Aufgabenblatt 6

Aufgabe – Implementierung eines Webclients.

Entwickeln Sie ein Programm, dass aus einer Steuerdatei Namen von Webservern und Seitenadressen auf diesen Servern ausliest und dann die Server im Netz kontaktiert und einen GET request an die Seiten sendet.

Anforderungen:

1. Das Programm soll aus mehreren Threads bestehen:
   - Ein Thread (reader-Thread) liest die Zeilen aus einer Steuerdatei und schreibt die gelesenen Daten in eine Queue mit einer (zur Compilezeit) konfigurierbaren Größe.
   - Eine (zur Compilezeit) konfigurierbare Anzahl von Threads (client-Threads) liest die Daten aus der Queue und kontaktiert die Web-Server.
2. Die Zugriffe auf die Queue müssen gegeneinander geschützt sein.
3. Die client-Threads bekommen als Parameter eine Instanznummer übergeben, die sie z.B. bei Debug-Ausgaben eindeutig identifiziert.
4. Die Antworten der Web-Server werden in Dateien gespeichert. Die Dateinamen sollen durchnummeriert sein. Sind also *n* Einträge in der Steuerdatei, sollen die Antworten in *file0.html*, *file1.html* …*file<n-1>.html* gespeichert werden.
5. In der Steuerdatei liegen die Daten in der Form:
   `<server> <seitenadresse>`.
   Pro Server ist eine Zeile vorgesehen. Beispiel:
   `www.heise.de /`
   `www.uni-hannover.de /en/`
   Der Name der Steuerdatei muss dem Programm über die Kommandozeile übergeben werden.
6. Bestimmen Sie näherungsweise die Laufzeit bei unterschiedlicher Anzahl von client-Threads. (Z. B. mit der Funktion `difftime( )` in main() .)
7. Das Programm darf keine globalen Variablen verwenden.

Einige Randbedingungen:

- Um die Webserver auszulesen steht die Funktion
  ```
  errorValue askServer(const char *address,
                       const char* page,
                       const char* filename);
  ```
  in der Datei *socket.c* zur Verfügung.
  Inkludieren Sie dazu die Header-Datei *msocket.h* in Ihr Programm.
  Die Funktion ist unter Linux (gcc 4.1.0), Solaris und cygwin getestet.
  Um die Funktionalität zu nutzen, muss der Linker auf einigen Systemen die Libraries socket und nsl einbinden. Siehe dazu den Kommentarblock in *socket.c*.
- Es steht eine Steuerdatei (*testSites.txt*) mit einigen Adressen zu Verfügung. Erweitern Sie gegebenenfalls diese Datei um weitere Server und Seiten.
- Als Beispielimplementierung ist eine Lösung des Erzeuger/Verbraucher Problems mit Pthreads der Aufgabe beigefügt (*pc.c*). Diese Beispielimplementierung kann als Rahmen für Ihr Programm genommen werden und um die spezifischen Anforderungen der Aufgabe erweitert werden.
- Falls Sie die Aufgabe lieber objektorientiert in C++ lösen möchten, tun Sie es.
- Für diese Aufgabe sind zwei Wochen eingeplant. Zum ersten Testattermin sollten Sie mindestens die Zugriffe auf die Queue und ein Konzept zur Synchronisierung vorlegen können.