Lab Exercise

- **Make chatting program using message queues**
  - Program gets user and receiver ID via STDIN when it starts
  - Users can receive messages while entering their message via fork
  - When the user receives a regular message from another user, print out the message
  - Once it reads the message, send back an ack message with read time to the other user
  - There should be no wait time between normal message and ack message (add **IPC_NOWAIT** flags to the ***msgget*** function)
  - Guaranteed to have two users running the program at the same time (i.e., cases where there is only one user are ignored) with only one user typing at a given time.
  - Skeleton code on iCampus

Lab Exercise (cont.)

- **Print Format**
  - User $RCV_ID:\t$USER_MSG // regular message
  - User $RCV_ID read message at $READ_TIME // ack message
  - Use localtime function + strftime (time.h) to match $READ_TIME format

```
char buf[SIZE];
time_t t = time(NULL);
struct tm tm = *localtime(&t);
strftime(ack.timestamp, SIZE, "%Y-%m-%d %H:%M:%S", &tm);
```

Lab Exercise (cont.)

- **Example**

```
User ID: 1
Receiver ID: 3
User 1: Hi!
User 3 read message at 2023-10-30 03:14:15
User 1: How are you?
User 3 read message at 2023-10-30 03:14:19
User 3: I am fine
User 3: And you?
User 1: quit
```

```
User ID: 3
Receiver ID: 1
User 1: Hi!
User 1: How are you?
User 3: I am fine
User 1 read message at 2023-10-30 03:15:03
User 3: And you?
User 1 read message at 2023-10-30 03:15:06
User 3: quit
```

Exercise submission

- **Submit your source code and Makefile**
  - via **iCampus**
  - Makefile should generate ./w9 executable
  - Bundle *source code* and *Makefile* with tar command
    - » *tar.gz* format

    $ tar cvzf [student_id].tar.gz week9
  - We'll grade your submission with **make**
    - » If compilation fails, your points for this exercise will be zero