Network Programming Project 1 - NPShell

NP TA

Deadline: Monday, 2018/10/22 23:59

1 Introduction

In this project, you are asked to design a shell with special piping mechanisms.

2 Scenario of using npshell

2.1 Some important settings

- This working directory includes the following:
  - bin/
  - test.html (test file)
- The structure of your working directory:

```
your_working_dir
       |-----bin              # The directory contains  executables.
       |      |---cat
       |      |---ls
       |      |---noop        # A program that does nothing.
       |      |---number      # Add a number to each line of input
       |      |---removetag   # Remove HTML tags and output to stdout,
       |      |                 without altering the input file.
       |      |---removetag0  # Same as removetag,
       |                        but outputs error messages to stderr.
       |
       |-----test.html
```

- In addition to the above executables, the following are built-in commands supported by your npshell.
  - setenv
  - printenv
  - exit

2.2 Scenario

```
bash$ ./npshell     # Execute your npshell
% printenv PATH     # Initial PATH is bin/ and ./
bin:.
```

```
% setenv PATH bin   # Set environment variable PATH to bin/ only.
% printenv PATH
bin
% ls
bin/ test.html
% ls bin
cat  ls  noop  number  removetag  removetag0
% cat test.html > test1.txt
% cat test1.txt
<!test.html>
<TITLE>Test</TITLE>
<BODY>This is a <b>test</b> program
for ras.
</BODY>
% removetag test.html

Test
This is a test program
for ras.

% removetag test.html > test2.txt
% cat test2.txt

Test
This is a test program
for ras.

% removetag0 test.html
Error: illegal tag "!test.html"

Test
This is a test program
for ras.

% removetag0 test.html > test2.txt
Error: illegal tag "!test.html"
% cat test2.txt

Test
This is a test program
for ras.

% removetag test.html | number
  1
  2 Test
  3 This is a test program
  4 for ras.
  5
```

```
% removetag test.html |1    # This pipe will pipe stdout to next command.
% number                    # The command's stdin is from the previous pipe.
  1
  2 Test
  3 This is a test program
  4 for ras.
  5
% removetag test.html |2 # |2 will skip 1 line and then pipe stdout to the next line.
% ls
bin/ test.html   test1.txt   test2.txt
% number                 # The command's stdin is from the previous pipe.
  1
  2 Test
  3 This is a test program
  4 for ras.
  5
% removetag test.html |2 # This pipe will pipe stdout to next next line.
% removetag test.html |1 # This pipe will pipe stdout to next line.
                          (Note: merge with the previous one)
% number                 # The command's stdin is from the previous pipe.
  1
  2 Test
  3 This is a test program
  4 for ras.
  5
  6
  7 Test
  8 This is a test program
  9 for ras.
 10
% removetag test.html |2    # This pipe will pipe stdout to next next line.
% removetag test.html |1    # This pipe will pipe stdout to next line.
                             (Note: merge with the previous one)
% number |1     # The command's stdin is from the previous pipe, but piped to next line.
% number        # The command's stdin is from the previous pipe.
  1   1
  2   2 Test
  3   3 This is a test program
  4   4 for ras.
  5   5
  6   6
  7   7 Test
  8   8 This is a test program
  9   9 for ras.
 10  10
% removetag test.html | number |1
% number
  1   1
```

```
  2   2 Test
  3   3 This is a test program
  4   4 for ras.
  5   5
% ls |2
% ls
bin/ test.html   test1.txt   test2.txt
% number > test3.txt
% cat test3.txt
  1 bin/
  2 test.html
  3 test1.txt
  4 test2.txt
% removetag0 test.html |1
Error: illegal tag "!test.html" # Error message write to strerr
% number
  1
  2 Test
  3 This is a test program
  4 for ras.
  5
% removetag0 test.html !1 # This pipe will pipe both stdout and stderr to next line.
                           # !n is the same as |n, except pipe both stdout and stderr.
% number    # The command's stdin is from the previous pipe.
  1 Error: illegal tag "!test.html"
  2
  3 Test
  4 This is a test program
  5 for ras.
  6
% date
Unknown command: [date].
# Let TA copy the command /bin/date into bin/ under your working directory
% date
Fri Sep 28 00:49:39 CST 2018
% exit
bash$
```

3 Requirements and Hints

1. The programs **removetag**, **removetag0**, **number**, **noop** are offered by TA in this project. TAs will upload them onto e3. Please compile them by yourself and put the executable file into folder **${working_dir}/bin/**

   ```
   Compile example: g++  noop.cpp  -o  my_working_dir/bin/noop
   ```

2. Two of the commands (**ls** and **cat**) are usually placed in the folder **/bin** in UNIX-like systems. Please copy them into the folder **${working_dir}/bin/**

```
e.g. cp  /bin/ls  /bin/cat  my_working_dir/bin
```

3. During demo, other commands will be copied to bin/ under your working directory by TAs. Your npshell program should be able to execute them.
4. You **must** use **exec**-based functions to run commands, **except** for built-in functions: **setenv**, **printenv** and **exit**.
   You **must not use** functions like **system()** or some other functions (in lib) to do the job.
5. When you implement **output** redirection to a file: >, if the file already exists, the file should be be overwritten.
6. You don't have to worry about outputting to both file and pipe for the same command.

```
% ls > test.txt | cat   # This kind of situations will not appear in any test cases.
```

7. You **do not** need to implement **input** redirection from a file: <.
8. You can only implement the npshell with **C** and **C++**.

4 Specification

4.1 About Input

1. The length of a single-line input will not exceed 15000 characters.
2. Each command will not exceed 256 characters.
3. There must be one or more spaces between commands and symbols (or arguments.), but no spaces between pipe and numbers.

```
% cat hello.txt | number
% cat hello.txt |4
% cat hello.txt !4
```

4. There will not be any '/' character in test cases.

4.2 About NPShell

1. Use '% ' as the command line prompt. Notice that there is one space after %.
2. The npshell terminates after receiving the **exit** command or **EOF**.
3. Note that you have to handle the forked processes properly, or there might be zombie processes.

4.3 About setenv and printenv

1. The initial environment variable **PATH** should be set to **bin/** and **./** by default.

```
% printenv PATH
bin:.
```

2. setenv usage: **setenv** [**environment variable**] [**value to assign**]

```
% setenv LANG en_US.UTF-8
```

3. printenv usage: **printenv** [**environment variable**]

```
% printenv NOTHING  # Print an environment variable that doesn't exist. Show nothing.
% printenv LANG
en_US.UTF-8
```

4. The number of arguments for **setenv** and **printenv** in all test cases will be correct.
5. **setenv** and **printenv** will appear solely in a line. No other cammands will be in the same line.
6. **setenv** and **printenv** will not be piped.

```
% printenv PATH | cat # There will be no such kind of test case.
```

4.4 About Numbered-Pipes and Ordinary Pipe

1. |**N** means the **stdout** of the last command should be piped to the first command of next Nth line, where $1 \leq N \leq 1000$.
2. **!N** means both **stdout and stderr** of last command should be piped to the first command of next Nth line, where $1 \leq N \leq 1000$.
3. |**N** and **!N** will only appear at the end of the line.
4. | is an ordinary pipe, it means the stdout of the last command will be piped to the next command within the same line. It will **only** appear between two commands.
   It will **not** appear at the beginning or at the end of the line.

```
% ls | number
   1 bin
   2 test.html
% ls |      # This kind of situation will not apear in any test cases.
% | ls      # This kind of situation will not apear in any test cases.
% ls |    | ls      # This kind of situation will not apear in any test cases.
```

5. If there is any error in a input line, the line number still counts.

```
% ls |2
% ctt       # Unknown command, line number is counted.
Unknown command: [ctt].
% number
1 bin/
2 test.html
```

6. **setenv** and **printenv** count as one line.

```
e.g.
% ls |2
% printenv PATH
bin:.
% cat
bin
test.html
```

7. New line **does not** count as one line.

```
e.g.
% ls |1
%           # Press Enter on your keyboard.
% number
   1 bin
   2 test.html
```

4.5 About Unknown Command

1. If there is an unknown command, print as the following:

```
Unknown command: [command].
e.g.
% ctt
Unknown command: [ctt].
```

2. You don't have to print out the arguments.

```
% ctt -n
Unknown command: [ctt].
```

3. The commands after unknown commands will still be executed.

```
% ctt | ls
Unknown command: [ctt].
bin/   test.html
```

4. Messages piped to unknown commands will disappear.

```
% ls | ctt
Unknown command: [ctt].
```

4.6 About Submission

1. E3
   (a) Create a directory named your student ID, put your files in the **same directory layer**.
   (b) You must provide a **Makefile**, which compiles your source code into one **executable** named **npshell**. The executable should be under the same directory as the source codes. We will use this executable for demo.
   (c) Upload **only** your code and Makefile.
   Do not upload anything else (e.g. **removetag**, **noop**, **test.html**, **ls**...)
   (d) **zip** the directory and upload the .zip file to the E3 platform
   Attention!! we only accept .zip format

```
e.g.
Create a directory 0756000, the directory structure may be:
0756000
    |-- Makefile
    |-- npshell.cpp
    |-- npshell.h
    |...
    |...
zip the folder 0756000 into 0756000.zip, and upload 0756000.zip onto E3
```

2. Bitbucket:
   (a) Create a **private** repository: ${your_student_ID}_np_project1 inside the **nctu_np_2018** team, under **np_project1**.
   Set the ownership to **nctu_np_2018**

```
e.g. 0756000_np_project1
```

   (b) For each project, you need to commit on bitbucket for **at least 5 times**.
   (c) You can push anything you need onto bitbucket (including **removetag**, **noop**, **test.html**...), as long as the size of the file is reasonable.

3. **We take plagiarism seriously.**

```
All projects will be checked by a cutting-edge plagiarism detector.
You will get zero points on this project for plagiarism.
Please don't copy-paste any code from the internet, this may be
considered plagiarism as well.
Protect your code from being stolen.
```

4.7 Notes

1. NP project should be run on NP servers (to be announced), otherwise, your account may be locked.
2. Any abuse of NP server will be recorded.
3. Don't leave any zombie processes in the system.
4. You will lose points for violating any of the rules mentioned in this spec.
5. Enjoy the project!