“**UNIX Shell and Program Selection Feature** “

“**Readme File** “

“**Computer Science** “

“**Course: CPSC-503-12-Operating Systems** “

“**Introduction:** “

“In this part we have modify the shell interface program from part A so that it provides “a program selection feature that allows the user to access the available executable programs “from the current directory. “

“The user will be able to access all executable programs, i.e., all files that a user has executable “permission, from a current directory by using the feature. “

“

“**Steps to run this program:** “

1. “copy attached **prog\_pt\_b.c** file to desktop of any UNIX environment. “
2. “open Terminal. “
3. “change directory to Desktop. “
4. “run **gcc -o prog\_pt\_b prog\_pt\_b.c** command to compile the code from the file. “
5. “run **./program\_pt\_b** command to run the program. “

**“Assumptions: “**

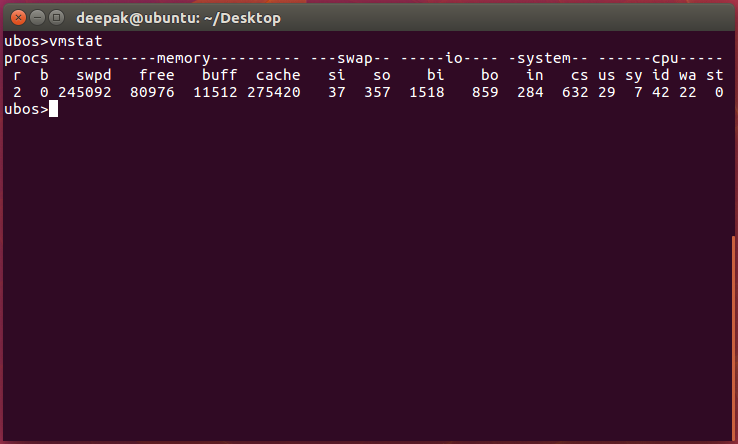
“In this part we have implement the selection() function that lists the available executable file of “the current directory in the alphabetical order with proper numbering. The user can select their “choice from the list to execute the program. “

* “We have used opendir() function to open the current directory which allows the user to “access the available executable programs from the current directory. “
* We have checked the opendir() i.e. if NULL the error message is thrown otherwise “directory access is granted and closedir() is called.
* “For sorting the executable file, we have implemented the quicksort which list all the “executable file by alphabetical order starting at 1. Then the user can enter the number “available to select the specific file to execute.
* For error handling in the program if count is 0, “No executable file in the directory.” “message is thrown.

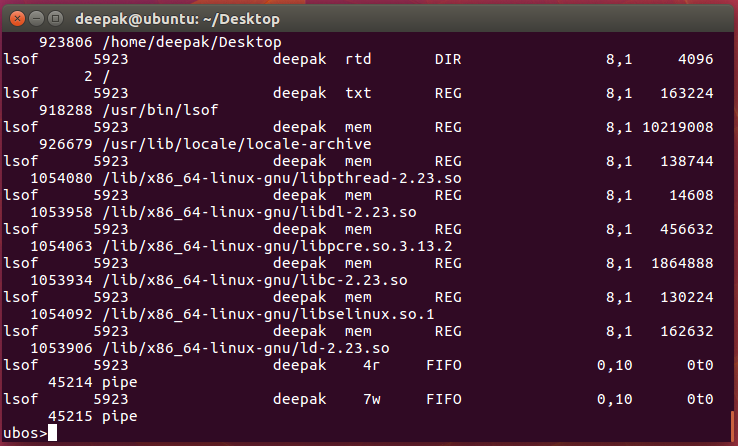
“**Analysis of performance measure with various parameter values: “**“

**“**To analyze the program, we can run various code on the ubos> shell. The command with **“**screenshot is given below: **“**

1. **“vmstat:** It displays static of virtual memory, io blocks and cpu activity. **“**



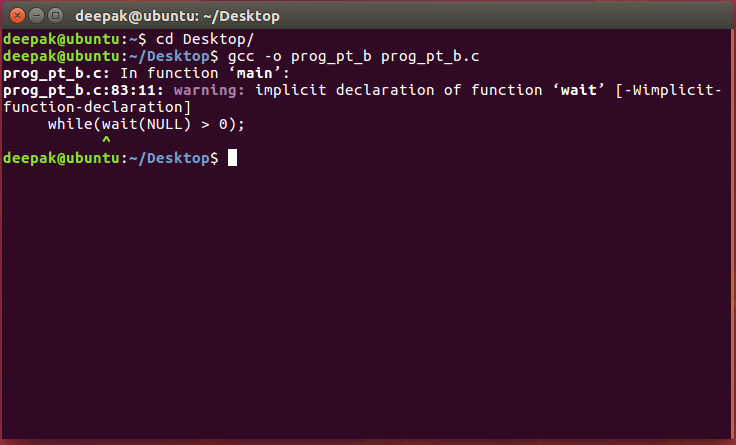
1. **“lsof:** It displays list of open files and processes. **“**

****

**“Bugs and Limitations: “**

**“**The code runs well and there are no any bugs encountered by the compiler. **“**

**“**There is only one warning which is shown in the screenshot below: **“**



“

“**Code:** ““

“/\* Name: Deepak Bhattarai ID: 1006053 \*/“

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

“#include <string.h>“

“#include <dirent.h>““

#include <sys/stat.h>

#define MAX\_LINE 80 /\* 80 chars per line, per command \*/

“void selection();“

“//Main function“

“int main()“

“{“

“ “char \*args[MAX\_LINE/2 + 1]; /\* command line (of 80) has“

“ max of 40 “arguments \*/“

“char line[MAX\_LINE]; “

“ int should\_run = 1; “

“ int isBackground = 0; “

“ while (should\_run) /\* while loop \*/“

“ { “

“printf("ubos>"); “

fflush(stdout);

“fgets(line, MAX\_LINE, stdin); /\* reads user input \*/“

“ if(line[strlen(line)-1] == '\n') “

“{“

“ line[strlen(line)-1] = '\0'; “

“}“

“ if(strcmp(line, "exit") == 0) /\* exit if statement \*/“

“ {“

“ should\_run = 0; “

“ exit(0); “

“}“

“/\* Parse the input taken from user \*/““

“ “int cnt = 0; ““

“ “ char \*token = strtok(line, " ");““

“ while(token != NULL) “

“ {“

“ args[cnt] = (char\*)malloc(sizeof(strlen(token) + 1)); “

“ strcpy(args[cnt], token); “

“ ++cnt; “

“ token = strtok(NULL, " ");“

“ }“

“ if(cnt > 0) “

“ {“

“ if(strcmp(args[cnt-1], "&") == 0) “

“ {“

“ free(args[cnt-1]); “

“ args[cnt-1] = NULL; “

“ isBackground = 1; “

“ }“

“ else“

“ {“

“ args[cnt] = NULL; “

“ }“

}

“ pid\_t pid = fork();“ /\* child process creation \*/“

“if(pid == 0) “

“ {“

“ /\* Part B: Selection \*/“

“ if(cnt == 1 && strcmp(args[0], "selection") == 0) “

“ {“

“ selection();“

“ }“

“ else “

“ {“

“ execvp(args[0], args); “

“ }“

“ }“

“ else if(pid > 0) /\* parent process \*/“

“ {“

“ /\* if command included & parent will invoke wait() \*/ “

“ “{““

“ “while(wait(NULL) > 0); ““

“ “ }“

}

“ “else /\* error handling \*/““

{

“ “printf("Fork Error!\n");““

}

}

“return 0; “

}

“int cmp(const void \*p1,const void \*p2) /\* compare function \*/“

“{“

“ return strcmp((char \*)p1, (char \*)p2); “

“}“

“void selection()“

“{“

“ DIR \*dp; “

“ struct dirent \*entry; “

“ struct stat statbuf; “

“ char filename[100][MAX\_LINE]; “

“ int cnt = 0; “

“ /\* to open directory \*/“

“ if(( dp = opendir(".")) == NULL) /\* for error handling \*/“

“ {“

“ fprintf(stderr, "Cannot open current directory\n");“

“ return; “

“ }“

“ while((entry = readdir(dp)) != NULL) “

“ {“

“ lstat(entry->d\_name, &statbuf); “

“ “if(strcmp(entry->d\_name, ".") != 0 && “

“strcmp(entry->d\_name, "..") “!= 0 && statbuf.st\_mode & S\_IXUSR) ““

“{“

“ strcpy(filename[cnt++], entry->d\_name); “

“ }“

}

closedir(dp);

“ qsort(filename, cnt, MAX\_LINE, cmp); /\* sorting the executable “files \*/“

“ if(cnt > 0) “

“ {“

“ printf("The output of selection command will be: \n");“

“ }“

“ /\* list the executable file by order \*/“

“ int i; “

“for(i = 0 ; i < cnt; ++i) “

“ {“

“ “printf("\t%d\t%s\n", i+1, filename[i]); ““

“ }“

“ “if(cnt == 0) /\* for error handling \*/““

“ {“

“ printf("No executable file in directory.\n");“

“ }“

“ else“

“ {“

“ printf("Select a file to execute: "); /\* read user input \*/“

“ int index; “

“ if(scanf("%d", &index) < 1 || index <= 0 || index > cnt) /\* for “error handling \*/“

“ {“

“ printf("File select error!\n");“

“ }“

“ else“

“ {“

“ char \*args[2]; “

“ args[0] = (char\*)malloc(strlen(filename[index-1] + 1)); “

“ strcpy(args[0], "./");“

“ strcat(args[0], filename[index-1]); “

“ args[1] = NULL; “

“ execvp(args[0], args); /\* to execute the file \*/“

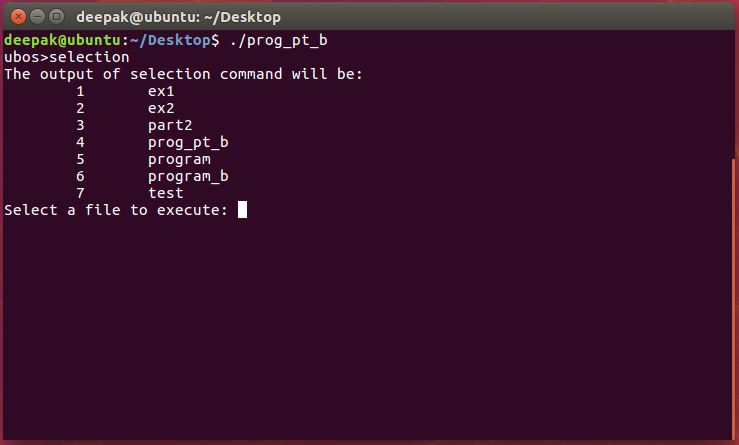
“ }“

“ }“

“}“

“**Screenshot:** “

1. “**selection command:** The output of selection command is: “



1. “**Selecting ex2 file:** Runs the program by the selection of number 2. “

