CS 536

Prashant Ravi — ravi18@purdue.edu

September 14, 2016

Problem 1

1 Part a

If the return value of execlp is not checked then the child process will continue in the while loop, never exiting this loop as it will wait for an input command from the terminal and then go on to call its own child. This cascading effect means every parent is waiting on its child to exit, causing much stress on the system.

2 Part b

If the parent does not wait on the child then it may cause a zombie process to linger in the system. A zombie process despite being dead will remain on the process table, waiting on it causes it to be taken off the process table and make way for creating more processes, in turn reducing stress on system.

3 Part c

Its not feasible for the server to attend to one client in a blocking waitpid call as other clients may want to gain attention at the same time. The solution to this is to delegate the actual processing of execlp in a separate child process and use a signal that lets the server know when child state has changed; SIGCHLD is this signal. On child state change, for instance, termination status, we can do a non blocking waitpid on the child with WNOHANG. It's quite possible that the SIGCHILD is raised at a time when several processes are terminating. So we could run a non blocking waitpid in a while loop to wait on several such processes that have just attained zombie status so as to clear them up from the proc table. In addition, for the timed case , we could set an a timed alarm that would alert the server that a process has spent too much time on its execlp so must be killed. This way the server is always ready to take on new clients, and by delegate the actual execution to separate child threads we provide for high availability.

4 Part d

Bug 1 : We are not doing a check on contents of the buffer. It could contain a fork bomb causing the system to crash. Also, we should allow for larger buffer size so as to fit the content of the command. Bug 2 : When we obtain the pid from the fork we are not checking if the pid is -1, which means that no child process could be created. This would in turn cause the server to go ahead and do waitpid on a -1 pid which means it would wait for any child to terminate. But its quite possible that no child was ever created. Thus, the server would wait indefinitely for some child process. We should check if k¿=0 and do a wait on that. If k == -1 then that means fork has failed and we should continue forth.