**Multi-threading Procedure**

1. All lock handling and distribution is handled by the mt\_open() function, which takes in an absolute file path and acquires the appropriate lock for that file. To prevent race conditions, mt\_open() is generally called before every call to the access() function, to check file existence/accessibility.
2. Locks are only obtained and released in client-facing user functions, such as create\_new\_user(), follow(), write\_woot(), load\_woots(), etc.
3. Internal functions used to implement the client-facing functions (create\_file(), flush\_user(), fids\_decrease()) have their lock handling done in the client-facing functions that call them.
4. Any functions that modify database contents (create\_new\_user(), follow(), write\_woot()) follow a strict lock ordering in where they first acquire the lock for their user's file, and then they acquire the lock(s) for whatever files they'll need to access or modify.
5. Any functions that simply READ database contents (read\_user(), load\_following(), load\_woots()) simply acquire the locks for the specific file(s) that they need.
   * a) Thus functions that modify database contents on some given file for a user do not block functions that simply want to read content on some other file for the user/any user in the same user file.
   * b) While read and write locks are not implemented on individual given files, (for the sake of reduced complexity) this allows for a similar performance improvement of read locks for operations on a given block of users that share the same user file.
6. While this seems to create race conditions, updates only occur in modifying calls like follow() within a single file at a time, and thus must follow the appropriate lock procedure. Thus the only race condition is a reading function obtaining data that will soon be eventually updated, and thus providing the client with stale data. However, stale data is an acceptable race condition as the odds of a modifying thread and a reading thread on the same user occurring this way are so low, that it's more likely stale data will be returned anyhow due to network latency differences of the update being sent to the server, and the read occurring just before the update happens to arrive.

**Special Cases:**

Two functions have their lock handling done in handle\_php itself: already\_following(), and flush\_user(). These two are an exception as they are non-client-facing functions that are directly called by the php. (already\_following() to update the "Follow" button appropriately, flush\_user() to update after deactivation.)