CS4400 *Alex Clemmer, u0458675*

Problem Set 13

1

A server's specific services are available on specific ports by convention. Daytime is on port 13, ftp on port 21, smtp on 25, and so on. It is reasonable to expect servers to follow this convention because then clients know exactly where and how to connect to get services they want. In the client's case, there is no real need for this convention, since the client largely dictates how it interacts with other machines on a network.

2

The client builds the socket descriptor by calling `socket`. It then uses `connect` to establish a connection with the server. Pretty simple process.

The server is a bit more complicated. It starts by building a socket descriptor by calling `socket`. Then it calls `bind` to associate a socket address with the descriptor. It converts the descriptor to a listening socket with `listen`, and then blocks with `accept`, waiting for the client to connect.

After the server accepts the connection, the client returns from `connect`, and the server returns from `accept`.

12.16

Easy:

```
void *thread(void *vargp)
{
    printf("Hello, world!\n");
    return NULL;
}

int main(int argc, char *argv[])
{
    int t, i;

    t = atoi(argv[1]);
    pthread_t tid;

    for (i = 0; i < t; i++) {
        pthread_create(&tid, NULL, thread, NULL);
        pthread_join(tid, NULL);
}
    exit(0);
}
```

12.17

1. Because `main` terminates before the other thread can print.
2. You can replace it with `pthread_join`, or `wait`. There are advantages and disadvantages to picking each, but suffice it to say, in `pthread_join`, you can really only wait for a specific thread, where with unix's `wait`, you can wait for an arbitrary thread.