SWE3019: SOFTWARE EXPERIMENT 3 (Spring 2013)

Final Project

Due: June 16, 11:59 PM

1. Introduction

√ 프로젝트를 통해 multi-thread 방식의 서버를 구현하여 thread를 이용한 프로그래밍과 Concurrent 프로그래밍에 익숙해지도록 한다.

2. Overview

√ Midterm 프로젝트에서 했던 Keyword Searcher의 연장으로 제출한 서버가 멀티 쓰레드 방식으로 동작하도록 작성하는 것이다.

√ Mutex를 사용하여 멀티 쓰레드 환경에서 공용으로 사용하는 변수에 대해 Concurrency control을 제공한다.

√ Code를 Optimize하여 빠른 시간 내에 user가 원하는 정보를 출력할 수 있도록 한다.

3. Specification

√ Server를 다중 쓰레드 방식으로 구현하여 PA4에서와 마찬가지로 다수의 클라이언트 접속에 대하여 키워드 검색을 처리할 수 있도록 한다.

√ Server에서 수행하는 키워드 검색 조건은 Midterm project에서 수행했던 것과 동일하다.

☞ 지난 과제의 Client가 보내고 받는 형식은 코드를 통해 제공된다.

☞ Server/Client 간 주고받는 양식이 다를 경우 채점이 되지 않는다.

√ 출력은 서버에서 클라이언트로 완성된 문자열을 전송하는 것으로 하며, 양식은 아래와 같다.

☞ 등수. 입력받은 검색어 [포함되는 장절의 수][검색 횟수]
등수. 입력받은 검색어 [포함되는 장절의 수][검색 횟수]...

☞ 등수의 경우 같은 경우 동일 등수로 표시하며 다음 등수는 앞의 합을 제외한 다음 등수부터 시작한다.

☞ 출력 예시

```
1. "god * created" [2][9]
2. god in [82][8]
3. created god the he [3][7]
3. he god created [3][7]
3. the created god he [3][7]
6. "created * god * he" [2][6]
6. "god created he" [1][6]
6. the god in he created [3][6]
9. "created * god" [4][5]
9. "created * he * god * the" [0][5]
```

√ Server에서는 **Genesis.txt, psalms.txt**에 대해 Index를 작성한다.

☞ 해당 .txt 파일은 atschool.eduweb.co.uk/SBS777/bible/text/ 에서 다운받을 수 있다.

√ Server는 최근 검색 키워드를 저장하도록 하며, 검색 횟수를 기준으로 상위 10개에 대하여 검색 키워드 검색 횟수를 보관한다.

√ Client에서 "!history"로 쿼리를 서버로 전송하면, Server는 최근 검색 키워드 10개를 내림차순으로 정렬하여 Client로 전송한다.

☞ 최근 검색 키워드가 10개 이하일 경우 서버에 저장된 키워드에 대해서만 검색 횟수를 기준으로 내림차순으로 정렬한다.

√ 이번 과제에서는 수행시간을 기준으로 과제를 채점한다.

☞ 과제의 수행 시간은 서버(http://sys.skku.edu)의 시간을 기준으로 하며, 제출 마감 전까지 실시간으로 확인할 수 있다.

☞ 서버에서 과도한 작업을 시도할 경우 서버가 다운될 수 있으므로 본인의 컴퓨터에서 충분한 테스트를 거친 뒤 서버에 제출한다.

4. Given Client Codes

√ 원활한 과제 진행을 위해 Client Template 코드를 제공한다.

√ 제공되는 파일

File Name	Description
client_t.c	다중 Thread로 서버에 접속하여 쿼리를 보내는 클라이언트
Makfile	makefile

√ 사용 방법

```
$ ./client [IP] [Port] [Thread 수]
```

√ 확인 방법

```
proshb@proshb-vm:~/swe3019/Untitled Folder$ ./client_t 127.0.0.1 12345 100
sizeof(ptstrQueryEx) = 20
QueryCount = 5
 --- Server History ---
> !history
1. "created he god" [49]
2. "created god in the he" [48]
2. "god * he * in * the * created" [48]
4. "created * he * the * god" [46]
5. god created he the [45]
6. "he * created * the * god" [44]
7. "created god" [42]
8. "in * he * god * created * the" [32]
9. "he god in the created" [30]
9. "the * created * he * god * in" [30]

 ------- Answer -------
1. "created he god" [49]
2. "created god in the he" [48]
2. "god * he * in * the * created" [48]
4. "created * he * the * god" [46]
5. god created he the [45]
6. "he * created * the * god" [44]
7. "created god" [42]
8. "in * he * god * created * the" [32]
9. "he god in the created" [30]
9. "the * created * he * god * in" [30]
Throughput: 2.325974 (s)
```

☞ 100개의 Thread를 생성하여 서버에 요청한 결과

☞ 실제 서버에서 테스트 되는 것은 더 복잡할 수 있다. 충분한 테스트를 거친후 제출할 것.

5. Restriction

√ 과제는 본인이 직접 설치한 리눅스 환경에서 수행한다.

√ 윈도우 환경에서 Visual-Studio 등을 이용해 프로그램을 작성한 후, 테스트 서버에서 컴파일이 되지 않거나 동작하지 않는 경우 과제 제출로 인정하지 않는다.

√ 구현이 완료된 경우 http://sys.skku.edu에 과제를 제출하고 수행하여 결과를 확인한다.

6. Hand in instructions

√ 과제 제출 시 본인이 작성한 Server의 Makefile, Source code(코드 및 헤더파일)를 모두 제출한다.

☞ **Make 후 실행 생성되는 실행파일의 이름은 "server"로 한다.**

√ 작성한 프로그램 코드 상단에 이름과 학번을 주석으로 표기한다.

√ 과제 및 보고서는 제출 시 "**.tar.gz / .zip**" 형식으로 압축하여 서버(http://sys.skku.edu)에서 테스트한다.

☞ 코드 제출 시 make clean을 하고 폴더가 아닌 파일을 압축하여 제출한다.

√ 구현에 관한 내용을 담은 보고서를 별도로 제출한다. 보고서에는 프로그램의 구조를 나타낸 Flow chart 또는 그림을 필수로 넣는다. 보고서 형식은 가능하면 PDF 포맷으로 제출한다.

☞ 과제 보고서에 컴파일하고 실행한 화면을 Capture하여 추가한다.

7. Logistics

√ 과제 제출 시간은 http://sys.skku.edu 서버시간을 기준으로 하며 **과제 제출 시간이후의 submit은 미제출**로 한다.

√ 과제에 대한 의논은 함께 할 수 있으나, 프로그램 소스코드 작성은 스스로 해야 한다.

√ 다른 사람의 과제를 copy한 경우, 두 사람 모두 0점 처리한다. 인터넷 등에서 찾은 소스 코드를 그대로 copy한 경우에도 0점 처리한다. 두 번 이상 이와 같은 이유로 0점 처리된 경우 F 학점을 받을 수 있다.