Process creation and termination system calls

In the given problem we have to calculate the average of the students of two sections A and B, using three system calls fork(), waitpid(), and exit(). The fork() system call spawns a child process which processes the data of section A students and parent process processes the data of section B students and prints the average of marks of each students. The waitpid() system call keeps the parent to wait till child process finishes its work, and after finishing the work of child parent processes starts it's execution.

For the reading the data I have used the open() system to open the file in Read Only Mode and return a file descriptor then reading it through read system call, which stores the complete content of the file in a buffer string. After this processing it to store the data of the students in a structure. And finally printing the data on the terminal.

read() system call:

The "read()" system call reads data from a open file.

```
read( <file descriptor>, <buffer>, <buffer length> );
```

The "read()" function returns the number of bytes it *actually returns.* At the end of file it returns 0, or returns -1 on error

open() system call:

The open() system call open the file and returns a file descriptor.

```
open( <file path>, <mode> );
```

exit() system call:

The exit() terminates the calling process without executing the rest code which is after the exit() function. Any open file descriptors belonging to the process are closed and any children of the process are inherited by process 1, init, and the process parent is sent a SIGCHLD signal.

```
exit( <status> );
```

status − This is the status value returned to the parent process.

This functions does not return any value.

waitpid() system call:

This suspends the calling process until a child process is stopped or end. It stops the calling process until system gets status information on the child. If the system already has status information on an appropriate child when waitpid() is called, waitpid() returns immediately. waitpid() is also ended if the calling process receives a signal whose action is either to execute a signal handler or to end the process.

```
waitpid(<pid_t pid>, <int *status_ptr>, <int options>);
```

Resources:

https://www.tutorialspoint.com/c_standard_library/c_function_exit.htm

https://www.tutorialspoint.com/unix_system_calls/open.htm

https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.bpxbd00/rtwaip.htm