Tuan Nguyen

Simple shell

-provided user with a list of build in internal command

ex:

- cd: change directory

-clr : clear the screen

-dir<directory>: list all contents under <directory>

-environ : list all environment variables

-echo<comment> : display <comment> in the screen

-help : display the user manual of use of program

-pause : pause the shell until user hit Enter to continues

-quit : quit the shell

Main.c

- how to run the shell, create processes, I/O redirection and piping

**int main(){**

**Boolean true; // set Boolean to true mean that run the command until user hit exits.**

**char \*input; // hold command input**

**//the shell runs in a loop**

**while(true){**

**printf(“Enter command to run: );**

**char \*input = read\_input() // read user input**

**char args = parse\_arg(input) // parse to get the command**

**run\_command(args) //execute and check state of the shell**

**}**

**#define BUFFER\_SIZE 1024 // here I defined buffer size, an abbreviation for the token 1024, it is the maximum input line length**

**//read user input and store in a buffer, the buffer will be double if user input exceed the size of the buffer**

**char \*read\_input()**{

int ch;

int buff\_size = BUFFER\_SIZE;

int pos = 0;

char \*buff = malloc(sizeof(char)\*buff\_size); // set the size of input to read , dynamic allocation the buffer

if(buff == NULL){ // if the buff is empty print cannot allocate

printf("Buffer Cannot allocate");

}

while(1){ // read every character from input

ch = getchar(); // get the character

if(ch == EOF || ch == '\n'){ // if it is not the end of the file or end of the line

buff[pos] = '\0';

return buff;

}

else{

buff[pos] = ch;

}

pos++;

if(pos > buff\_size){ //reallocate if size of buffer is exceed

buff\_size += BUFFER\_SIZE;

buff = realloc(buff , sizeof(char)\*buff\_size);

if(buff == NULL){ // buffer allocation failed

printf("Buffer Cannot reallocate");

}

}

}

free(buff);

}

**//parse user input into args which separated by t\r\n\a**

**char \*\*parse\_input(char \*input){**

char \*\*tokens;

char \*token;

int pos = 0;

if ((tokens = malloc(sizeof(char\*) \* TOKEN\_SIZE)) == NULL) // token is empty, print cannot allocate

printf(“Buffer cannot allocation”);

exit(EXIT\_FAILURE);

token = strtok(input, SYMBOLS); // split input string into token characters, symbols such as t\r\n\a

while (token != NULL) { // token is not empty

tokens[pos++] = token; // kepp reading the string

token = strtok(NULL, SYMBOLS);

}

tokens[pos] = NULL;

return tokens; // return the string of input

}

**// defined the build in command such as cd , dir, clr .. etc**

**char \*command[] = {**

“cd”, // change directory

“clr” // clear the screen

“dir” // list of content in directory

“environ” // list of enviroment string

“echo” //display comment on display

“help” // help manual to use the shell

“pause” //pause the input until user hit continues

“quit” // exits the shell

}

**//function to run internal command from terminal take an argument as input**

**int run\_command( char \*\* input){**

if (“input[0] == NULL”) //

return 1; // empty input

}