3.11 Programming Project: sh Simulator

The programming project is to write a C program which simulates the Linux sh for commands executions. The object is for the reader to understand how Linux sh works. The tools used are fork (), exec(), close(), exit(), pipe() system calls and string operations. It is suggested that the reader should try to implement the project in successive steps.

3.11.1 Single Command with I/O Redirection

1. Prompt the user for an input line, which is of the form

   ```
   cmd arg1 arg2 arg3 .... argn
   ```

   where cmd is a command, and arg1 to argn are command line parameters to the cmd program. Valid commands include "cd", "exit" and ANY Linux executable files, e.g. echo, ls, date, pwd, cat, cp, mv, cc, emacs, etc. In short, the sh simulator can run the same set of commands as Linux sh.
2. Handle simple commands:

   ```
   cmd = "cd"   :  chdir(arg1) OR chdir(HOME) if no arg1;
   cmd = "exit" :  exit(0) to terminate;
   ```

3. For all other commands:

   ```
   fork a child process;
   wait for the child to terminate;
   print child's exit status code
   continue step 1;
   ```

4. **Child process:** First, assume NO pipe in command line:

4-1. Handle I/O redirection:

```
cmd  arg1 arg2 ...  <  infile    // take inputs from infile
cmd  arg1 arg2 ...  >  outfile   // send outputs to outfile
cmd  arg1 arg2 ...  >> outfile   // APPEND outputs to outfile
```

4-2. Execute cmd by execve(), passing parameters

```
char *myargv[ ], char *env[ ]
```

to the cmd file, where myargv[ ] is an array of char * with

```
myargv[0]->cmd,
myargv[1]->arg1, .....,
End with a NULL pointer
```

Sh searches for executable commands in the directories of the PATH environment variable. The sh simulator must do the same. Thus, the simulator program must tokenize the PATH variable into individual directories for locating the command files.

3.11.2 Commands with Pipes

5. After verifying mysh works for simple commands, extend it to handle pipes. If the command line has a | symbol, divide it into head and tail: e.g.

```
        cmd1 < infile | cmd 2 > outfile
head = "cmd < infile";   tail = "cmd 2 > outfile"
```

   Then implement the pipe by the following steps.

   create a pipe
   fork a child process to share the pipe
   arrange one process as the pipe writer, and the other one as the pipe reader.
   Then let each process execve() its command (possibly with I/O redirection).

6. **Multiple pipes:** If the command line contains multiple pipe symbols, implement the pipes by recursion. The reader may consult Chap. 13 of (Wang 2015) for the recursive algorithm.

3.11.3 ELF executable vs. sh script files

In Linux, binary executables are in the ELF file format. The first 4 bytes of an ELF file are the ELF identifier '0x7F'ELF. Sh scripts are text files. The first line of most sh script files contains

```
#!  /bin/sh
```

With these hints, the reader should be able to devise ways to test whether a command is an ELF executable or a sh script. For sh script files, run the Linux command /bin/bash on the files.

3.11.4 Sample Solution

Sample solution of the programming project is available online for download. The reader may download the binary executable file kcsh.bin and run it under Linux. Source code of the project for instructors is also available on request. Figure 3.9 shows the sample outputs of running the sh simulator program.