Without even running it, it looks like C code for a pseudo fork bomb. It doesn't actually call fork, but it calls argv[0]. Argv[0] is the first argument in a command, which in the case of running a program would be the program name itself. Thus it is an infinite loop which starts another instance of the program each iteration, which then itself begins spawning more instances of the program, causes memory to be filled up with an exponentially increasing number of instances of the program and CPU cycles to be taken up by the constantly running loops. This would be used by an attacker to cause system slowdown.

Having now run it in an Ubuntu VM, it is definitely a fork bomb (it's actually pretty interesting watching the RAM usage meter slowly fill up after hitting enter). However it seems that there's a limit to how many processes a user can run by default on Ubuntu, or at the very least how many it can fork. After the program filled up about 1.5-2 GB of memory the program began reporting that it could no longer fork more processes. This occured even when running the program with the 'sudo' command for root privileges. There's still a noticable slow down, but it doesn't seem like the system will crash entirely. Closing the shell instance of the user who called the fork bomb appeared to clear memory by killing all of the users active processes.

Theoretically, we could try to see what users are using the most memory and kill processes on that user. Or you could just restart the whole machine, that works too.