18.Write a C program to implement the following unix/linux command (use fork, pipe and exec system call). Your program should block the signal Ctrl-C and Ctrl-\ signal during the execution.ls –l | wc –l

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <signal.h>

int main() {

// Blocking SIGINT (Ctrl-C) and SIGQUIT (Ctrl-\) signals during execution

sigset\_t mask;

sigemptyset(&mask);

sigaddset(&mask, SIGINT);

sigaddset(&mask, SIGQUIT);

sigprocmask(SIG\_BLOCK, &mask, NULL);

// Creating a pipe

int pipefd[2];

if (pipe(pipefd) == -1) {

perror("pipe");

exit(EXIT\_FAILURE);

}

// Forking a child process

pid\_t pid = fork();

if (pid == -1) {

perror("fork");

exit(EXIT\_FAILURE);

} else if (pid == 0) { // Child process

// Closing the read end of the pipe

close(pipefd[0]);

// Redirecting stdout to the write end of the pipe

dup2(pipefd[1], STDOUT\_FILENO);

close(pipefd[1]);

// Executing the ls -l command

execlp("ls", "ls", "-l", NULL);

perror("execlp");

exit(EXIT\_FAILURE);

} else { // Parent process

// Forking another child process

pid\_t pid\_wc = fork();

if (pid\_wc == -1) {

perror("fork");

exit(EXIT\_FAILURE);

} else if (pid\_wc == 0) { // Child process

// Closing the write end of the pipe

close(pipefd[1]);

// Redirecting stdin to the read end of the pipe

dup2(pipefd[0], STDIN\_FILENO);

close(pipefd[0]);

// Executing the wc -l command

execlp("wc", "wc", "-l", NULL);

perror("execlp");

exit(EXIT\_FAILURE);

} else { // Parent process

// Closing both ends of the pipe in the parent process

close(pipefd[0]);

close(pipefd[1]);

// Waiting for the child processes to finish

waitpid(pid, NULL, 0);

waitpid(pid\_wc, NULL, 0);

}

}

// Unblocking the previously blocked signals

sigprocmask(SIG\_UNBLOCK, &mask, NULL);

return EXIT\_SUCCESS;

}