CSE 344 hw2 Report

Kaan Can Bozdoğan

161044070

I used SIGUSR1 for parent to signal children processes. And SIGUSR2 for children to signal parent process.

In order to provide synchronization to the program I maintained a sig_atomic_t variable to hold the count of signals coming from children to parent. At the handler for SIGUSR1 I incremented that variable. Parent process suspend itself unless that variable has reached 8 which means all the children processes finished their work and signaled the parent about it.

At the begining of the program (before any process forked yet) I assigned the handlers for the corresponding signals so none of the processes start before their synchronization is mechanism is ready.

And then parent process forks the children and immediatelly children start processing data from file with locking the file so only one children reads and writes the file, avoiding complications. And after they calculated the data and wrote it to the file they are suspended, waiting for SIGUSR2. At the same time, at the start of the parent process, it is suspended, waiting for the SIGUSR1. When it gets the SIGUSR1 signal, it checks if the atomic variable has reached the value of 8. If not then it suspends itself again. If yes then it reads the file and calculates the avarage error for the degree 5 polynomials and prints it to the terminal.

After its job is done, the parent process sends SIGUSR2 signal to its children one by one and waits SIGCHLD signal from them. Suspended children continue to do their job as they get SIGUSR2 signals from the parent process. After they calculated and wrote the data to the file and to the terminal, they terminate, implicitly sending SIGCHLD to their parent. After parent got 8 SIGCHLD signals it knows that all of its children has terminated so it can terminate in peace too, after calculating and printing the avarage error for the degree 6 polynomials.

I used read() and write() functions for file I/O since c library functions can have undesirable results while working on such a low level method which is signals. All the processes shared a common file descriptor. Since writing to the file with write() function overrides the number of bytes it is writing, I needed to read and save the rest of the file before, write what I want to write to the file and after that, write the rest of the file I just saved (basicaly shifted the rest of the file the hard way). Because of that constant file offset change, I used lseek() a lot of times (saving the offset where I left off or calculating how many bytes remaining on the file).

Since it is not the main focus of our hw, I get the Lagrange's Interpolation function form: https://www.geeksforgeeks.org/lagranges-interpolation/

And translated a javascript code to a C code which I get from (the last answer): https://stackoverflow.com/questions/9860937/how-to-calculate-coefficients-of-polynomial-using-lagrange-interpolation