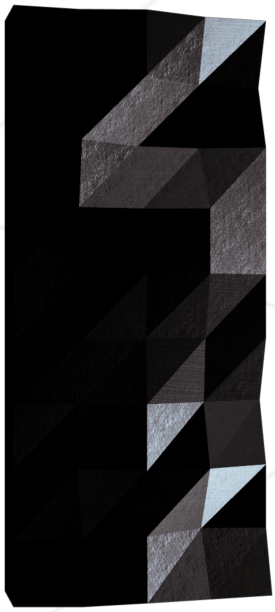

Microshell Piscine

microshell-00

Summary: This document is the subject for the microshell-00 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

builtin...?

Chapter III

Exercise 00 : exit

	Exercise 00
exit	
Turn-in directory : *ex*00/	
Files to turn in : `*.c, *.h, Makefile`	
Allowed functions : `write, read, malloc, free, exit`	

Create a program which meets the following requirements.

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

  Example)

```
?> ./microshell-00
$> aaa
builtin not found: aaa
$> wwwwwwwwwwww
builtin not found: wwwwwwwwwwwww
$>
$>
$> exit
?>
```

`shell builtin`

Chapter IV

Exercise 01 : echo

	Exercise 01
echo	
Turn-in directory : $ex01/$	
Files to turn in : `*.c, *.h, Makefile`	
Allowed functions : `write, read, malloc, free, exit`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement `echo` builtin so that the program behave as shown in the example below.

  Example)

```
?> ./microshell-00
$> aaa
builtin not found: aaa
$> wwwwwwwwwwww
builtin not found: aaa
$> echo

$> echo a
a
$> echo ''
''
$> echo '"` a
'"` a
```

Chapter V

Exercise 02 : cd && pwd

	Exercise 02
cd && pwd	
Turn-in directory : *ex*02/	
Files to turn in : `*.c, *.h, Makefile`	
Allowed functions : `write, read, malloc, free, exit, getcwd, chdir, access`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement `cd` and `pwd` builtin so that the program behave as shown in the example below.

  Example)

```
?> ./microshell-00
$> pwd
/tmp
$> pwd aaa
pwd: too many arguments
$> cd /private/tmp
$> pwd
/private/tmp
$> cd ..
$> pwd
/private
$> cd ./tmp/../tmp
/private/tmp
$> cd
cd: not enough arguments
$> cd aaa aaaa
cd: too many arguments
$> cd /testtesttest
cd: no such file or directory: /testtesttest
$> cd /bin/ls
cd: not a directory: /bin/ls
$> cd noright
cd: permission denied: noright
```

No need to handle white space characters. man isspace

Chapter VI

Exercise 03 : env

	Exercise 03
env	
Turn-in directory : $ex03/$	
Files to turn in : `*.c, *.h, Makefile`	
Allowed functions : `write, read, malloc, free, exit, getcwd, chdir, access`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement `env` builtin so that the program behave as shown in the example below.

  Example)

```
?> env -i micro=shell ./microshell-00
$> env
micro=shell
$> env aaa
env: too many arguments
```

Chapter VII

Exercise 04 : export

	Exercise 04
export	
Turn-in directory : $ex04/$	
Files to turn in : `*.c, *.h, Makefile`	
Allowed functions : `write, read, malloc, free, exit, getcwd, chdir, access`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement `export` builtin so that the program behave as shown in the example below.

  Example)

```
?> env -i ./microshell-00
$> export
export: not enough arguments
$> export toto=tata
$> env
toto=tata
$> export toto=
$> env
toto=
$> export =tata
export: error
$> export toto=tata titi=tutu
$> env
toto=tata
titi=tutu
```

Chapter VIII

Exercise 05 : unset

	Exercise 05
unset	
Turn-in directory : $ex05/$	
Files to turn in : `*.c, *.h, Makefile`	
Allowed functions : `write, read, malloc, free, exit, getcwd, chdir, access`	

Create a program which meets the following requirements.

- Implement previously required features.
- Implement `unset` builtin so that the program behave as shown in the example below.

  Example)

```
?> env -i ./microshell-00
$> unset
unset: not enough arguments
$> export toto=tata
$> env
toto=tata
$> unset toto
$> env
$> export toto=tata titi=tutu
$> env
toto=tata
titi=tutu
$> unset toto titi
$> env
$>
```

Chapter IX

Bonus

	Exercise 06
more builtin	
Turn-in directory : $ex06/$	
Files to turn in : `*.c, *.h, Makefile`	
Allowed functions : `*`	

Create a program which meets the following requirements.

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