dash - An Extended Specialized Shell Emulator

Name: Charles Parsons
Date: March 1, 2015
Course: CSC 456 - Operating Systems
Professor: Dr. Karlsson
Location: McLaury 306
Time: MWThF 2:00pm - 2:50pm

Description: This program emulates a shell prompt. It allows for most of the common shell commands, including but not limited to: cndnm, systat, cd, gcc/g++, ls, and piping. These commands perform as they are expected to perform in a normal shell prompt. This program uses fork to created child processes and execvp to call the functions to be run in those child processes.

Libraries used: fstream, iostream, sstream, string, cstdlib, cstring, fcntl.h, sys/resource.h, sys/time.h, sys/wait.h, unistd.h

Compiling instructions: use provided makefile with the command make.

Testing and verification: I ran many test cases, including the ones provided by Dr. Karlsson on the assignment write up. The program correctly responds to ¡enter¿ key presses, random entries, and correct entries. It detects invalid commands and prints a usage statement to the console. It also verifies the first argument of commands that require an additional argument.

Submission description: makefile, dash.cpp which contains the function main, commands.h which contains the #include statements and function prototypes for commands.cpp, and commands.cpp which contains the code for the command handling functions.

Functionality not implemented: pid, redirecting, signals.

Functions:
int main(int argc, char* argv[])
/* This function prints the dash prompt to the console, reads in the user * input, handles the exit command and ¡return¿ entries. When the command * entered is not empty or exit then it calls the interpretCommand function

* and sends the command and arguments to be processed. * */

void interpretCommand(std::string cmd)
/*takes in the raw command string from the command line and calls the appropriate * function. If no functions are matched then it is passed to the error message * function.*/

void commandName(int pid)
/* Deals with the cmdnm ¡pid¿ command by printing the name of the * command that created that process.*/

void systemStats()
/*Deals with the systat command. Prints cpu info, memory info, uptime, and version info.*/

void errorMessage()
/*Print error message*/

std::string itoa(int num)
/*Converts integer into string. * idea came from: * http://stackoverflow.com/questions/228005/ * alternative-to-itoa-for-converting-integer-to-string-c */

void pipedCommand(std::string cmd)
/*Handle command with piping. Uses piping example program modified to work with my program.*/

void parsePipedCommand(std::string cmd, std::string &input, std::string &output)
/*Parses the command string into the input section and output section of * the piped commands.*/