```
// Riley Ruckman
// TCES420, Au20
// Project 2
//
// This program implements an Command Interpreter (CI) to accept and execute user
commands.

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <string.h>
#define MAX_ARGS 200

void parsecmd(char*, char**);
int runcmd(char*);

int main(void) {

        char cmd[50] = "\0";    // String for user input

        printf("OS420> "); /* Prompts OS420> */
        fgets(cmd,50,stdin); /* Reads user input */
        cmd[strlen(cmd)-1] = '\0'; /* Removes '\n' character read from fgets() */

        // Continues to ask for user input until user inputs "exit"
        while (strcmp(cmd,"exit")){

                runcmd(cmd); /* Executes command */

                printf("OS420> "); /* Prompts OS420> */
                fgets(cmd,50,stdin); /* Reads user input */
                cmd[strlen(cmd)-1] = '\0'; /* Removes '\n' character read from
fgets() */

        }
        return 0;
}

// Parses user input from the command line and executes the specified program with
the given arguments.
// Returns the status of the child process
int runcmd(char *cmd) {

        char* argv[MAX_ARGS];   // String array pointer for program and arguments
        pid_t child_pid;        // PID of child process
        int child_status;
        parsecmd(cmd,argv); /* Parses user input into program and arguments */

        // Creates child process
```

```
        child_pid = fork();
        if(child_pid == 0) {
                /* This is done by the child process. */
                execvp(argv[0], argv);
                /* If execvp returns, it must have failed. */
                printf("Unknown command\n");
                exit(0);
        }
        else {
                /* This is run by the parent. Wait for the child to terminate. */
                pid_t tpid = wait(&child_status); /* wait on state change*/
                return child_status;
        }

}
// Takes a string pointer and parses it based off the characters ' ', '\t', and
'\n'.
// Tokens are stored in sequential order using a string array pointer
// Given in CreateProcess.pptx
void parsecmd(char *line, char **argv) {

        while (*line != '\0') {     /* if not the end of line ..... */
                while (*line == ' ' || *line == '\t' || *line == '\n')
                        *line++ = '\0';     /* replace white spaces with \0 */
                *argv++ = line;         /* save the argument position */
                while (*line != '\0' && *line != ' ' && *line != '\t' && *line !=
'\n')
                        line++;             /* skip the argument until ... */
        }
        *argv = '\0';                   /* mark the end of argument list */
}
```