Opertating Systems: Program 1

Introduction to the POSIX programming environment: Diagnostic Shell

Note: After speaking with you on Wednesday and regaining use of my computer Friday after class I realized that the program I had written was gone. I'm not ceirtan of what I did that caused my system to start deleting block records but I have learned 2 very important things. 1. The importance of backups/ versioning in a very real way. 2. Don't go slinging random signals at pids in the low hundreds with root privileges. After nearly three days attempting to recreate the code I had been working on the previous two weeks I have reached a point where I feel I am heavily pushing the bounds of any reasonable sympathy I deserve. As such, nearly all functionality has been implemented and the bug's i've had the time to test for will be documented.

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.

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.
WARNING: Crashes program if run twice, Not printing as pretty as it should, Uses other functions instead of /proc.

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 10^{th}, 14^{th}, 15^{th} and 36^{th} item in file pointer. Used to output data after forked program termination.
WARNING: In present testing condition data outputed by this function tends to contain wrong data for times. Thought these bugs also seem to exist in the proc file itself

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 dsh.c
cc -o 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: Systat crashes the second time you use it. No idea why.
On occasion the final argument in system program calls is corrupted. Has something to do with the dynamic memory but was unable to pinpoint problem.
Information from proc on finished programs is mostly there but the documentation is either lacking or not terribly well explained as the data is not always correct.
There is a warning on compilation regarding the exit function. Its in an array of function pointers that are different than it but is more convinient in program flow.
Occationally does not recognise the cd function. May be related to final argument corruption
If the fork function fails there isn't a graceful handeling mechanism.
Program begins exibiting VERY odd behavior if the shell is recompiled and re-run in itself, more so if done multiple times.
Not really a bug but as a thought I use spawn a lot when similar functionality could have been achived without forking.

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