

**CS 471 – Operating Systems – Fall 2004
Group Project – Prof. Daniel A. Menascé**

This project will give you an idea of how a Web server works and what kind of synchronization problems are involved in designing a Web server. Consider the Web server illustrated in Fig. 1.

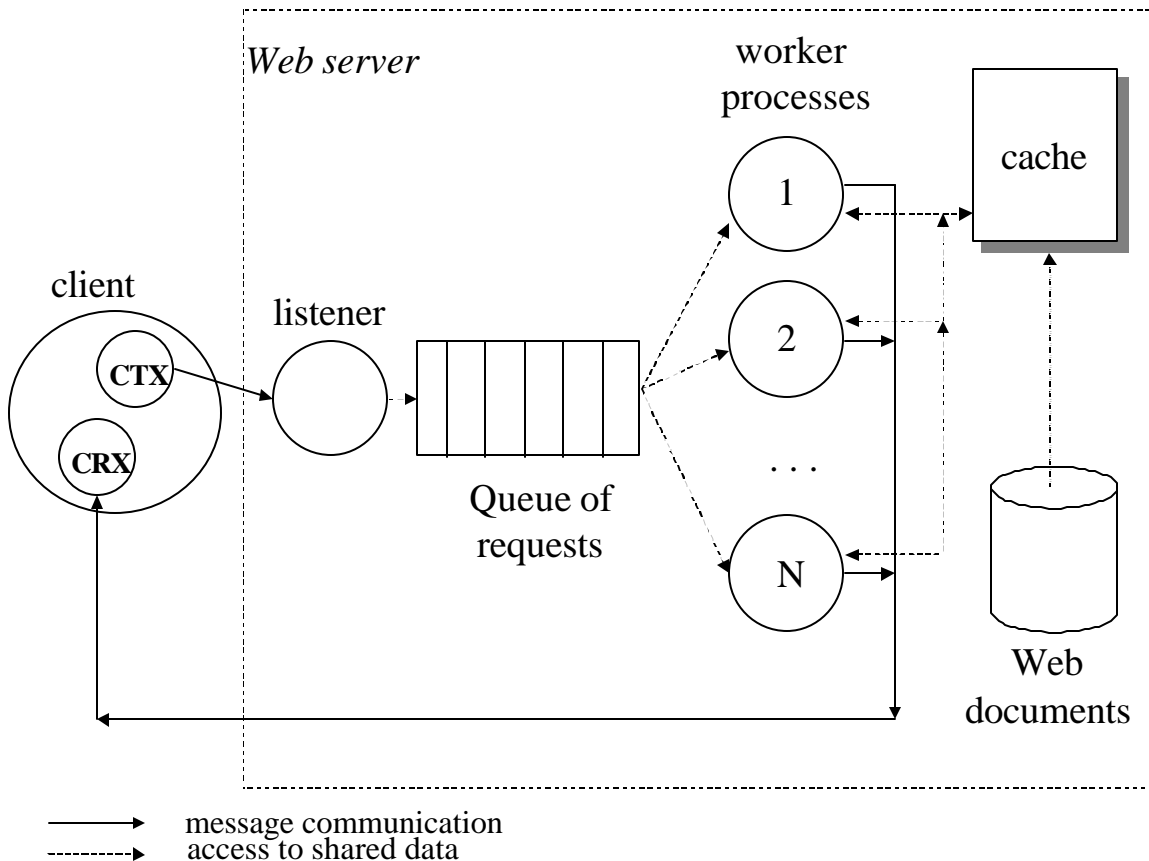


Fig 1 – Web Server Software Architecture

The client process is divided into threads of execution: CTX and CRX. These two threads are to be implemented as two distinct Unix processes. The CTX process sends requests to a listener process. The listener process adds the request to the queue of requests to be processed. When a worker process becomes idle, it takes the first request from the queue and processes the request. This involves checking if the requested file is in the cache. If it is, the worker can send a message to process CRX with the contents of the file. In case of a cache miss, the file has to be read from disk, stored in the cache, and then sent to the client. The solid arrows in Fig. 1 indicate message communication and are implemented using message passing in Unix. The dashed arrows indicate access to shared data. To solve the mutual exclusion and synchronization problem between the listener and the workers, you must use monitors. Since C does not have monitors, you have to simulate monitors with semaphores. To solve the mutual exclusion access to the cache you must use semaphores.

1. Implement procedures *up* and *down* that receive a semaphore id as a parameter and do a down and up operation, respectively, on the semaphore. These procedures encapsulate Unix's semop operation.
2. Implement the set of procedures necessary to simulate monitors with semaphores: *enter_monitor*, *leave_normal*, *leave_with_signal*, and *wait*.
3. Design and implement the monitor that solves the mutual exclusion and synchronization problem between the listener and the workers.
4. Implement the web server. You should have a parent process that does the following:
 - ❑ Creates a shared memory segment for the queue of requests and a shared memory segment for the cache. The shared memory segment for the queue of requests is shared among the listener process and the worker processes. The shared memory for the cache is shared among the worker processes.
 - ❑ Creates all needed semaphores.
 - ❑ Create two message queues: one for the communication between the client (CTX) and the listener and another for the communication between the workers and the client (CRX).
 - ❑ Creates the processes CTX, CRX, listener, and N workers.

The pseudo code for processes CTX and CRX are:

Process CTX:

```

  Read configuration file into an array;
  Loop from 1 to NumberRequests
    T1 = current-time
    Send message to Listener with request: <RequestNumber, T1, FileName>
    Sleep for a time uniformly distributed in the interval [A, B]
  EndLoop
EndProcess CTX

```

Process CRX:

```

  Loop Forever
    Blocked Receive message <RequestNumber, T1, File>
    ResponseTime = current-time - T1;
    Print log line;
    If RequestNumber = NumberRequests
      Then
        GenerateFinal Statistics and Print Results;
        Exit
      EndIf;
    EndLoop
EndProcess CRX

```

You will be supplied with the files to be served by the Web server. You should instrument your code to obtain the following measures:

- ❑ ObjectHitRatio: percent of files served from the cache.
- ❑ ByteHitRatio: percent of bytes served from the cache.
- ❑ AvgResponseTime: average time spent to serve a request. The response time is measured at the client as the time elapsed since a request is sent by the client until the reply is received. The client should number each request so that it can keep track of the response times. You should use gethrtime to obtain the high resolution time (please refer to the manual for the proper use of this function).
- ❑ RequestThroughput: average number of requests served per second.
- ❑ ByteThroughput: average number of bytes served per second.

You should experiment with two cache replacement policies:

- ❑ LRU: when the cache is full, you should remove the document with the smallest value for the LRU field.
- ❑ SmallestDocumentFirst: remove the smallest document first. Use the LRU field to break a tie.

In addition to the policies above, you should experiment with two cache admission policies:

- ❑ Unrestricted: cache any type of document that will fit in the cache.
- ❑ LimitedDocumentSize: cache only documents that do not exceed MAXSIZE.

In summary, you should report results for the following combinations: (LRU, Unrestricted), (LRU, LimitedDocumentSize), (SmallestDocumentFirst, Unrestricted), (SmallestDocumentFirst, LimitedDocumentSize).

The client should generate a log in which each line has the following format:

```
<request sequence number> <request submission time> <file requested> <response time> <First 10 bytes of the file> ... <last 10 bytes of the file>
```

Your report should have the following sections:

1. Executive summary: one page description of the purpose of the project, the approach taken to carry it out, and the results observed.
2. Table of Contents
3. Introduction: brief description of the problem and organization of the paper.
4. Software Architecture: describes the processes and data structures of the Web server.
5. Design: describes the design, in pseudo code, of your solution.
 - a. Monitor specification for the mutual exclusion and synchronization problem between the listener and workers.
 - b. Specification of the client process.
 - c. Specification of the worker process.
6. Experimental Results and Analysis: describes the experiments carried out and the results obtained. Analyze the results you obtain. You should use tables and or graphs to illustrate your points.

7. Concluding Remarks
8. Appendix A: Source Code. Please provide a very well documented code. Points will be taken for lack of or poor documentation.
9. Appendix B: Sample Output.

Your code should be available for execution in case it is required. The accuracy, as well as the presentation of the report will be considered for grading purposes. Please make sure you use a spell and a grammar checker. I will schedule a 10 minute meeting with each group in my office so that you can demonstrate your project to me.

You will be supplied with all the constants mentioned in this specification: CacheSize, N, NumberRequests, A, and B . Make sure not to hard code these values.