File Documentation

---

bash.c code structure

Headers

- #include <stdio.h>
- #include <stdlib.h>
- #include <string.h>
- #include <limits.h>
- #include <unistd.h>
- #include <sys/wait.h>
- #include <signal.h>
- #include <errno.h>
- #include <sys/types.h>
- #include <sys/stat.h>
- #include <dirent.h>

Data Structures

- struct node

Macros

- #define BUFFSIZE 1024
- #define TOKSIZE 16
- #define DELIMETERS " \t\n\r"
- #define YELLOW "\033[1;33m"
- #define NC "\033[0m" // No Color
- #define HISTSIZE 10

Functions

- void push_command (char *cmd)
- void pop_command ()
- char * get_input ()
- char ** tokenize (char *cmd, char *delimeter)
- void history ()
- void free_commands ()
- void kill_child(int sig)
- void execute_command (char *cmd, int in, int out)
- int is_io_redirection (char **tokens)
- int copy (char *source, char *dest, int c)
- void io_handler (char *cmd, int in, int ot)
- void pipe_handler (char *cmd)
- void bash ()
- int main ()

Typedefs

- typedef struct node node

Globals

- node * command_stack = NULL
- pid_t pid = -1

Macro Definition Documentation

- #define BUFFSIZE **1024**
- #define DELIMETERS **" \t\n\r"**
- #define HISTSIZE **10**
- #define NC **"\033[0m"**
- #define TOKSIZE **16**
- #define YELLOW **"\033[1;33m"**

Typedef Documentation

- typedef struct node node
  - Node-Data structure for storing user commands in history

Function Documentation

- void bash ()
  - function which prompts user for command and call appropriate functions.
- int copy (char * source, char * dest, int c)
  - utility function for copying strings.
- void execute_command (char * cmd, int in, int out)
  - function which executes the user command by identifying the first token of it.
  - All bulletin commands are run in bash itself and it figures this by various if else statement.
  - Here it runs all the commands which have their executable in /bin folder.
  - here child process runs and executes the command.
  - duplicating the file descriptor of stdin and 'in' for piping use.
  - duplicating the file descriptor of stdout and 'out' for piping use.
  - commands are run by the execvp function which have their executable in /bin directory.
  - here parent process runs and wait for the child process to terminate.
  - freeing the memory used for storing tokens.
- void free_commands ()
  - function to free the allocated memory in heap.

- char* get_input ()
  - function for getting user input
  - takes input character by character till EOF is encountered or ‘\n’ is encountered.
- void history ()
  - prints the recent 10 commands entered by user.
- void io_handler (char * cmd, int in, int ot)
  - Function to handle the I/O redirection.
  - have I/O redirection in user command.
  - opens file corresponding to I/O redirection. and sets corresponding file descriptors for duplicating.
  - concating tokens to make user command.
  - when I/O redirection is not there.
- int is_io_redirection (char ** tokens)
  - function to check whether there is I/O redirection in the user command or not. Return 1 if true else 0.
- int main ()
  - to call bash() function
- void pipe_handler (char * cmd)
  - tokenize the user command by '|' and creates a pipe to run each command one by one.
  - sets file descriptor for first use. First it should read from stdin.
  - creating pipe and passing file descriptor for output as f[1].
  - setting input file descriptor for next iteration.
  - handling the last command.
- void pop_command ()
  - function for deleting recent command in the linked list
- void push_command (char * cmd)
  - function for inserting user command in the linked list
- char** tokenize (char * cmd, char * delimeter)
  - function to parse the user command and tokensize it by given delimeter.
- void kill_child(int sig)
  - function used to kill child process, it is used in <program_name> no_of_seconds command

---

Global Documentation

- node* command_stack = NULL
  - pointer of latest command
- pid_t pid = -1
  - Used for <program_name> no_of_seconds command to kill child after given number of seconds