System Programming HW2 Report
B09901027 賀崇恩

1. What is busy waiting? How do you avoid busy waiting in this assignment? Is it possible to have busy waiting even with select()/poll() ?

   Busy waiting means to call read/write as much as possible in the process for I/O (for socket I/O in this case), without suspending the process. In my assignment, I used the select() system call to avoid busy waiting, which blocks the process until select detects a I/O event. In practice, the timeout in select() is set to NULL. If the timeout of select() is set to 0, then the process runs like busy waiting.

2. What is starvation? Is it possible for a request to encounter starvation in this assignment? Please explain.

   Starvation means that a client waits forever for response as long as there are some other requests blocking the server, and appropriate multiplexing is not implemented to deal with this case. In my program, since the file descriptors are processed one by one in the while(1) loop, if a request is written to the socket file descriptor, it will be detected by select() and be processed in the next while(1) loop.

3. How do you handle a file's consistency when multiple requests within a process access to it simultaneously?

   For read processes, since the file cannot be manipulated, there's no need to handle file consistency. For write processes, I use a flag array, whose index is the id (input id - 902000) to mark which student's data is in a written process (just like a local lock handler). Those who request to write to those student's data with the corresponding flag on will received "lock" and disconnect from the server.

4. How do you handle a file's consistency when different process access to it simultaneously?

   I used the fcntl() system call to handle it. To be specific, if a process acquires a read lock successfully, it should handle a local reference count to count how many clients are currently reading the data. The process should release the read lock once the local read reference count is decreased to 0. For write locks, it is easier to

handle between processes, because since an exclusive write lock is acquired, any process trying to acquire read lock or write lock will fail.