CIS 3207
Project 2A.
Shell Pseudocode

Leomar Durán

19th February 2020

1 Main program

1. Start a loop while **Calling** prompting.
   (a) **Call** Parse the commandline string.
   (b) Set up the redirects for file input and output.
   (c) Depending on the first argument in the parsing:
      (i) Case "`cd`": **Call** Change the directory.
      (ii) Case "`clr`": **Call** Clear the screen.
      (iii) Case "`dir`": **Call** Print the directory listing. See 2020–02–19 (Shell, Week 2) (PPTX), slide 8 for the implementation.
      (iv) Case "`env`": **Call** Access the environment variable.
      (v) Case "`echo`": **Call** Echo the commandline.
      (vi) Case "`help`": **Call** Ask help with the current command.
      (vii) Case "`pause`": **Call** Pause until the user hits Enter.
      (viii) Default: **Call** Run an external command.
   (d) If parallel next, set up the next file to run concurrent to this one.
   (e) If pipe out, then wait before running the next file.
2. Loop back.

2 Prompt

1. Check the prompt environment variable.
2. Decode it and print it as appropriate.
3. Accept a commandline string from the prompt.

4. If the string contains an end-of-file character (Ctrl+`D`), is the string "`exit`", or the string "`quit`", return false.

5. Return true and the commandline string.

3 Process structure

- command name : string
- arguments : string[]
- input file : FILE*
- input append : Boolean
- output file : FILE*
- pipe in : Process*
- pipe out : Process*
- parallel next : Process*

4 Parse the commandline string

1. Let redirect characters := {" < ", " > ", "|", "&"}.
2. Create a queue of strings.
3. Initialize a left counter to zero.
4. Initialize a tokenizer state that can be "in text", "in space", "in redirect" to "in space".
5. Initialize a in-quote state that can be "outside quotes", "in double quotes" or "in single quotes" to "outside quotes".
6. Start a loop until the null character '`\0`' is reached.
    (a) Depending on the in-quote state:
        (i) Case "outside quotes":
            A. If the current character is a single quote:
                I. Change the state to "in single quotes".
            B. Otherwise, if the current character is in double quotes:
                I. Change the state to "in double quotes".
        (ii) Case "in single quotes":
            A. If the current character is a single quote:
                I. Change the state to "outside quotes".

B. Otherwise, skip this iteration of the loop.

(iii) Case "in double quotes":

A. If the current character is a double quote:

I. Change the state to "outside quotes".

B. Otherwise, skip this iteration of the loop.

(b) Depending on the tokenizer state:

(i) Case "in text":

A. If the current character is a space:

I. Change the state to "in space".

II. Copy a substring from the left counter to one less than the current character to the.

III. Enqueue the substring.

B. If the current character is in the list of redirect characters:

I. Remember the character.

II. Change the state to "in redirect".

(ii) Case "in space":

A. If the current character is a non-space character:

I. Change the state to "in text".

II. Set the left counter to the current character.

III. Stay on this character.

(iii) Case "in redirect":

A. If the current character is another redirect character:

I. Copy the characters from the original redirect to this character to redirect.

II. Depending on the original character, place this as an input file, output file, pipe out, or parallel next of the current file.

III. If this is a pipe out, mark the current file as its pipe in, and change to this file.

IV. If this is parallel next, change to this file.

7. Loop back.
8. Create an array with the same size as the queue.
9. While the queue has a next string.
   (a) Dequeue the item into the array.
10. Return the length and the starting position of the array.

5 Change the directory

1. Use `chdir` on the second argument.
2. If the result is nonzero, check `errno` and print the directory name and error message.

6 Clear the screen

1. Print the escape sequence "`\033[H\033[2J`".

7 Access the environment variable.

1. Depending on the number of arguments:
   (a) Case 1:
      (i) Loop through the environment variables and print them one per line.
   (b) Case 2:
      (i) Print the value of the environment variable specified by the second argument.
   (c) Case 3:
      (i) Replace the value of the environment variable specified by the second argument by the third argument.

8 Echo the commandline

1. Search for the first instead of "echo" in the original prompt.
2. Print every character one character after that until any other special characters, and a newline.

9 Ask help with the current command

1. Check if the second argument is an internal command.
   (a) If so, print a manual page for that command.
   (b) If not, **Call** Run the external help command.
2. Use the `more` filter to split up the pages.

10 Pause until the user hits [Enter]

1. Print "Please press [Enter] to continue . . ."
2. Loop until the user hits enter.
   (a) Get a character from the user.
   (b) Flush the input stream.
   (c) Check if the character was newline or carriage return.

11 Run an external command

1. If the command is not help:
   (a) Check if it exists in the current directory.
   (b) If it does not exist in the current directory:
      (i) Loop through the paths stored in the `$PATH` variable.
      (ii) Check each path for a command with that name until it is found.
2. Create a child process.
3. Launch the external command using the new child process in its appropriate directory.