Junxiang Wen

Edit by 9/25/2019

3207

Project 2A: Pseudocode for Implementing a Linux Shell

Program Myshell

while true do

line <- read line from stdin

infile <- get arg from args after '<'

redirect stdin to infile

outfile <- get arg from args after '>'

redirect stdout to outfile

pipe <- create pipe

commands <- split line by '|'

for command in commands

do

runOneCommand(command, pipe)

end

runOneCommand(args, pipe)//type is array

//run the specific function for the command you input

//use switch-case flow control to finish job.

//when input 'help', just print the help document content.

//when input 'quit', call system call 'exit' to leave program.

while true do

switch args[0]

case 'cd' then

call cd(args[1:])

case 'clr' then

call clr()

case 'dir' then

call dir()

case 'environ' then

call environ()

case 'echo' then

call echo(args[1:])

case 'help' then

display help text

case 'pause' then

call pause()

case 'quit' then

call 'exit' from system

default then

call run(args, pipe)

end

cd(args)//type is array

//"cd x/y/z"

// split args to get all subpath

// for subpath: '..', move to parent directory

// for subpath: '.', don't move

// for subpath: '...', move to grand-parent directory

currentPath <- get current path

for arg in args split by '/'

do

if arg = '..' then currentPath <- parent path

else if arg == '...' the currentPath <- grand-parent path

else if arg = '.' then currentPath <- pwd

else then currentPath <- go to path + arg

end

dir()

// List the file name in current directory

dir <- get current path

while dirpoint <- readdir dir not NULL

do display file name

end

clr()

//The way to clear the screen: just print 50 empty lines

// let the previous content leave the line of sight

for i < - 0 to 50

do display empty line

end

environ()

// we can call genenv system call to get all environs

// iterate every environ and display its value

envs <- read all current environs from system ‘getenv’

for env in envs

do display env

end

echo(args) //type is array

// just print input data to the screen.

display args

end

pause()

// keep waiting until you enter a 'enter'

// processor will block at library function getchar(), to wait input from stdin

// when you input 'enter', processor will continue to run, leave the while loop.

while true

do

ch <- get a charactor from stdin, call getchar()

if ch == '\n' then return

end

end

run(args)//type is array

// use fork() to fork a child processor, then child process will call execv() to

// run the command you input, child processor will write the results to pipe,

// then parent processor will read data from this pipe, and then display to screen.

pid <- fork a child processor

if pid == 0 then

child redirect stdout to pipe[1]

call exec(args) from system

else then

read from pipe[0]

**Testing Plan**

Writing a set of commands:

Run cd ../.../path/to

Then run pwd command, verifying the currency directory is expected directory.

Run echo $PATH and print $PATH

Then run environ and print $PATH again. If it printed the same, that mean run successfully.

Prepare a file test.txt and the context is:

1

2

3

4

5

6

And input the command: a.txt | sort > output.

If the result if 1 2 3 4 5 6, that fork/exec/pipe/redirect IO implement correctly.