README Group 4

1 CONTENTS

- ServerMain.cpp: Server code which listens for new connections and spawns threads to handle them
- TheadMain.cpp: Thread code which handles specific client requests
- profanity.txt: List of words to filter from web pages
- cache.db: SQLite3 database which tracks cached and blacklisted web pages
- Cached pages: Local copies of web pages created by the server as it fetches them from the web

2 COMPILING THE SERVER

```
g++ -std=gnu++0x ServerMain.cpp ThreadMain.cpp -o server -lsqlite3 –lpthread
```

3 RUNNING THE SERVER

To start: `./server <port-number>`

To end: `<Cntl+C>`

Each time a client requests a webpage, the server will print that webpage's URL to the console. Cached webpages will be saved directly into the server's working directory.

Only root web pages may be requested (e.g. www.unt.edu).

4 MANAGING THE CACHE

All cache and blacklist data is kept in the cache.db database. This database comes preloaded with three blacklisted sites: youtube, facebook, and hulu. Cached webpages are saved directly into the server's working directory.

To view current cache data: `sqlite3 cache.db "SELECT * FROM Cache"`

To clear cached pages: `sqlite3 cache.db "DELETE FROM Cache WHERE date != 'BLACKLISTED'"`

To clear blacklisted pages: `sqlite3 cache.db "DELETE FROM Cache WHERE date == 'BLACKLISTED'"`

To clear all cache data: `sqlite3 cache.db "DELETE FROM Cache"`

To add a page to the blacklist: `sqlite3 cache.db "INSERT OR REPLACE INTO Cache VALUES ('<webpage-URL>', 'BLACKLISTED')"`

5 FILTERING PROFANITY

Profanity.txt contains a new line delineated list of words to filter from webpages. Because our server only returns root webpages, and because these rarely contain profanity, profanity.txt is preloaded with words likely to be found on a university's homepage.

Modifications to profanity.txt will not affect webpages that have already been cached.