CS Project Report

Student1:

Name: Ng Chi To
Sid: 55742115
Email: chitong2-c@my.cityu.edu.hk

Student2:

Name: Wong Hin Leung
Sid: 56245016
Email: hlwong335-c@my.cityu.edu.hk

Student3:

Name: Hon Shing Hei
Sid: 56627774
Email: shhon-3@my.cityu.edu.hk

Problem1 design idea:

The program takes an eight-digit decimal number as an input parameter from the command line. It then creates a semaphore using sem_open() for synchronization and a shared memory segment using shmget() for inter-process communication.

After creating the semaphore and shared memory, the program forks a child process. In the child process, the multi_threads_run() function is called with the input parameter as an argument. The multi_threads_run() function converts the input parameter to an array of integers, initializes an array of eight semaphores and eight threads, and starts the threads. Each thread represents a digit in the input parameter, and each thread updates its own digit and the next digit in the array in a loop. Once all threads finish, the final array is converted back to an integer, and the saveResult() function is called to save the result to a file.

In the parent process, the global_param, local_param, and shared_param_c variables are set to the input parameter, and the parent process waits for the child process to finish using the wait() function. Once the child process has finished, the parent process deletes the shared memory segment using shmctl() and the semaphore using sem_unlink().

The multi_threads_run() function uses sem_init() to initialize an array of eight semaphores. Each semaphore is initialized to 1, which means that it is initially available for use. It then creates an array of eight threads using pthread_create(). Each thread represents a digit in the input parameter, and each thread updates its own digit and the next digit in the array in a loop. The threads use sem_wait() and sem_post() to acquire and release the semaphores, respectively, in order to ensure that only one thread can update a digit at a time.

Once all threads have finished updating the digits, the final array is converted back to an integer, and the saveResult() function is called to save the result to a file. The saveResult() function opens a file using fopen() and writes the result to the file using fprintf(). It then closes the file using fclose().

Problem2 design idea:
The design idea of the program is to count the total number of words in all text files in a given directory and its subdirectories using shared memory and semaphores for inter-process communication and synchronization between the parent and child processes.

The program first traverses the source directory and its subdirectories to find all text files and record their paths. It then calculates the number of times the child process will read data from shared memory based on the size of the text files and the size of the shared memory segment.

The program then forks a child process, and the parent process and child process run concurrently. In the parent process, data is read from each text file and written to shared memory while using semaphores to synchronize access to the shared memory between the parent and child processes. In the child process, data is read from shared memory, and the total number of words in all text files is counted while also using semaphores to synchronize access to the shared memory between the parent and child processes.

Once the child process has finished counting the number of words, it saves the result to a file, and the program cleans up by deleting the semaphores and freeing the memory allocated for the paths array.

Problem3 design idea: