Lecture 20: IP Servers
A Concurrent Server

- We have been writing servers that can only communicate with a single client. There are a couple of techniques that can be used to handle ________ clients.
- In one method we can fork a new child process for each client that connects.
- In a second method we can use select() or poll() to monitor multiple file descriptors for activity simultaneously.

Lecture 20: IP Servers
A Concurrent Server

- Skeleton code for a __________ server ...
  listenfd = socket( ... );
  bind( listenfd, ... );
  listen(listenfd, ... );
  while ( 1 ) {
    connfd = accept( listenfd, ... );
    if( pid = fork() == 0 ) {
      close( listenfd );
      doit(connfd); // or exec(...)
      close( connfd );
      exit(0);
    }
    close( connfd );
  }

Lecture 20: IP Servers
I/O Multiplexing

- I/O ________ allows us to use a single server process to handle all connections. In order to do this properly we need to monitor multiple file descriptors for activity.
- I/O multiplexing and is provided by the select() or poll() routines. (They provide similar functionality.) I/O multiplexing may be used with any type of file descriptor (not just sockets).

Lecture 20: IP Servers
File Descriptor Sets

- To use select() we must first define one or more file descriptor ______. A descriptor set is usually implemented as an array of integers and each bit in an integer corresponds to a particular file descriptor. Four macros are used to manipulate descriptor sets:
  void FD_ZERO(fd_set *fdset);
  void FD_SET(int fd, fd_set *fdset);
  void FD_CLR(int fd, fd_set *fd_set);
  int FD_ISSET(int fd, fd_set *fd_set);
Lecture 20: IP Servers
The select() Routine

- The `select()` routine prototype:
  ```c
  int select(int maxfdp1,
             fd_set *readset,
             fd_set *writeset,
             fd_set *exceptset,
             struct timeval *timeout);
  ```

- `maxfdp1` is the maximum file descriptor number to be tested (plus 1). It exists for
  __________ (so that the kernel does not have to test all 1024 possible descriptors).

Lecture 20: IP Servers
The select() Routine

- The `timeout` argument allows us to specify how long to wait until a descriptor is ready:
  ```c
  struct timeval {
    long tv_set;  /* secs to wait */
    long tv_usec; /* usecs to wait */
  }
  ```

- `timeout` may be NULL (typical), `select()` will then wait forever. If the members of the
  struct are zero, `select()` returns immediately (useful for __________).

Lecture 20: IP Servers
In Class Exercise

- Examine `slit_server.cpp` and `slit_client.cpp` as examples of handling
  multiple clients using `select()`.

- Modify `slit_server.cpp` so that the call to `select` (and the for loop) uses a variable
  holding the maximum fd number plus one instead of `FD_SETSIZE`. This number should
  be updated every time a client connects or disconnects.