COMP 304 Shellfyre: Project 1

Spring 2022

Sinan Cem Erdoğan / 68912

Part I - Basic Commands

For the basic commands, after path resolving the given command run with the exevc function in the child. If there is a background symbol (&) in front of the command, parent does not wait for the child to complete execution. Otherwise, it waits.

Part II - Custom Commands

- **filesearch**

For the filesearch command, a helper function called "fileSearch" was implemented. Function filesearch takes four arguments:
void fileSearch(char *keyword, char *current_dir, int recursive, int open);
char *keyword,
char *current_dir
int recursive
int open.

keyword is the keyword that command is looking for. current_dir is the current working directory at the time of calling filesearch command. recursive is the flag for recursive option, 1 means recursive 0 means not recursive. open is the flag for the open, if open is 1, it opens, if not it does not open. After calling filesearch , depending on the command line options, fileSearch function is called. fileSearch function find, prints, and open the files then return.

fileSearch opens current_dir directory using opendir. If the directory is "." or "..", it ignores. For the other directories it checks if it contains the keyword. If contains then it prints. If it is recursive then it resolves the path for the inner directories, open them, do the same comparing on them. If the open option is selected, it simply opens the directories using xdg-open. Finally, is closes the directory.

- **cdh**

For this command, there are four helper functions:

void printCdHistory(char *cdHistory[]);

void addCdToHistory(char *cd);

void writeToCdhFile();

void readFromCdhFile();

And three global variables:

Fixed sized string array (list) for keeping directory history,

char *cdHistory[10];

Index for reaching elements of the history list,

int cdCount = 0;

Path to directory in which shellfyre exist,

char pathToShellfyre[512];

After every cd command, current working directory is added to addCdToHistory and increment cdCount. After 10 cd is added to cdHistory array, addCdToHistory function moves every element +1 index and adds 11 cd to first index. Also, after changing the directory using cdh command, current working directory is added to cdHistory.

When calling cdh command, printCdHistory prints the cdHistory. According to input from the user, cdh changes the directory.

When extiting from the shellfyre, directory is changed to pathToShellfyre and writeToCdhFile called to store the cdHistory array. The reason for changing directory is to store the file in the same directory with the shellfyre.c.

When opening the shellfyre readFromCdhFile function read the stored cdh history and writes it to cdHistory array.

- **take**

  take command takes the argument and tokenize it using splitter "/". For every tokenized element, it makes directory using mkdir. After making the directory, it changes directory to lastly created directory. This process goes until tokens reaches to NULL. Upon changing directory, it adds the directories to the cdHistory.

- **joker**

  joker command first, saves the current working directory. The it changes directory to pathToShellfyre. The reason for this is to fix the location of the cronFile. After that it open a file and writes the time and command to run to a file. It calls crontab with the newly created cronFile. Lastly, changes the current working directory to where it called.

- **madmath**
  madmath command is basically a calculator for basic math operations as sum, factorial etc. The fun thing about this command is that it is mad. It does not want to be used as a simple calculator. Because of that the shell mock user's mathematical skills by sending sarcastic jokes as notifications. After every math operation, it randomly sends a sarcastic joke.

  Available operations: sum, sub, factor, pi, pow. Mod
  Usage: madmath [option] [number1] [number2]

Part III - Kernel Modules

Fort his part, a basic character device file implemented with the functionality of printing processes. The older character device files that are written by the author were used a template for this one. Since reading functionality is not need, there is no reading functionality. File operations works as expected. Dfs and bfs functions, starting from the given process id, prints the id and executable names of processes using dfs or bfs depending on the option.

Sources that was useful during implementation:

- https://stackoverflow.com/
- https://linuxhint.com/exec_linux_system_call_c/
- https://www.geeksforgeeks.org/
- https://tldp.org/LDP/lkmpg/2.4/html/c854.html
- https://github.com/coreutils/coreutils

Github : https://github.com/sinancemerdogan/Comp-304/tree/main/Project-1