

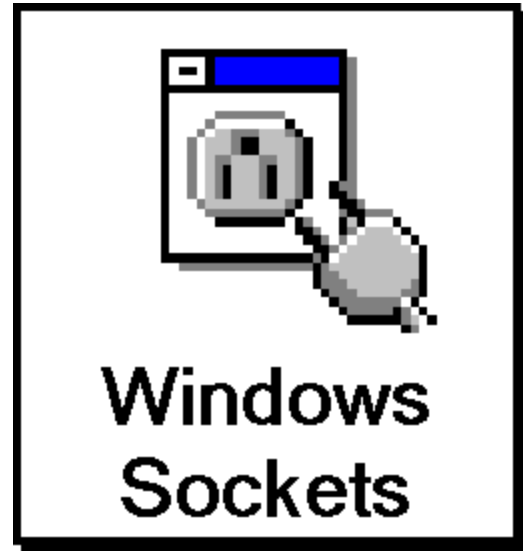
Sockets and concurrency

15-441 Fall 2010, Recitation 3

Your TAs

Recitation Outline

- **Sockets Reloaded**
- Checkpoint 2
- Project 1 Q&A
- General Q&A



(A little throwback to Windows 3.1)

Revisiting Network Sockets

- Sockets, used for network IPC
 - How does this differ from IPC with two local processes?
- What did we need to setup for a socket?
 - Address of endpoint
 - Port associated with application
- How does reading/writing differ from file I/O?
 - `read()` not guaranteed to provide all requested data
 - Blocking much more of an issue

Socket Review

- Sockets are treated just like files:
 - `socket()` returns a file descriptor
 - `read()` & `write()`: same interface as for files
- Types of sockets
 - Blocking (synchronous)
 - Non-blocking
 - Polling
 - **Asynchronous notification**

Handling Connections

- Examples of network applications that rely on handling concurrent connections?
 - *Web servers, file servers (AFS), IRC, AIM, etc...*
- How can an application handle multiple connections?
 - *Threading & select()*
- Which do you have to use for **all** projects?
 - ***select()***

Select()

- How select works
 - Allows you to monitor multiple sockets
 - API for monitoring multiple file descriptors
- Two main sets of descriptors
 - Read & write descriptors
- Use bit-array *fd_set* to monitor
 - *fd_set readfds, writefds;*
- First set all to 0
 - *FD_ZERO(&readfds); FD_ZERO(&writefds);*

Telling Select What To Monitor

- After zeroing out, set FDs to be monitored
 - Original FD returned by `socket()` for incoming connections
 - All currently connected client's FDs
- So, assuming *sfd=socket(...);*
 - *FD_SET(sfd, &readfds);*
 - Loop through client FDs:
FD_SET(client[i].sfd, &readfds)

Using select()

- Select checks from bit 0 in the bit-array up until `maxfd`
 - Initially: *maxfd=sfd*;
 - Looping through clients: *if(client[i].fds>maxfd) ...*
- Now, call it!
 - *select(maxfd+1, &readfds, NULL, NULL, NULL);*
- **IMPORTANT:** `select()` overwrites `&readfds` with new bit-array, representing which file descriptors are *ready*

After Select() Returns

- Check the new bit-array
- What if select() sets the bit for *sfd*?
 - You have a new client
 - *if (FD_ISSET(sfd, &readfds)) { accept_client(...); }*
- What if select() sets the bit for a client FD?
 - Data is ready to be read
 - Must loop through all of your client FDs with FD_ISSET, if it's set: *handle_input(...)*

I've handled the whole array...

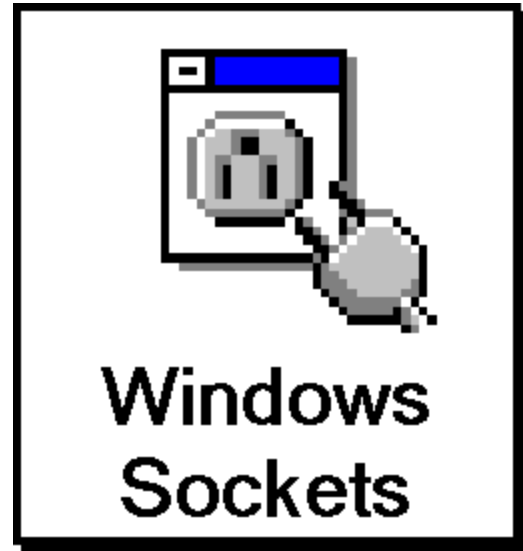
- After you've gone through and used `FD_ISSET()`:
 - Start all over!
 - `FD_ZERO(&readfds);`
 - `FD_SET(sfd, &readfds);`
 - ...
- In other words, create a `while(1)` around this and loop and loop!
 - Remember: keep the `while(1)` loop thin

More on Accepting New Clients

- Whenever *FDISSET(sfd, &readfds)*, you have a new client
- You use *accept()* on *sfd* and save the returned file descriptor as your new client's FD
- Store all of these FDs and make sure to set them in *readfds* before you call *select()*

Recitation Outline

- Sockets Reloaded
- **Checkpoint 2**
- Project 1 Q&A
- General Q&A



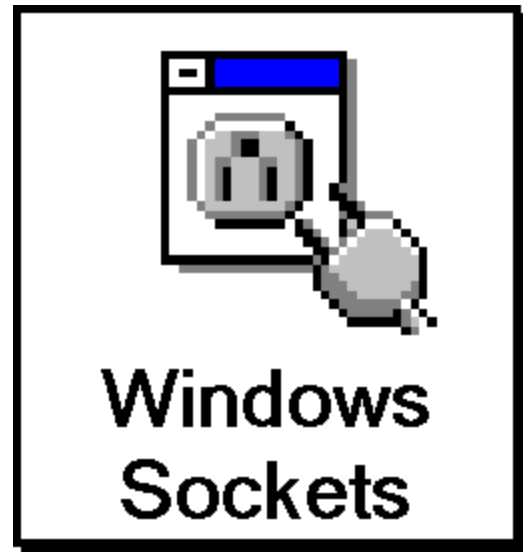
(A little throwback to Windows 3.1)

Checkpoint 2

- Available under “Assignments”
- What you need to do:
 - Handle **concurrent connections** (this lecture!)
 - **Echo back** any commands to the client
 - **Avoid blocking**, one client should not be able to stall server by sending a partial command
 - Handle a **malicious client**
 - Sends you long data with no ‘\n’ ... don’t crash!
 - Handle gracefully (preferably truncating)

Recitation Outline

- Sockets Reloaded
- Checkpoint 2
- **Project 1 Q&A**
- **General Q&A**



(A little throwback to Windows 3.1)