Design Document: httpserver.cpp

Cristian C. Castillo
CruzID: ccarri11
Baubak Saadat
CruzID: bsaadat

1.0 Introduction

1.1 Goals and objectives

1.2 Assumptions

2.0 Data design

2.1 Internal data structure

2.1 Global data structure

2.2 Temporary data structure

2.3 Global variables

- *pthread_mutex_t:* A type used to initialized statically
- *pthread_cond_t:* Used for appropriate functions for waiting and later of process continuation. This variable enables threads to suspend execution, voluntarily cease to keep the processor until a condition presents itself as true. To avoid race conditions that were created by one thread, we associated pthread_cond_t with a mutex. While that one thread is preparing to wait, an additional thread may

signal the condition before the first thread actually waits on the resulting deadlock. Hence, the thread will wait for a signal that is never sent and any mutex can be used as there is no link amongst mutex and the condition variable.

2.4 Local variables

- *pthread_t:* The data type used to uniquely identify a thread. It is returned by pthread_create() and used by the application in function calls that require a thread identifier. In our program this was used to help create the dispatcher of threads (array).

- *pthread_t_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr):*
A function that initialises the mutex referenced by mutex with attributes specified by attr. If attr is NULL, the default mutex attributes are used; the effect is the same as passing the address of a default mutex attributes object. Uon successful initialization, the state of the mutex becomes initialized and unlocked. Any attempts to initialize an already initialized mutex results in undefined behavior.

- *pthread_mutex_lock(pthread_mutex_t *mutex):*
The mutex object referenced by mutex is locking by calling pthread_mutex_lock(). If the mutex is already locked, the calling thread blocks until the mutex becomes available. This operation returns with the mutex object referencedf by mutex in the locked state with the calling thread as its owner.

- *pthread_mutex_unlock(pthread_mutex_t *mutex):* This function releases the mutex object referenced by mutex. The manner in which a mutex is released is dependent upon the mutex's type attribute. If there are threads blocked on the mutex object referenced by mutex when this function is invoked, this results in the mutex becoming available, the scheduling policy is used to determine which thread shall acquire the mutex.

- *pthread_cond_signal(pthread_cond_t *cond):* Unblocks at least one of the threads that are blocked on the specified condition variable cond (if any threads are blocked on cond).

- *pthread_mutex_unlock(pthread_mutex_t *mutex):* This function releases the mutex object referenced by mutex. The manner in which a mutex is released is dependent upon the mutex's type attribute. If there are threads blocked on the mutex object referenced by mutex when this function is invoked, this results in the mutex becoming available, the scheduling policy is used to determine which thread shall acquire the mutex.

- *pthread_cond_signal(pthread_cond_t *cond):* Unblocks at least one of the threads that are blocked on the specified condition variable cond (if any threads are blocked on cond).

- *pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex):* Used to block on a conditional variable. They are called with mutex locked by the calling thread or undefined behavior will result. These functions atomically release mutex and cause the calling thread to block on the condition variable cond.

2.5 Functions

2.6 Helper Functions

3.0 Architectural and Component-level design

3.1 System Structure

3.1.1 Architectural diagram

Visit the design flow-chart: Multi-threading-Flow-Chart

3.2 Description for Component n

3.2.1 Processing narrative (PSPEC) for component n

3.2.2 Component n interface description

3.2.3 Component n processing detail

3.3 Dynamic Behavior for Component n

3.3.1 Interaction Diagrams

3.2 Description for Component n

4.0 User interface design

4.1 Description of the user interface

4.1.1 Screen images

4.1.2 Objects and actions

4.2 Interface design rules

4.3 Components available

5.0 Restrictions, limitations, and constraints

6.0 Testing Issues

6.1 Classes of tests

6.2 Expected software response

6.3 Performance bounds

6.4 Identification of critical components

7.0 Pseudocode