KOÇ UNIVERSITY

COMP 304: Operating Systems

Project - 1

Spring 2023

Ali Gebeşçe, 64294

Yakup Enes Güven, 64045

KOÇ UNIVERSITY

Table of Contents

Important Note **3**

A. Part I **3**

B. Part II **3**

C. Part III **3**

D. Part IV **3**

E. References **3**

Important Note

Please note that all project components, except for custom commands, were completed by both team members during in-person and online meetings. However, only one team member's computer was used to commit the corresponding work after these meetings. As a result, Git commits do not accurately reflect individual contributions. Nevertheless, we made an effort to ensure that both team members' Git accounts contributed equally in terms of the weight of points for the project.

A. Part I

- cd

```
if (strcmp(command->name, "cd") == 0) {
    add_dir(strdup(command->args[1]));
    if (command->arg_count > 0) {
        r = chdir(command->args[1]);
        if (r == -1) {
            printf("-%s: %s: %s\n", sysname, command->name,
                   strerror(errno));
        }
        return SUCCESS;
    }
}
```

```
yakup@admin:~/Desktop/project-1---shell-bisey-master$ make
make: 'mishell' is up to date.
yakup@admin:~/Desktop/project-1---shell-bisey-master$ ./mishell
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ cd src
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$
```

- exit

```
if (strcmp(command->name, "exit") == 0) {
    return EXIT;
}
```

```
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$ exit

yakup@admin:~/Desktop/project-1---shell-bisey-master$
```

B. Part II

-

C. Part III

1. Dice Roll

```
if (strcmp(command->name, "roll") == 0) { // roll command

    if (command->arg_count < 3 ||
        command->arg_count > 3) { // if there are not 3 arguments
        printf("roll: Works with one argument: %s 'digit'd'digit'\n",
               command->name); // print error
        return UNKNOWN; // return unknown
    } else { // if there are 3 arguments
        int num_rolls = 1; // number of rolls
        int num_sides; // number of sides on dice
        char *token =
            strtok(command->args[1], "d"); // split string into tokens
        if (atoi(&token[2]) == 0) { // if the first token is a number
            num_sides =
                atoi(&token[0]); // set number of sides to first token
        } else { // if the first token is not a number
            num_rolls = atoi(token); // set number of rolls to first token
            token = strtok(NULL, "d"); // split string into tokens again
            num_sides = atoi(token); // set number of sides to second token
        }
        int result = 0; // result of roll
        int *rolls; // array of rolls
        rolls = (int *)malloc(num_rolls *
                              sizeof(int)); // allocate memory for rolls
        for (int i = 0; i < num_rolls; i++) { // for each number of rolls
            rolls[i] = rand() % num_sides + 1; // roll dice
            result += rolls[i]; // add roll to result
        }
```

-

```
        // If rolled once
        if (num_rolls == 1) {
            printf("Rolled %d", result); // print result
        } else { // if rolled more than once
            printf("Rolled %d (", result); // print result
            for (int i = 0; i < num_rolls; i++) { // for each roll
                printf("%d", rolls[i]); // print roll
                if (i < num_rolls - 1) { // if there are more rolls
                    printf(" + "); // print plus sign
                }
            }
            printf(")\n"); // print new line
        }
        free(rolls); // free memory
        return SUCCESS; // return success
    }
}
```

```
yakup@admin:~/Desktop/project-1---shell-bisey-master$ ./mishell
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ roll
roll: Works with one argument: roll 'digit'd'digit'
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ roll 3d6
Rolled 11 (2 + 5 + 4)
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ roll d6
Rolled 2
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$
```

2. **cd History**

```
if (strcmp(command->name, "cdh") == 0) { // cdh command
    if (command->arg_count > 0) {
        print_dirs();
        char ch;
        printf("Select directory by letter or number: ");
        scanf("%c", &ch);
        //printf("You entered the character '%c'.\n", ch);
        //printf("dirs '%s'.\n", dirs[1]);
        navigate_dir(ch, dirs);
        return SUCCESS;
    }
}
```

-

```c
// This function prints a list of recent directories.
void print_dirs() {
    // Print a heading for the list.
    printf("Recent directories:\n");
    // Loop through the array of directories.
    for (int i = 0; i < num_dirs; i++) {
        // Print the index of the directory (as a letter) and its name.
        printf("%c %d) %s\n", 'a' + i, i + 1, dirs[i]);
    }
}
```

```c
void navigate_dir(char ch, char *dirs[]) { // Define a function called navigate_dir that takes a char
    printf("You entered the dir value '%c'.\n", ch); // Print a message to the console that displays

    int index = -1; // Initialize the index variable to an invalid value

    if (ch >= 'a' && ch <= 'i') { // Check if the character is in the range of a to i
        index = ch - 'a' + 1; // Calculate the index based on the ASCII value of the character
    }
    if (ch >= 49 && ch <= 59) { // Check if the character is in the range of 1 to 9
        index = ch - 49 + 1; // Calculate the index based on the ASCII value of the character
    }

    char cwd[1024]; // Define a character array to store the current working directory

    if (getcwd(cwd, sizeof(cwd)) != NULL) { // Get the current working directory and check if it's va
        printf("Current working dir before: %s\n", cwd); // Print a message to the console that displ
    } else {
        perror("getcwd() error"); // Print an error message to the console if getcwd() fails
    }

    chdir(dirs[index-1]); // Change the current working directory to the directory at the calculated
    if (getcwd(cwd, sizeof(cwd)) != NULL) { // Get the current working directory and check if it's va
        printf("Current working dir after: %s\n", cwd); // Print a message to the console that displa
    } else {
        perror("getcwd() error"); // Print an error message to the console if getcwd() fails
    }
}
```

```c
void add_dir(char *dir) { // Define a function called add_dir that takes a pointer to a character as its argument
    //printf("Added directory: %s\n", dir); // Commented out line of code that prints a message to the console
    if (num_dirs == MAX_DIRS) { // Check if the number of directories is equal to the maximum allowed directories
        free(dirs[num_dirs - 1]); // Free the memory allocated for the last directory added to the array
        num_dirs--; // Decrement the number of directories
    }
    for (int i = num_dirs; i > 0; i--) { // Iterate over the directories in the array, starting from the last one
        dirs[i] = dirs[i - 1]; // Move each directory up one index in the array
    }
    if (strcmp(dir, "..") == 0) { // Check if the directory being added is ".."
        dirs[0] = "project-1---shell-bisey"; // Set the first directory in the array to a specific value
        //printf("The directory is equal to: %s\n", dirs[0]); // Commented out line of code that prints a message to the console
    } else {
        dirs[0] = dir; // Set the first directory in the array to the value of the argument passed to the function
        //printf("The directory is equal to: %s\n", dirs[0]); // Commented out line of code that prints a message to the console
    }
    num_dirs++; // Increment the number of directories in the array
}
```

```
yakup@admin:~/Desktop/project-1---shell-bisey-master$ ./mishell
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ cd src
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$ cd ..
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ cd src
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$ cd ..
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ cdh
Recent directories:
a 1) project-1---shell-bisey
b 2) src
c 3) project-1---shell-bisey
d 4) src
Select directory by letter or number: b
You entered the dir value 'b'.
Current working dir before: /home/yakup/Desktop/project-1---shell-bisey-master
Current working dir after: /home/yakup/Desktop/project-1---shell-bisey-master/src
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$ cd ..
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ cdh
Recent directories:
a 1) project-1---shell-bisey
b 2) project-1---shell-bisey
c 3) src
d 4) project-1---shell-bisey
e 5) src
Select directory by letter or number: 5
You entered the dir value '5'.
Current working dir before: /home/yakup/Desktop/project-1---shell-bisey-master
Current working dir after: /home/yakup/Desktop/project-1---shell-bisey-master/src
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master/src mishell$
```

3. **Count Lines of Code**

```c
if (strcmp(command->name, "cloc") == 0) {
    if (command->arg_count < 3 || command->arg_count > 3) {
        printf("cloc: Works with one argument: %s '/directory'\n",
               command->name);
        return UNKNOWN;
    } else {
        int file_count = 0, processed_count = 0;
        int total_blanks = 0, total_comments = 0, total_code = 0;
        int c_blanks = 0, c_comments = 0, c_code = 0;
        int cpp_blanks = 0, cpp_comments = 0, cpp_code = 0;
        int py_blanks = 0, py_comments = 0, py_code = 0;
        int txt_blanks = 0, txt_comments = 0, txt_code = 0;

        DIR *dir = opendir(command->args[1]);
        if (dir == NULL) {
            perror(command->args[0]);
            return 1;
        }
        struct dirent *entry;
        while ((entry = readdir(dir)) != NULL) {
            if (is_dotfile(entry->d_name)) {
                continue;
            }

            char path[512];
            snprintf(path, sizeof(path), "%s/%s", command->args[0],
                     entry->d_name);

            if (is_binary(path)) {
                continue;
            }

            ++file_count;
            int blanks, comments, code;
            if (count_lines(path, ".c", &blanks, &comments, &code)) {
                printf(".c counting");
                ++processed_count;
                c_blanks += blanks;
                c_comments += comments;
                c_code += code;
```

-

```
            } else if (count_lines(path, ".cpp", &blanks, &comments,
                                   &code)) {
                printf(".cpp counting");
                ++processed_count;
                cpp_blanks += blanks;
                cpp_comments += comments;
                cpp_code += code;
            } else if (count_lines(path, ".py", &blanks, &comments,
                                   &code)) {
                printf(".py counting");
                ++processed_count;
                py_blanks += blanks;
                py_comments += comments;
                py_code += code;
            } else if (count_lines(path, ".txt", &blanks, &comments,
                                   &code)) {
                printf(".txt counting");
                ++processed_count;
                txt_blanks += blanks;
                txt_comments += comments;
                txt_code += code;
            }
            total_blanks += blanks;
            total_comments += comments;
            total_code += code;
        }

        printf("%d text files.\n", file_count);
        printf("%d unique files.\n", processed_count);
        printf("%d files ignored.\n", file_count - processed_count);
        printf(
            "-------------------------------------------------------------------------------\n");
        printf(
            "Language          files             blank       comment             code\n");
        printf(
            "-------------------------------------------------------------------------------\n");
        printf(
            "C                 %5d           %5d           %5d           %5d\n",
            c_blanks + c_comments + c_code, c_blanks, c_comments, c_code);
        printf(
            "C++               %5d           %5d           %5d           %5d\n",
            cpp_blanks + cpp_comments + cpp_code, cpp_blanks, cpp_comments,
            cpp_code);
```

```
        printf(
            "Python            %5d           %5d           %5d           %5d\n",
            py_blanks + py_comments + py_code, py_blanks, py_comments,
            py_code);
        printf(
            "Text              %5d           %5d           %5d           %5d\n",
            txt_blanks + txt_comments + txt_code, txt_blanks, txt_comments,
            txt_code);
        printf(
            "-------------------------------------------------------------------------------\n");
        printf(
            "Total             %5d           %5d           %5d           %5d\n",
            total_blanks + total_comments + total_code, total_blanks,
            total_comments, total_code);
        printf(
            "-------------------------------------------------------------------------------\n");
        return 0;
    }
}
```

```
yakup@admin:~/Desktop/project-1---shell-bisey-master$ ./mishell
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ cloc src
1 text files.
0 unique files.
1 files ignored.
-------------------------------------------------------------------------------
Language          files            blank       comment            code
-------------------------------------------------------------------------------
C                     0                0              0               0
C++                   0                0              0               0
Python                0                0              0               0
Text                  0                0              0               0
-------------------------------------------------------------------------------
Total            -321071041              0          -321136576          65535
-------------------------------------------------------------------------------
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$
```

4. **Custom Commands**
   a. **Ali**
      -
   b. **Yakup**

```c
if (strcmp(command->name, "bomb") == 0) { //kill your mishell
    //printf("%d\n",command->arg_count);
    if (command->arg_count < 3 ||
        command->arg_count > 3) { // if there are not 3 arguments
        printf("bomb: Works with one argument: %s <duration in seconds>\n",
               command->name);
        return UNKNOWN;
    } else {
        //printf("%s\n",command->args[0]);
        //printf("%s\n",command->args[1]);
        int dur = atoi(command->args[1]);
        mishell_timebomb(dur);
    }
}
```

- **bomb**

```c
// Function to handle the timebomb
void timebomb_handler() {
    kill(getppid(), SIGKILL); // sends SIGKILL signal to the parent process
    system("logout"); // executes the "logout" command to log out the current user
}

// Function to activate a timebomb that will terminate the parent process after a certain duration
int mishell_timebomb(int dur) {
    signal(SIGALRM, timebomb_handler); // registers the timebomb_handler function to be called when the alarm signal is received
    alarm(dur); // sets an alarm that will trigger the timebomb_handler function after the given duration
    for (int i = dur; i >= 0; i -= 1) {
        printf("WARNING: bomb active! %d seconds remaining.\n", i); // displays a warning message to the user with the remaining time
        sleep(1); // waits for 1 second
    }
    return SUCCESS; // returns a constant value to indicate success
}
```

```
yakup@admin:~/Desktop/project-1---shell-bisey-master$ ./mishell
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ bomb 10
WARNING: bomb active! 10 seconds remaining.
WARNING: bomb active! 9 seconds remaining.
WARNING: bomb active! 8 seconds remaining.
WARNING: bomb active! 7 seconds remaining.
WARNING: bomb active! 6 seconds remaining.
```

D. Part V

-

```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/list.h>
#include <linux/pid.h>
#include <linux/sched.h>
#include <linux/slab.h>
#include <linux/types.h>
#include <linux/unistd.h>
#include <linux/proc_fs.h>


// Meta Information
MODULE_LICENSE("GPL");
MODULE_AUTHOR("bisey");
MODULE_DESCRIPTION("A simple module to print the process tree of a given PID"); // Description of the module

int pid; // PID of the process to be monitored
module_param(pid, int, S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH); // Set the PID of the process to be monitored
MODULE_PARM_DESC(pid, "PID of the process"); // Description of the PID parameter


void print_dots(int depth){ // Print the dots
    int i; // Loop variable
    for(i = 0; i < depth; i++){ // Print the depth of the process
        printk(KERN_CONT "*****");
        if (i == depth-1 ){ // Print the arrow
            printk(KERN_CONT "---->");
        }
    }
}
void print_process_tree(struct task_struct* ts, int depth){ // Traverse the process tree
    struct list_head *list_; // List of children
    struct task_struct *child; // Child process
    print_dots(depth); // Print the dots
    printk(KERN_CONT "%d - PID: %d (start time: %lld)\n", depth, ts->pid, ts->start_time); // Print the PID and start time of the process

    list_for_each(list, &ts->children){ // Traverse the children of the process
        child = list_entry(list, struct task_struct, sibling); // Get the child process
        print_process_tree(child, depth+1); // Traverse the child process
    }

}
```

```c


int psvis_start(void){ // Start the module
    struct task_struct *ts; // Task struct of the process to be monitored
    ts = get_pid_task(find_get_pid(pid), PIDTYPE_PID); // Get the task struct of the process to be monitored
    if (ts != NULL){ // If the process exists
        print_process_tree(ts, 0); // Traverse the process tree
    }
    return 0; // Return 0
}

void psvis_finish(void){ // Finish the module
    pid = 0; // Reset the PID value
    printk("FINISHED psvis\n", ); // Print the finish message
}



module_init(psvis_start);
module_exit(psvis_finish);
```

```
yakup@admin:~/Desktop/project-1---shell-bisey-master$ ./mishell
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ psvis 19784 output.txt
-mishell: psvis: Success
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ ps
    PID TTY          TIME CMD
  17990 pts/2    00:00:00 bash
  19831 pts/2    00:00:00 mishell
  20134 pts/2    00:00:00 ps
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$ psvis 20134 output.txt
-mishell: psvis: Success
yakup@admin:/home/yakup/Desktop/project-1---shell-bisey-master mishell$
```

E. References

1. https://devarea.com/linux-kernel-development-and-writing-a-simple-kernel-module/
2. https://www.informit.com/articles/article.aspx?p=368650
3. **https://elixir.bootlin.com/linux/latest/source**
4. https://www.tldp.org/LDP/lkmpg/2.4/html/x354.htm
5.