Yeditepe University
Department of Computer Engineering

CSE 232
Systems Programming
Fall 2018

Term Project

Due to: 6th January 2019

In this project you will implement an **interpreter** (named MyShell) for a subset of **Linux shell** commands.

You will use **C language** for programming. Use **gcc** compiler in Linux environment. Your implementation should be compatible with the setup in the laboratory. Prior to project submission, you must make sure that your project executes correctly with the specified setting. Projects producing errors during the demonstrations due to the differences in the platform will not be accepted.

Your interpreter should read a command from the keyboard, parse the command into its tokens and display the tokens, then process the command, and wait for a new command. These steps should be repeated for all lines until the user enters -1.

Following commands must be implemented.

Simple commands

echo
displays a line of text or the value of a variable
ex:
```
echo "this is a text"
echo $a
```

cp
copies the source file to the destination file
ex:
```
cp srcfile destfile
```

wc
displays line (-l), word (-w) and byte (-c) counts for a file
ex:
```
wc -l myfile
```

cat
displays the contents of the file
ex:
```
cat myfile
```

expr
evaluates the expression and displays the result. Operators: +, -, *, /, %
ex:
```
expr $a + 2
```

Compound commands

| **pipe**

Writes the output of the first command to a temporary file named temp, and the second command takes its input from temp
Ex:
```
cat myfile | wc -l
```

Special characters

=	assignment
` `	result of a command
" "	string
$	value of a variable
-	option of a command

Ex:
```
c=5
c=`expr $a + 2`
```

The project consists of two parts. In Part 1, you will implement a parser. In Part 2, you will implement the interpreter.

Part 1

Your parser should read the command given from the keyboard, parse it into its tokens and place the tokens in the following parse table. Tokens are separated by a blank or one of the above special characters.

Parse Table

	Exists	Command	Parameter1	Parameter2	Parameter3
Command 1					
Command 2					
Assignment					

Use the following data structure:

```
struct ParseTable {
      int exists;       // 1-exists, 0- does not exist
      char cmd[10];     // command
      char par1[10];    // parameter 1
      char par2[10];    // parameter 2
      char par3[10];    // parameter 3
};
struct ParseTable PT[3];
```

Print the parse table for each command, separated by ";"

Ex: for `c=`expr $a + 2``

print:

```
1; expr; a; +; 2
0; ; ; ;
1; ; c; ;
```

Ex: for `cat f1 | wc -l`

print:

```
1; cat; f1; ;
1; wc; l; ;
0; ; ; ;
```

While implementing this part, keep in mind that you will need the parser in Part 2.

Part 2

In Part 2 you will implement the interpreter. Write the following functions:

```
void echo(char *str);      // if parameter is a string, display it, if it is a variable name search ST and display its value
void cp(char *fs, char *fd);    // read the contents of fs and write it to fd
int wc(char opt, char *fname); //depending on the option opt, line, word or character count of fname is returned
void cat(char *fname);          // read file and display
int expr(char *opnd1, char opr, char *opnd2);     // search the operands opnd1 and opnd2 in ST,
                                                  //evaluate the expression and return the result
void assign(char *lhs, int value);    // search for lhs in ST and change its value
```

Use the following data structure for symbol table:

```
struct SymbolTable {
      char symbol[10];
      int value;
};
struct SymbolTable ST[20];
```

Parse table and Symbol table will be global.

In the main loop, read a command, parse the command and create the parse table and print it, check the command type and call the related function with appropriate parameters.