Report (Prog05)

2020.11.10

Baheem Ferrell

1. Dispatchers (C++)

1) Version 1.

Dispatcher version 1 expects 4 arguments in the following order.

Absolute path to the job file, absolute path to the input image directory, absolute path to the output directory and absolute path to the directory where utility executables are located.

With these 4 parameters process1() function is called to process all the lines in the given job file. (In order to handle special characters such as white spaces in the path string, double quotes are used for all of the absolute path values.)

The given job file is read line by line and the line data is split into separate arguments using strtok(). Path values are combined with the given directory path to obtain absolute paths.

Depending on the first argument of the job line, appropriate utility executable is called in a separate thread using fork() and execvp().

The main process waits every time a new job is being executed.

The corresponding result is checked for any error and it is written into a log file.

After all lines are processed, the main process exits.

2) Version 2.

Dispatcher version 2 does not expect any arguments.

All of the input data used in version 1 are passed to the dispatcher using a pipe from bash script.

The main process reads from the standard input one line at a time which contains the following data.

Absolute path to the job file, absolute path to the input image directory, absolute path to the output directory and absolute path to the directory where utility executables are located.

Due to the similarity of the arguments involved, process2() is almost similar to process1() for version 1 except for the case when end command is read.

In version 1, when end command is read the rest of the job file is ignored.

But in version 2, when end command is read, not only does the dispatcher exits but also does the script that executed the dispatcher using 2 pipes, one from the script to the dispatcher and the other from the dispatcher to the script.

3) Version 3

Version 3 differs from version 2 in the implementation of utility calls in process3() function. Instead of executing the utilities directly, it does so by sending commands through pipes to specialized dispatchers running in the

background. Only one-way data communication is used to send commands from the main dispatcher to the specialized ones. When end command is encountered, the main dispatcher sends it to all of the running dispatchers and exits only after all of them are finished. At the same time the script that executed the main dispatcher is also finished by sending End command through pipe.

2. Scripts

1) script01.sh (The builder)

This script builds ImageLibrary both in shared and static format.

It expects one argument which is the absolute path to the ImageLibrary directory.

When building dispatchers, it is assumed that the script is called from Scripts directory and the executables are all built in the same directory.

Hence, when building version 1, the script must be executed in Version 1 directory where all the dispatcher source files are located.

Similarly, when building version 2 and version 3, the script must be executed in Version 2 directory where all the dispatcher source files are located.

2) script02.sh (The watcher)

This script executes Dispatcher version 1 to process job files one at a time.

It expects 3 arguments in the following order.

Absolute path to the job file directory to watch, absolute path to the input image directory and absolute path to the output directory.

It is assumed that the executables are located in the current Script directory (where the script is called).

It continuously watches the job directory for any changes and if there is a new job file, it executes it by sending it to the dispatcher version 1.

When the dispatcher finishes processing the job file, it needs to be stopped using ctrl+c.

3) script03.sh (Piped Version)

In this version, the job files are passed to the dispatcher version 3 using pipes.

When the dispatcher encounters end command in any of the given job file, it is sent back to the script using pipes and the script exits gracefully.

Here, 2 pipes are used for input to and output from the dispatcher.

3. Miscellaneous

In this project, two assumptions are made regarding the path of the dispatcher source files and the dispatcher executables as described in 2 1) and 2 2).