and reason behind it
  – pthread_yield
    • Yields voluntarily, but does not guarantee ordering
  – pthread_join
    • Wait for a thread to complete and get return value
  – pthread_exit
    • Exits the thread and returns value
Be Prepared to Write Code

```c
#include <stdio.h>
#include <pthread.h>

void *do_stuff(void *p) {
    printf("Hello from thread %d\n", *(int *)p);
}

int main() {
    pthread_t thread;
    int id, arg1, arg2;

    arg1 = 1;
    id = pthread_create(&thread, NULL, do_stuff, (void *)&arg1);
pthread_join(thread, NULL);
    arg2 = 2;
    do_stuff((void *)&arg2);

    return 0;
}
```
Be Prepared to Write Code

```c
struct Value { ... };
void* thread_func(void *p) {
    struct Value* val1 = malloc(sizeof(struct Value));
    ...
    pthread_exit(val1);
}
int main() {
    struct Value *val2;
    ...
    pthread_join(thread, &val2);
    // val1 == val2
}
```
Synchronization
Synchronization

• Core concepts
  – Data races
  – Critical Section
  – Atomicity

• Synchronization constructs
  – Join
  – Mutex
  – Condition variable
Synchronization

• Deadlocks
  – What are the necessary conditions
  – How to avoid using locks

• Synchronization API
  – Pthread_join
  – Pthread_mutex_lock
  – Pthread_mutex_unlock
  – Pthread_cond_wait
    • Why do have a mutex as the second parameter?
  – Pthread_cond_signal

• Various pitfalls
void enqueue(int value)
{
    A[tail] = value;
    tail++;
}

- How would you synchronize the above?

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
void enqueue(int value)
{
    pthread_mutex_lock(&mutex);
    A[tail] = value;
    tail++;
    pthread_mutex_unlock(&mutex);
}
Practical C Issues
Practical C Issues

• How to write generic macros
• The #if directive
  – What does it do?
  – When is it useful?
• Multifile development
  – Why is it important
  – Need for proper scoping
  – How to divide your project into multiple files
    (How to write header files, why they are needed)
Practical C Issues

• Makefiles
  – Rules, targets, dependencies, actions
  – Dependency tree
  – Automatic variables, pattern matching
  – How to read / write Makefiles
Makefiles

malloctest: mymalloc.o mallocdriver.o
    gcc -o $@ $^

%.o: %.c
    gcc -c $< -o $@

mymalloc.o: mymalloc.h
mallocdriver.o: mymalloc.h

clean:
    rm -f *.o malloctest
Makefiles

malloctest

mymalloc.o
mymalloc.c
mymalloc.h

mallocdriver.o
mallocdriver.c
Makefiles

• Build from scratch

```
thoth $ ls
Makefile mallocdrv.c mymalloc.c mymalloc.h
thoth $ make
gcc -c mallocdrv.c –o mallocdrv.o
gcc -c mymalloc.c –o mymalloc.o
gcc -o malloctest mymalloc.o mallocdrv.o
thoth $ make
make: `malloctest' is up to date.
```

• Partial build after modifying mymalloc.c

```
thoth $ touch mymalloc.c
thoth $ make
gcc -c mymalloc.c
gcc -o malloctest mymalloc.o mallocdrv.o
```