Ονοματεπώνυμο: Παντελεήμων Μαλέκας
A.M: 1115201600268

A) Mirror_client

Compilation command 1: make (with Makefile)
Compilation command 2: gcc -o mirror_client mirror_client.c List.c (without Makefile)

Run command: ./mirror_client -n id -c common -i input -m mirror -b buffer_size -l logfile

In order for the program to work the arguments must be given in that order.
Also, please type only the names of the desired folders. For example: “input1”. Don’t type “./input1”, or the path to it, since all the necessary work regarding paths is handled by the program. Also, make sure the logfiles don’t already exist. Finally IDs have to be at most 50 chars.

After getting the parameters, the program creates the paths to the desired folders and checks their validity. If the input folder doesn’t exist, the program exits. If the mirror folder does exist, the program also exits. The assert macro was used in both cases. The program also creates the logfile and writes the ID of the client in it. Then the .id file which will be placed in the common folder is created. If it exists already, assert is used again to end the program. Otherwise, we get the id of the process and we place it in the .id file. Finally, a list with the IDs is created and its own ID is placed because we’ll need it later on. Then the process of checking the common folder begins.

The general idea: Enter a While 1 loop, sleep for some time (7 sec), and after waking up, first check if new IDs arrived, then if any of the existing ones left the system.

If we find a new ID, we make sure it is not in the list. If it’s not, we add it and we create the names of the .fifo files and we begin the Inter Process Communication. Two processes are created, a writer and a reader. They code for the processes is placed in a While 1 loop. This will be explained later.

Writer Process works as follows: It sends different bytes for extra information necessary for the reader. Bytes 0,1,2,3 may be sent. Byte 0 is used to send name of the input folder. If the reader gets this, he knows he will have to store files in the mirror/id/ folder. Byte 1 is used so the reader knows it has to receive a file. Byte 2 is to tell the reader that we did all the work for the items in the input folder, so it may exit. Byte 3 is used in the recursive listdirs function to send paths from sub-directories. So, the writer process opens the input folder and does the necessary work for every item in the folder. If we stumble upon a subdir, we use the recursive listdirs function which does the same work for every folder. After doing everything in the input directory we exit the writer process.

Reader Process works as follows: First, we use the select function to check if we've had 30 seconds without input. We start a clock before the first call of select. Then we get time after calling select. We get the difference between start and end times and if it's more than 30 sec, we send the SIGUSR1 signal. Then, the reader enters a While 1 loop where it waits to read one of the 4 available bytes to proceed. Byte 0 means it will read the name of the input folder of the writer. Byte 1 means we are reading a file, according to the transfer protocol. Byte 2 means we are free to exit the While 1 loop. Byte 3 means we may read a path from a subdir.

The father process waits for the children to finish. If the reader process exited successfully, we may print a message for the successful transfer.

Logfile work: Whenever a file is sent/read we write to the logfile the number of bytes and the file's name.

Signal work: If we had more than 30 sec without input, or if any of the functions regarding the transfer failed, the child sends SIGUSR1 signal to the parent. The parent increases a global variable, trycount and kills his children. If trycount is less than 4 we enter the While 1 loop to begin the forking process anew. If it's 4 this means we tried 3 times without success, so we break from While 1 and move on.

After all that, we do the work for the clients who exited the system. We open the common folder and check if every .id file has a match with the IDs of the client's list. If one of them is not found in the list, we fork a child process that removes the local mirror dir of the client who exited. After deleting all the mirror dirs, we also remove the IDs from the list of the client.

If a client receives SIGINT or SIGQUIT signals we exit the program. Before exiting, in the handler function, we write the string "I am gone!" to the logfile so it will be used by the get_stats.sh script, we destroy the list of the client IDs, we remove the .id file from the common dir and finally we remove the mirror folder of the client.

B) create_infiles.sh

Use: ./create_infiles.sh dir_name num_of_files num_of_dirs levels

Note: Please run it again, for every input directory.

First we check the validity of the parameters. Then the input dir is created. Then the dir names are created and placed according to the levels given. Then the file names are created and they are placed with the round robin algorithm described in the presentation. Finally we go at every file and place a random string, from 1 KB to 128 KB.

C) get_stats.sh

Use: cat log1 log2 … logn | ./get_stats.sh

The script goes at every logfile, and reads every line. If it has “ID: “ we place the ID at an array. If we are on a “Bytes Received “ or “Bytes Sent “ line we add the bytes at the appropriate counters. If we’re on a “File Received “ or File Sent “ line we increment the appropriate counters. If we’re on a line with “I am gone!”, we increment the counter of the clients who exited. Then we print how many clients entered, the array of their IDs, the minimum/maximum ID, the bytes sent/received, the files sent/received and how many clients exited the system.