CSE 344 Systems Programming
Final Project Report

Mehmet Hafif
171044042

Gebze Technical University — June 28, 2020

1 Daemon Process

In order to make the process a daemon process some actions were taken.

Called fork and have the parent exit. If the daemon was started as a simple shell command, having the parent terminate makes the shell think that the command is done. The child inherits the process group ID of the parent but gets a new process ID, so we're guaranteed that the child is not a process group leader. And forked again after setting session id so that we are no more session leader and unable to open new terminals

Called setsid to create a new session. The process, becomes the leader of a new session, becomes the leader of a new process group, and is disassociated from its controlling terminal.

Unneeded file descriptors are closed.

Used a named semaphore, so that if another instance of the program tries to execute, it cant open same semaphore and exit directly.

2 Main Thread

Main thread at start makes argument checks then loads the graph to memory from its input file then closes unneeded file descriptors like stdout, then initializes the thread pool, after that it starts listening the given port, accepts incoming connection delegates the job to the thread pool and repeats this. The important thing is the job is given to the thread if there are threads in waiting condition, if all of them working, main thread blocks.

3 Thread Pool

The thread pool is created at main thread startup, at first it initializes a number of thread given as argument, every thread executes the core executer function, where most of the thread synchronisation occurs, the thread struct pointer is given as argument to this function so struct members can be accessed through that pointer, The core loop checks whether a job exist in thread structure, if there exist a job, a thread locks the job mutex takes it and executes, if not it increments the waiting thread count in the struct and waits. Threads are signaled when new job arrives. and main thread is signaled when waiting thread count is bigger than zero so that it can add more job.

An additional resizer thread is also deployed at the creation of the pool, it checks the system load and waits if it is lower than %75, and this conditon is signalled when a thread starts working and load is checked. If needed the pool size is extended an new threads are added to the pool.

When an interrupt is received threads are waited to complete then the pool is destroyed, threads are

waited by changing the run flag of the pool struct, every thread checks that value and starts execution so when it is changed, thread dont accept new job.

4 Efficiency

4.1 Cache

In order to make the program more efficient a cache structure is created and reader writer paradigm is implemented where writers have priority over reader. When a path is requested the cache is fisrt checked if path does not exist there it is calculated then added into cache, if path exist in the cache recalculation is not made.

Cache is a linked list because of ease of implementation and it is unknown that the node numbers are always start from 0 so indexing is not possible thats why sequential search is made.

5 Graph

Graph is basically an Adjacency list, Making it an array would be faster in search but since nodes are not always 0 to N we cant index them so a list structure is useful here.

6 Client

The Client simply request path between two nodes and prints the response then exits.