Network Programming

Michio Honda

Networking abstraction

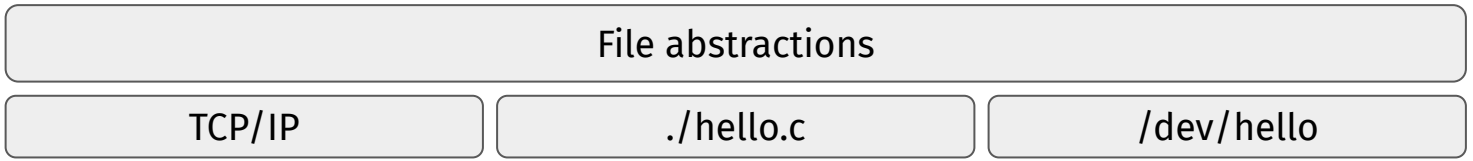
```
fd = socket(AF_INET,
     SOCK_STREAM, IPPROTO_TCP);
...
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

```
fd = open(“./hello.c”, O_RDWR);
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

```
fd = open(“/dev/hello”,
          O_RDWR);
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

user

kernel



Networking abstraction

```
fd = socket(AF_INET,
     SOCK_STREAM, IPPROTO_TCP);
for which service
active or passive
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

```
fd = open(“./hello.c”, O_RDWR);
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

```
fd = open(“/dev/hello”,
                O_RDWR);
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

Networking abstraction

```
fd = socket(AF_INET,
     SOCK_STREAM, IPPROTO_TCP);
bind(fd, …);
listen(fd, …);
newfd = accept(fd, …);
write(newfd, “hello”, 6);
read(newfd, buf, sizeof(buf));
```

```
fd = open(“./hello.c”, O_RDWR);
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

```
fd = open(“/dev/hello”,
                O_RDWR);
write(fd, “hello”, 6);
read(fd, buf, sizeof(buf));
```

Asynchronous I/O

- `read()` blocks until something arrives (from the network)
- What if we want to serve multiple clients?

```
fd = socket(AF_INET,
     SOCK_STREAM, IPPROTO_TCP);
bind(fd, …);
listen(fd, …);
newfd = accept(fd, …);
read(newfd, buf, sizeof(buf));
// wait until the client sends
// a request
write(newfd, buf, 6);
```

```
fd = socket(AF_INET,
     SOCK_STREAM, IPPROTO_TCP);
bind(fd, …);
listen(fd, …);
epfd = epoll_create1(,);
epoll_ctl(epfd, EPOLL_CTL_ADD, fd);
for (;;) {
  int nfds = epoll_wait(epfd, &evts,);
  // wait until any client sends a request
  int i=0;
  for (; i< nfds; i++) {
    if evts[i].fd == fd) {
      newfd = accept(fd,);
      epoll_ctl(epfd, EPOLL_CTL_ADD, newfd);
    } else {
      read(evts[i].fd, buf, sizeof(buf));
      write(evts[i].fd, buf, sizeof(buf));
    }
  }
}
```