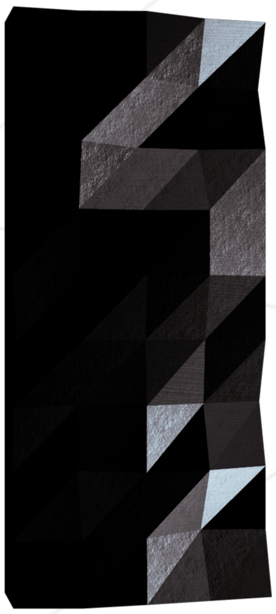

Microshell Piscine

microshell-02

Summary: This document is the subject for the microshell-02 of the Microshell Piscine @ 42 Tokyo.

Contents

Chapter I

Instructions

- Your project must be written in C.
- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / man / the Internet / ....`
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- Instruction which are not written or not shown on the example are considered undefined. you should define those undefined behavior reasonably.
- Segmentation Fault or other unexpected termination of a program(double free, infinite loop) should not happen. If it occurs, your grade will be `0` during evaluation.
- No memory leak are allowed. If it occurs, your grade will be `0` during evaluation.
- If the subject requires it, you must submit a Makefile which will compile your source files to the required output with the flags -Wall, -Wextra and -Werror, use gcc.
- Your Makefile must at least contain the rules $(NAME), all, clean, fclean and re. If it doesn't compile with these flags, your grade will be `0` during evaluation.

- Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a 0 if there is a norm error inside.
- Your project must compile and executed on guacamole.42tokyo.jp. If It doesn't compile or execute on guacamole.42tokyo.jp, your grade will be `0` during evaluation.

Chapter II

Foreword

`Abstract Syntax Tree...?`

Chapter III

Exercise 00 : Integer and Boolean

	Exercise 00
Integer and Boolean	
Turn-in directory : *ex*00/	
Files to turn in : `*`	
Allowed functions : `read, malloc, free, write`	

Create a program which meets the following requirements.

- When the program is launched, display a prompt(For example "$> ").
- Implement necessary functionality so that the program behave as shown in the example below and follow the grammar.

  Example)

```
?> ./microshell-02
$> 123
$> true
$> false
$> qqqq
Syntax Error
$>
$>
```

```
/* -----------------------------------------------------------
 The grammar symbols
 ----------------------------------------------------------- */

%token BOOL // a bool consisting of reversed keyword: TRUE, FALSE.
%token NUM /* a number consisting solely of digits.
            number's value is between -32767 and 32767. */
%token NEWLINE // '\n'

/* The following are the reserved words. */

%token  TRUE     FALSE
/*      'true'   'false' */

/* -----------------------------------------------------------
 The Grammar
 ----------------------------------------------------------- */
%start program
%%

program          : command NEWLINE
                 | NEWLINE

command          : value

value            : BOOL
                 | NUM
```

Chapter IV

Exercise 01 : write and exit

	Exercise 01
write and exit	
Turn-in directory : *ex*01/	
Files to turn in : `*`	
Allowed functions : `read, malloc, free, exit, write`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement necessary functionality so that the program behave as shown in the example below and follow the grammar.

  Example)

```
?> ./microshell-02
$> 123
$> write 123
123
$> true
$> write true
true
$> write false
false
$> write qqqq
Syntax Error
$> exit
?>
```

```
/* ---------------------------------------------------------
 The grammar symbols
 --------------------------------------------------------- */

%token BOOL // a bool consisting of reversed keyword: TRUE, FALSE.
%token NUM /* a number consisting solely of digits.
            number's value is between -32767 and 32767. */
%token NEWLINE // '\n'

/* The following are the reserved words. */

%token  TRUE     FALSE
/*      'true'   'false' */

%token  WRITE     EXIT
/*      'write'   'exit' */

/* ---------------------------------------------------------
 The Grammar
 --------------------------------------------------------- */
%start program
%%

program          : command NEWLINE
                 | NEWLINE

command          : write_stmnt
                 | exit_stmnt
                 | value

write_stmnt      : WRITE value

exit_stmnt       : EXIT

value            : BOOL
                 | NUM
```

Chapter V

Exercise 02 : Basic arithmetic

	Exercise 02
Basic arithmetic	
Turn-in directory : *ex*02/	
Files to turn in : `*`	
Allowed functions : `read, malloc, free, exit, write`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement necessary functionality so that the program behave as shown in the example below and follow the grammar.

  Example)

  ```
  ?> ./microshell-02
  $> 123
  $> write 123+123
  246
  $> write 3*4/2
  6
  $> write 1+3*4/2
  7
  $> write false+1
  Execution Error
  ```

```
/* ----------------------------------------------------------
 The grammar symbols
 ---------------------------------------------------------- */

%token BOOL // a bool consisting of reversed keyword: TRUE, FALSE.
%token NUM /* a number consisting solely of digits.
            number’s value is between -32767 and 32767. */
%token NEWLINE // ’\n’

/* The following are the reserved words. */

%token  TRUE     FALSE
/*      ’true’   ’false’ */

%token  WRITE     EXIT
/*      ’write’   ’exit’ */

/* The following are the reserved characters. */

%token  PLUS    MINUS    MUL    DIV
/*      ’+’     ’-’      ’*’    ’/’ */

/* ----------------------------------------------------------
 The Grammar
 ---------------------------------------------------------- */
%start program
%%

program           : command NEWLINE
                  | NEWLINE

command           : write_stmnt
                  | exit_stmnt
                  | expr

write_stmnt       : WRITE expr

exit_stmnt        : EXIT

expr              : term sub_expr
                  | term

sub_expr          : PLUS term
                  | MINUS term
                  | PLUS term sub_expr
                  | MINUS term sub_expr
```

```
term                : factor sub_term
                    | factor

sub_term            : DIV factor
                    | MUL factor
                    | DIV factor sub_term
                    | MUL factor sub_term

factor              : value

value               : BOOL
                    | NUM
```

Chapter VI

Exercise 03 : Variable and parenthesis

	Exercise 03
Variable and parenthesis	
Turn-in directory : *ex*03/	
Files to turn in : `*`	
Allowed functions : `read, malloc, free, exit, write`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement necessary functionality so that the program behave as shown in the example below and follow the grammar.

  Example)

```
?> ./microshell-02
$> a=1
$> write $a
1
$> b=(1+1)*2
$> write $b
4
$> write $a + $b
5
$> write $$
Syntax Error
$> write $c
Undefined Variable: c
```

```
/* ----------------------------------------------------------
 The grammar symbols
 ---------------------------------------------------------- */

%token BOOL // a bool consisting of reversed keyword: TRUE, FALSE.
%token NUM /* a number consisting solely of digits.
            number's value is between -32767 and 32767. */
%token NAME /* a word consisting solely of underscores, digits,
             and alphabetics from the portable character set.
             The first character of a name is not a digit.*/
%token NEWLINE // '\n'

/* The following are the reserved words. */

%token  TRUE     FALSE
/*      'true'   'false' */

%token  WRITE     EXIT
/*      'write'   'exit' */

/* The following are the reserved characters. */

%token  PLUS    MINUS    MUL    DIV
/*      '+'     '-'      '*'    '/' */

%token  LP    RP
/*      '('   ')' */

%token  ASN
/*      '='  */

%token  EXPAND
/*      '$'  */

/* ----------------------------------------------------------
 The Grammar
 ---------------------------------------------------------- */
%start program
%%

program          : command NEWLINE
                 | NEWLINE

command          : write_stmnt
                 | exit_stmnt
                 | assignment_stmnt
                 | expr
```

```
write_stmnt      : WRITE expr

exit_stmnt       : EXIT

assignment_stmnt : variable ASN expr

expr             : term sub_expr
                 | term

sub_expr         : PLUS term
                 | MINUS term
                 | PLUS term sub_expr
                 | MINUS term sub_expr

term             : factor sub_term
                 | factor

sub_term         : DIV factor
                 | MUL factor
                 | DIV factor sub_term
                 | MUL factor sub_term


factor           : value
                 | LP expr RP
                 | expand_variable

expand_variable  : EXPANDvariable

value            : BOOL
                 | NUM

variable         : NAME
```

Chapter VII

Bonus

	Exercise 04
more functionality	
Turn-in directory : *ex*04/	
Files to turn in : *	
Allowed functions : *	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement other `features` which improve users experience. (For example: loop, functions, etc...)
- For each `features` which improve user's experience, it will be graded 1point.(MAX 5points)

  Example)

```
?> ./minishell-02
$> i=0
while [i < 10]
do
write $i
i = $i + 1
done
0
1
2
3
4
5
6
7
8
9
$>
```