Report:

1. There is no memory leak. The pointer *cmd is allocated with malloc() and then later freed with free().

If a memory leak were to occur, it would be dangerous. This is because the pointer allocated in the child process is separately allocated from the parent process. After the child exits, any dynamically allocated memory that has not been freed can never be accessed again.

2. The two commands do not produce the same output. The processes are different. The first command will call the bash library located in “/bin/bash” to execute “bash” process with the arguments “ls –ll”. However, the second command will call the ls library located in “/bin/ls” to execute the “ls” process with the arguments “-ll”.

The first will execute the bash library’s version of “ls –ll” listing the directory’s files. The second command will execute listing the directory’s files along with their permissions, dates, author, and size.

3. The exec command will need a program specified as the first argument because argument ordering matters. It uses the program binary first then arguments after to determine what order processes are forked when the command is executed.

4. The program would stall and wait for input from the pipe or the program would wait for the pipe to close which would tell the program to stop listening to the pipe.

5. Yes, there is a limit. The pipe will reach the limit when the buffer has filled with data and will proceed to block the producing end until the consuming end can consume all the data. The limit is thus the amount of buffer space available for the producing end.