/\* My Shell

\* Author: Zane Durkin [<durk7832@vandals.uidaho.edu](mailto:<durk7832@vandals.uidaho.edu)>

\* CS 240

\*/

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/types.h>  
#include <unistd.h>   
  
#define MAX\_COMMAND 255

int main(){

pid\_t childpid;

char\* command[MAX\_COMMAND];

int child\_ret\_val;

while( 1 ){

printf( "?: " );

int com\_len = 0; // holds word length

int com\_arr = 0; // holds number of words

char c[MAX\_COMMAND];

while( ( c[com\_len] = getchar() ) != '\n' ) { // read until end of line

if( c[com\_len] == ' ' ){ // read until word ends

c[com\_len] = '\0'; // close each word

command[com\_arr] = malloc(com\_len); // create space for word in memory

strcpy(command[com\_arr], c); // copy string into array

com\_arr = com\_arr+1; // increment words in array

com\_len = 0; // reset character counter

}else{

com\_len++; // increment character counter

}

}

if( c[com\_len] == ' ' || c[com\_len] == '\n' )

c[com\_len] = '\0'; // be sure last argument ends with terminator

command[com\_arr] = c; // copy over last argument

command[com\_arr+1]= NULL; // be sure to set last array element to null

if( strcmp(command[0], "exit") == 0 ) // exit if requested

return 0;

if( ( childpid = fork() ) == 0 ){ // child function

if( execvp( command[0], command ) < 0 ){ // if exec failed, say why

fprintf(stderr, "Exec of %s failed\n", command[0]);

exit(1); // exit with error

}

}else if( childpid > 0){ // parent function

waitpid(childpid, &child\_ret\_val , 0); // wait for child to return

if( ( WEXITSTATUS(child\_ret\_val) ) ){ // get return value of child

fprintf( stderr, "Error Code: %d\n", WEXITSTATUS(child\_ret\_val) );

}

}else{ // if fork failed

// if childpid == -1

fprintf(stderr, "\nError in fork\n" ); // say fork failed

return 1;

}// end fork

} // end while(1)

} // end main