Opertating Systems: Program 2

Introduction to the POSIX programming environment: Diagnostic Shell 2

Program Description: This program sets out to emulate a shell with added functions and convinience. This includes a more convinient kill command, easy access to a range of program statistics and the ability to track forked functions work load easily. Program is written in plain C.

Libraries used:
stdio.h: used for file and console io
stdlib.h: Need
string.h: needed for parsing strings and converting strings to integers
sys/types.h: used for Wait() and structures that waitid() uses.
Sys/wait.h: holds the wait() function. Needed for post fork output
unistd.h: Holds fork and exec functions
time.h: Needed for time functions
dirent.h : used for easy file io and checking contence of directories.
pthread.h : Used to create Pthreads and Join
sys/types.h : Structs for sockets
sys/socket.h : Socket Functions
netinet/in.h : Helper functions for Creating Socket
netdb.h: Helper functions for creating Socket

Program flow: Nearly everything stems from our event loop in main. The program sets up the functions we have predefined and prints the prompt. Program then waits for input and passes that to a parser and tokenizer. Those tokens are then passed to a function that determines to either use a built in function or pass it to another function that compiles the input for passing to exec. In the former the function is simply used. In the latter the program is forked and then the program waitid()'s for the program to finish so we can print out the time pagefaults and swaps for that process.

Functions:

Main(): contains main program event loop. Prints prompt, gets input executes.

Prompt(): prints dsh> prompt.

Findspace(): returns index to whitespace character in C-style string

execute(): Calls parse to tokenize the user input and checkdef to see if the desired program is predefined and then decides how to run the program. Frees data claimed in parse()

Parse(): Seperates and tokenizes and "   returns" the results given console input. Looks through the string finds whitespace and returns input as useful tokens.

checkdef(): checks the first input for defined functions.

Isitdigit(): For when isdigit() just didn't work well enough. Runs on entire c-strings

dircheck(): Given a name and directory to look in returns if that directory or file exists in that directory.

Cmdnm(): returns name used to run the program running under a given pid

mysignal(): sends signal to a process by pid. Only called mysignal because there's a c function named signal()

Systat(): Prints various system information.

Cd(): front end for the chdir function. Similar funtionality to cd. Should be able to change directory based on reletive and absolute paths.

Pwd(): calls spawns pwd function.

Definedrun(): For weird reasons needs to break if the first character is whitespace. Shouldn't happen but does on occasion. Otherwise calls function pointer from funtcion array.

Systemrun(): Parses input for use by the spawn() function. Contains various errorhandling mechanisms. Also makes sure the function you try to call is in /bin. If it isn't it won't attmpt to call it.

Printstat(): Outputs 10th, 14th, 15th and 36th item in file pointer. Used to output data after forked program termination.

Outspawn(); Forks and executes a new program and directs all standard output to the given file. WARNING: Can only use one redirect per command

inspawn(): Forks and executes a new program in the termianal with its standard input being replaced with and existing file. WARNING: Can only use one redirect per command

Pipespawn(): Forks 2 programs off of itself and pipes the output from the first to the input of the second using anonymous pipes. Returns process info about the second forked process. WARNING: Can only use one redirect per command

clientspawn(): Contains socket code from "http://www.linuxhowtos.org/C_C++/socket.htm" Uses sockets to receive input from over the internet and forks onto a standard Unix program. WARNING: Can only use one redirect per command

severspawn(): Contains socket code from "http://www.linuxhowtos.org/C_C++/socket.htm" Uses sockets to send output to a client on the internet and forks from a standard Unix program. WARNING: Can only use one redirect per command

hbthread(): The threaded function called from hb(). Counts down in microseconds and usleeps.

Cmdnmthread(); reads the file in /proc that holds the name of the command that created this process. Also its pthreadable

systatthread(): Prints the systatinformation but can do so in a threaded envirnment.

spawn(): Mostly stolen from code samples here (https://d2l.sdbor.edu/content/enforced/2015FA/777862-CSC_456_M001_2016SP_538486/Code_Examples.html). Although basic flow is unchanged the function is heavily altered for errortesting and printing child pid's and data on finished programs.

Compilation instructions: Terribly simple. Program was written to be compiled with the basic cc compiler in linux. No other dependencies should be nessisary Makefile is included to ease further however compilation can be achieved :

```
cc -c -pthreads dsh.c
cc -o -pthreads dsh dsh.o
```

Testing: Testing was through (given time avalable) and special casings were created where ever problems arose program should be able to pass mulitple flags even together. And whitespaces are mostly ignored although we are aware of some bugs.

Bugs: Bugs are rare. On Occation the remote pipe can fail for no decernable reason. And hb() sometimes will fail if it is the first command given and by fail I mean never stop printing.

Archive contence:
dsh.c: Contains the entirity of the program.
Makefile: contains explicit compilation instructions