Purpose: This assignment is designed to familiarize you with the simplest form of preallocation.

THE PREALLOCATED BROWSER SERVER

You will build a browser server that uses preallocation. You server will preallocate 3 (total) servers, each of which is willing to do an accept. Three means one original plus two you forked.

Start with a copy of the `browserd.c` you submitted. Call your new verion `pbrowserd.c`.

THE BROWSER CLIENT

Use the `browserc.c` you submitted. Do not change it.

Testing: Try to start 4 clients. The first three should start fine; the fourth should hang until one of the first three exits.

Submit: A print out of the server. In addition, the source code for the server must be placed in your home directory in a file named `pbrowserd.c`.

Discussion:

You will need to rearrange some of the code. In particular, the forks and associated switch should be done during the startup code; not inside the while loop of the main program. The while loop of the main program needs to become that for an iterative server; an unconditional accept followed by a call to the service procedure. The service procedure does not need to be changed.

I selected 3 to because it is small enough to be tested. A better server would preallocated and terminate servers much like a web server, but that is too much code for a short assignment.

The behavior of this server reveals some items about the `connect` call. In particular, the `connect` is only losely ties to the `accept` in that the `connect` of the client will complete before the server does an `accept`; as long as: (1) the server has done a `listen` and (2) the queue length (`QLEN`) hasn't been exceeded. In effect, the client anticipates that an `accept` will be done. Hence, the fourth client will "hang" in the sense that the commands (`l`, `g`, `c`) will not work, but it will get to the prompt after the connect. However, with a queue length of 5, the ninth client will see `connect` return a error.