#include <stdio.h> // getline

#include <stdlib.h> // exit

#include <string.h> // strtok

#include <unistd.h> // execvp

#include <sys/wait.h> //wait

#include <fcntl.h> // fopen

#define MAX\_ARGS 256

size\_t n = 0;

char \*line = NULL;

char \*args[MAX\_ARGS];

int redirect = 0; // redirect flag

char \*filename = NULL; //redirection filename

void parse\_command\_from\_user()

{

int rc = getline(&line, &n, stdin); //The command itself

if (rc < 0) // Close the shell at end-of-input

exit(0);

int i = 0;

args[i++] = strtok(line, " \n"); //splits the command

if(args[0]!= NULL && strcmp(args[0],"exit")==0){ //Non-empty string and

//"exit" typed?

exit(0); //

}

while (i < MAX\_ARGS && args[i - 1] != NULL)

args[i++] = strtok(NULL, " \n");

int j= 0;

while(args[j] != NULL){

if (strcmp(args[j], ">") == 0) { //operator found

redirect = 1; //redirect flag is set

filename = args[j+1]; //filename user stated

args[j] = NULL; //reset flag to exit while loop

break; //done all that's needed here...

}

j++;

}

}

int write\_to\_file(char \* filename) {

int file = open(filename, O\_CREAT|O\_WRONLY, S\_IRWXU);

//int saved\_stdout = dup(STDOUT\_FILENO); //to keep original stdout

dup2(file, STDOUT\_FILENO); //duplicate descriptor and refer to stdout

fflush(stdout);

//dup2(saved\_stdout, STDOUT\_FILENO); //restore original

//close(saved\_stdout);

//printf("This will be written to the console.\n");

close(file); //close file descriptor

return 0;

}

int main() {

int start=1;

//for(;;){

while(start==1){

printf("tokar4067> ");

parse\_command\_from\_user();

int pid = fork(); // create a child process

if (pid == 0) { // child process is here

if (redirect==1) {

write\_to\_file(filename);

redirect=0;

}

execvp(args[0],args);

exit(EXIT\_SUCCESS); //exit child when finished

}

else if (pid > 0) { // parent process is here

wait(NULL); // wait for the child process to finish

}

}

return 0;

}