Operating system

Lab 1|simple Unix shell

Omar khalid mohamed 160

Omar Nasr younis 173

Mohamed Diaa Eldin 233

4/13/2020

➢ Functions Documentation

Function name	Description
read_parse_line()	This functions takes two arguments , string which stores the command line and array of strings the command line after splitting. It returns background state.
`read_line()`	This function takes one argument , string that receives the command line from the user and process it to remove enter from the end of the command line and replace it by NULL.
`process_line()`	This functions takes two arguments , string which stores the command line and array of strings the command line after splitting. It splits the command line to substrings that command execution function "execvp" can deal with.
check_background()	This function take the command line string to Check if the process is to be executed in background or not.
execute()	The execute function takes three arguments. The first argument is the main command, the second is a null terminated array if the command and its token, the third is an integer that indicates whether the command to be executed in the background or not. The function forks a child process assigning an identification to each process. For the child process, it executes the command. The main process waits for the child process to be terminated or continues execution if the child process is a background process

➢ Functions implementation

- execute

```
52  void execute(char* command, char *str[MAX_WORD], int IsBackground){
53      pid_t pid, pid_done; //process ID
54      int status;
55
56      pid = fork();
57  //For the CHILD PROCESS pid = 0,
58  //For PARENT PRROCESS pid > 0,
59  //pid < 0 if CHILD PROCESS failed to be created.
60
61      if(pid < 0){                                    //No child process created
62          printf ("FAILED TO CREATE CHILD PROCESS");
63          exit(666);                                  //exit program with error code 666
64      } else if (pid == 0){                           //For the child process, execute this
65          execvp(command, str);
66          exit(0);
67      } else {                                        //For the parent process, execute this
68          if (IsBackground) pid_done = wait(&status);
69      }
70  }
```

- read_parse_int()

```
43  int read_parse_line(char*str[],char line[]){
44      int IsBackground;
45
46      read_line(line);
47      IsBackground = check_background(line);
48      process_line(str,line);
49      return IsBackground;
50  }
```

- read_line()

```
11  void read_line(char line[]){
12      int i = 0 ;
13      fgets(line,MAX_CHAR,stdin);
14      //remove enter
15       while(line[i]!='\n')
16          i++;
17      line[i] ='\0';
18  }
```

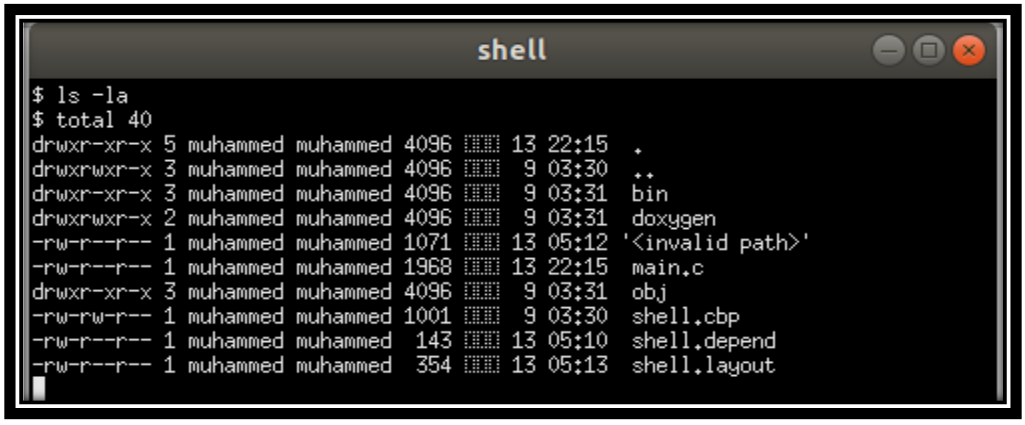
- check_background()

```
20  //Check if the process is to be executed in background or not
21  int check_background(char line[]){
22      char* ret;
23
24      ret = memchr(line, '&', strlen(line));
25      if (ret == NULL) return 0;
26      else return 1;
27  }
```

- process_line()

```
29  int process_line(char*str[],char line[]){
30      int i =0 ;
31      str[i] = strtok(line, " ");
32
33      if(str[i] == NULL) return 1 ;
34
35      while(str[i] != NULL){
36          i++;
37          str[i] = strtok(NULL," ");
38      }
39      return 1 ;
40  }
```

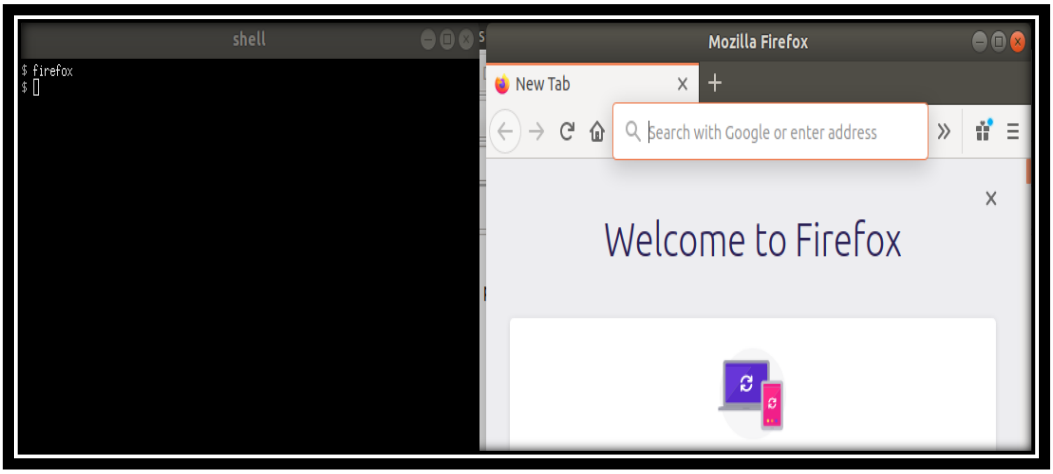
- Run samples
  - ls –la



  - pwd

  - clear

- firefox



➢ the processes hierarchy in KSysguard

Process Name	Virtual Mem	% CPU	CPU Time	Memory	Session
systemd	79.8 MiB	0	0:01.57	4.8 MiB	
codeblocks	1.2 GiB	1	0:20.16	71.8 MiB	
xterm	90.9 MiB	0	0:00.36	3.9 MiB	
cb_console_runner	13.7 MiB	0	0:00.01	144.0 KiB	
sh	4.5 MiB	0	0:00.00	68.0 KiB	
sell	4.4 MiB	0	0:00.01	60.0 KiB	
gedit	733.9 MiB	0	0:02.24	15.2 MiB	
firefox	2.8 GiB	0	0:33.89	118.5 MiB	
Web Content	2.5 GiB	0	0:08.75	40.8 MiB	