

**Module Code & Module Title**

**CS5001NA Networks and Operating System**

**Assessment Weightage & Type**

**20% Individual Coursework**

**Year and Semester**

**2020-21 Autumn**

**Student Name: Umesh Shahu Shrestha**

**London Met ID: 20049086**

**College ID: NP01CP4S210288**

**Assignment Due Date: 25 April 2022**

**Assignment Submission Date: 25 April 2022**

**Word Count (TASK B): 2152 words**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

**Table of Content**

[1. Task A 1](#_Toc101779019)

[1.1) INTRODUCTION 1](#_Toc101779020)

[1.2) Script 2](#_Toc101779021)

[1.3) Testing 10](#_Toc101779022)

[1.4) Contents of three files. 29](#_Toc101779023)

[ JL 29](#_Toc101779024)

[ AY 29](#_Toc101779025)

[ FM 29](#_Toc101779026)

[1.5) Conclusion 30](#_Toc101779027)

[2. Task B 31](#_Toc101779028)

[2.1) Introduction 31](#_Toc101779029)

[2.2) Background 32](#_Toc101779030)

[2.2.1 Process Architecture 32](#_Toc101779031)

[2.2.2 Process Control Blocks 33](#_Toc101779032)

[2.2.3 the Process States 35](#_Toc101779033)

[2.2.4 Process Hierarchies 38](#_Toc101779034)

[2.2.5. Implementation of Processes 38](#_Toc101779035)

[2.2.6 Threads 39](#_Toc101779036)

[2.3) Conclusion 39](#_Toc101779037)

[3. References 41](#_Toc101779038)

[4. Appendix 42](#_Toc101779039)

[4.1) Appendix -A(Glossary) 42](#_Toc101779040)

[4.2) Appendix – B (Process Scheduling) 42](#_Toc101779041)

[4.3) Appendix – C (Priority Scheduling) 42](#_Toc101779042)

Table of figure

[Figure 1: Displaying error message "Program can't execute without username and Id" 10](file:///C:\Collegecourse\Networking%20and%20operating%20system\Coursework%202\20049086%20Umesh%20Shahu%20Shrestha\part2\20049086%20Umesh%20Shahu%20Shrestha.docx#_Toc101777257)

[Figure 2: run with username and id 11](#_Toc101777258)

[Figure 3: Running an incorrect password three times 12](#_Toc101777259)

[Figure 4: Running correct password part 1 13](#_Toc101777260)

[Figure 5: Running correct password part 2 14](#_Toc101777261)

[Figure 6: Displaying band name. 15](#_Toc101777262)

[Figure 7: Running incorrect band Code part 1. 16](#_Toc101777263)

[Figure 8: Running incorrect band Code part 2. 17](#_Toc101777264)

[Figure 9: Running correct band code part 1. 18](#_Toc101777265)

[Figure 10: Running correct band code part 2. 18](#_Toc101777266)

[Figure 11: Picking a 4-member name. 21](#_Toc101777267)

[Figure 12: Running wrong user id. (Parameter validation) 23](#_Toc101777268)

[Figure 13: Inserting the right user Id which has an external file of a member. 24](#_Toc101777269)

[Figure 14: Inserting the right user Id which has no external file of a member. 26](#_Toc101777270)

[Figure 15: To check whether a program will continue when the user Enters yes. 27](#_Toc101777271)

[Figure 16: Entering No to continue the program. 29](#_Toc101777272)

[Figure 17: Process in memory. 32](#_Toc101777273)

[Figure 18: Phases of Process Control Block (PCB) (Onsman, 2018). 34](#_Toc101777274)

[Figure 19:: Process States. 36](#_Toc101777275)

[Figure 20: The lowest approves-structured operating system handles interrupts and scheduling. 38](#_Toc101777276)

Table of Tables

[Table 1: Table of test 1 10](#_Toc101776465)

[Table 2: Table of test 2 11](#_Toc101776466)

[Table 3: Table of test 3 11](#_Toc101776467)

[Table 4: Table of test 4 12](#_Toc101776468)

[Table 5: Table of Test 5 14](#_Toc101776469)

[Table 6: Table of Test 6 15](#_Toc101776470)

[Table 7: Table of test 7 17](#_Toc101776471)

[Table 8: Table of test 8 19](#_Toc101776472)

[Table 9: Table of test 9 20](#_Toc101776473)

[Table 10: Table of test 10 22](#_Toc101776474)

[Table 11: Table of test 11 23](#_Toc101776475)

[Table 12: Table of test 12 24](#_Toc101776476)

[Table 13: Table of test 13 26](#_Toc101776477)

[Table 14: Table of test 14 28](#_Toc101776478)

# Task A

# INTRODUCTION

Operating System is the backbone of the computer which controls or operates all the operations of the computer. It performs and manages tasks such as file management, memory management as well as all of its software and hardware which are connected to a computer. With the help of an operating system, we can communicate with a computer without knowing how to speak the computer’s language. A computer becomes useless without installing an operating system installed on it. Some examples of the operating system include Apple macOS, UNIX, Linux Operating System, Google Android OS, and Microsoft Windows (Thakur, 2013).

UNIX is case-sensitive and strongly oriented toward lowercase characters, which are faster and easier to type. Although Unix and Linux are two distinct operating systems, they are strikingly similar in many aspects. While UNIX was created several decades before Linux, the two operating systems currently compete in almost every market. Overall, Unix has three major advantages: it is portable from large to small systems, it has extremely powerful utilities, and it provides application programs and devices independence (McHoes & Ida, 2012).

Commanding files, often called shell files or script files can be used to automate repetitious tasks. Each line of the file is a valid command and the script file can be executed by simply typing sh and the name of the script file. Another way to execute is to define the files as an executable command and simply type the filename at the system prompt. Script files are used to automate reparative tasks and to simplify the complex procedure (McHoes & Ida, 2012). 2306

## Script

1. #!/bin/bash
2. fName=$1
3. userId=$2
4. selection=('JL' 'AY' 'FM' 'DH' 'KC')
5. redColour="\033[31m"
6. greenColour="\033[0;32m"
7. yellowColour="\033[0;33m"
8. purpleColour='\033[1;35m'
9. noColour="\033[0m"
10. #Function to validate Name and UserId
11. function validUser() {
12. if [ -z $fName ] || [ -z $userId ]
13. then
14. echo -e "${redColour}"
15. echo
16. echo "Program can't execute without username and Id Hint:fileName \_name\_ \_ID\_"
17. echo
18. echo -e "${noColour}"
19. else
20. user
21. fi
22. }
23. #Function to validate user secret key
24. function user() {
25. echo -e "Enter your Secret key to access: \c"
26. read -s key
27. case $key in
28. [1][2][3][4])
29. echo
30. welcome
31. ;;
32. \*)
33. a=2
34. until [ $a -eq 0 ]
35. do
36. echo -e "${redColour}"
37. echo "Please Enter correct key. $a attempt remaining"
38. echo -e "${noColour}"
39. echo -e "Enter the secret key again: \c"
40. read -s newKey
41. # Switch case
42. case $newKey in
43. [1][2][3][4])
44. echo
45. welcome
46. ;;
47. \*)
48. echo
49. ;;
50. esac
51. a=$((a-1))
52. done
53. ;;
54. esac
55. }
56. #Welcome function
57. function welcome() {
58. echo -e "${greenColour}"
59. figlet -c "Welcome User"
60. figlet -c "UssrId:$userId Name:$fName"
61. echo -e "${noColour}"
62. date '+DATE: %m/%d/%y%nTIME:%H/%M/%S'
63. musicBand
64. }
65. #Function to display band name
66. function musicBand() {
67. echo -e "${purpleColour}"
68. echo "Guess which one is best music Band?"
69. echo "++++++++++++++++++++++++++++++++++++++++++++"
70. echo "+    Band Code         |    Band Name      +"
71. echo "++++++++++++++++++++++++++++++++++++++++++++"
72. echo "+    BEA               |    Beatles        +"
73. echo "+    AD                |    AC/DC          +"
74. echo "+    QUE               |    Queen          +"
75. echo "+    BLO               |    Blondie        +"
76. echo "+    NIR               |    Nirvana        +"
77. echo "++++++++++++++++++++++++++++++++++++++++++++"
78. echo -e "${noColour}"
79. #InFunction of Choosing Favorite Band
80. function bands() {
81. echo -e "Choose which one is mine favourate band: \c"
82. read bCode
83. #case statement
84. case $bCode in
85. [B][E][A])
86. echo -e "${greenColour}"
87. echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Beatles \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"
88. echo "The Beatles where an English rock band, formed in Liverpool in 1960, that compried Jhon Lennon, Paul McCartney, George Garridon and Ringo Starr. They are regarded as the most influential band if all time."
89. echo -e "${noColour}"
90. #Function calling
91. selectone
92. ;;
93. [A][D])
94. echo -e "${yellowColour}"
95. echo "This is not my favourate band choose correct one."
96. echo -e "${noColour}"
97. bands
98. ;;
99. [Q][U][E])
100. echo -e "${yellowColour}"
101. echo "This is not my favourate band choose correct one."
102. echo -e "${noColour}"
103. bands
104. ;;
105. [B][L][O])
106. echo -e "${yellowColour}"
107. echo "This is not my favourate band choose correct one."
108. echo -e "${noColour}"
109. bands
110. ;;
111. [N][I][R])
112. echo -e "${yellowColour}"
113. echo "This is not my favourate band choose correct one."
114. echo -e "${noColour}"
115. bands
116. ;;
117. \*)
118. echo -e "${redColour}"
119. echo "Please enter valid key word ie,BEA "
120. echo -e "${noColour}"
121. bands
122. ;;
123. esac
124. }
125. bands
126. }
127. #Function to display the table of Band Members
128. function selectone() {
129. echo -e "${purpleColour}"
130. echo "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
131. echo "+         Fivestar Band member Code  |         Band Member        +"
132. echo "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
133. echo "+             JL                 |         Jhon Lennon            +"
134. echo "+             AY                 |         Angus Young            +"
135. echo "+             FM             |         Freddie Mercury        +"
136. echo "+             DH             |         Debbie Harry           +"
137. echo "+             KC             |         Kurt Cobain            +"
138. echo "+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
139. echo -e "${noColour}"
140. ola
141. }
142. #Function for the selection of three band member.
143. function ola() {
144. echo -e "Choose any three band members using space ie,(JL AY FM): \c"
145. read selectionFirst selectionSecond selectionThird selectionForth
146. #first selection validation.
147. if [ -z $selectionForth ]
148. then
149. while [ -z $selectionFirst ]
150. do
151. echo -e "${yellowColour}"
152. echo "First selection shouldn't be empty."
153. echo -e "${noColour}"
154. ola
155. done
157. if [ $selectionFirst == ${selection[0]} ] || [ $selectionFirst == ${selection[1]} ] || [ $selectionFirst == ${selection[2]} ] || [ $selectionFirst == ${selection[3]} ] || [ $selectionFirst == ${selection[4]} ]
158. then
159. secondSelection
160. else
161. echo -e "${redColour}"
162. echo "\*\*\*\*\*\*\*\*\*\*ERROR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"
163. echo "Invalid Input on first selection."
164. echo "For Type JL for Jhon Lennon4, AY for Angus Young, FM for Freddie Murcury, DH for Deddie Harry and KC for Kurt Cobain"
165. echo -e "${noColour}"
166. ola
167. fi
168. else
169. echo -e "${yellowColour}"
170. echo "You can't pick four member name"
171. echo -e "${noColour}"
172. ola
173. fi
174. }
175. #Function of second selection and validation.
176. function secondSelection() {
177. if [ -z $selectionSecond ]
178. then
179. echo -e "${yellowColour}"
180. echo "Second selection shouldn't be empty."
181. echo -e "${noColour}"
182. ola
183. fi
185. while [ $selectionFirst == $selectionSecond ]
186. do
187. echo -e "${yellowColour}"
188. echo "First selection and Second Selection Should not be same."
189. echo -e "${noColour}"
190. ola
191. done
192. if [ $selectionSecond == ${selection[0]} ] || [ $selectionSecond == ${selection[1]} ] || [ $selectionSecond == ${selection[2]} ] || [ $selectionSecond == ${selection[3]} ] || [ $selectionSecond == ${selection[4]} ]
193. then
194. thirdSelection
195. else
196. echo -e "${redColour}"
197. echo "\*\*\*\*\*\*\*\*\*\*ERROR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"
198. echo "Invalid input on second Selection."
199. echo "For Type JL for Jhon Lennon4, AY for Angus Young, FM for Freddie Murcury, DH for Deddie Harry and KC for Kurt Cobain"
200. echo -e "${noColour}"
201. ola
202. fi
203. }
204. #Function of third selection and validation
205. function thirdSelection() {
206. while [ -z $selectionThird ]
207. do
208. echo -e "${yellowColour}"
209. echo "Third selection shouldn't be empty."
210. echo -e "${noColour}"
211. ola
212. done
213. while [ $selectionFirst == $selectionThird ]
214. do
215. echo -e "${yellowColour}"
216. echo "First selection and Third Selection Should not be same."
217. echo -e "${noColour}"
218. ola
219. done
220. while [ $selectionSecond == $selectionThird ]
221. do
222. echo -e "${yellowColour}"
223. echo "Second selection and Third selection should not be same."
224. echo -e "${noColour}"
225. ola
226. done
227. if [ $selectionThird == ${selection[0]} ] || [ $selectionThird == ${selection[1]} ] || [ $selectionThird == ${selection[2]} ] || [ $selectionThird == ${selection[3]} ] || [ $selectionThird == ${selection[4]} ]
228. then
229. chooseOne
230. else
231. echo -e "${redColour}"
232. echo "\*\*\*\*\*\*\*\*\*\*ERROR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"
233. echo "Invalid Input on Third selection."
234. echo "For Type JL for Jhon Lennon, AY for Angus Young, FM for Freddie Murcury, DH for Deddie Harry and KC for Kurt Cobain"
235. echo -e "${noColour}"
236. ola
237. fi
238. }
239. #Function of choosing band member
240. function chooseOne() {
241. PS3="Select any singer Name from Option: "
242. select var in $selectionFirst $selectionSecond $selectionThird
243. do
244. while [ -z $var ]
245. do
246. echo -e "${redColour}"
247. echo "\*\*\*\*\*\*\*Please Enter valid Input\*\*\*\*\*\*\*\*\*\*\*\*\*\*"
248. echo "Choose value 1 for $selectionFirst 2 for $selectionSecond and 3 for $selectionThird."
249. echo -e "${noColour}"
250. chooseOne
251. done
252. if [ $var == $selectionFirst ] || [ $var == $selectionSecond ] || [ $var == $selectionThird ]
253. then
254. while [ $var == "JL" ]
255. do
256. echo -e "${greenColour}"
257. cat JL
258. echo -e "${noColour}"
259. endFunction
260. done
261. if [ $var == "AY" ]
262. then
263. echo -e "${greenColour}"
264. cat AY
265. echo -e "${noColour}"
266. endFunction
267. elif [ $var == "FM" ]
268. then
269. echo -e "${greenColour}"
270. cat FM
271. echo -e "${noColour}"
272. endFunction
273. elif [ $var == "DH" ]
274. then
275. echo -e "${yellowColour}"
276. echo "DH (Debbie Harry) band member File is not available"
277. echo -e "${noColour}"
278. musicBand
279. elif [ $var == "KC" ]
280. then
281. echo -e "${yellowColour}"
282. echo "KC (Kurt Cobain) band member file is not available"
283. echo -e "${noColour}"
284. musicBand
285. else
286. echo -e "${redColour}"
287. echo "Choose value 1 for $selectionFirst, 2 for $secondSelection and 3 for $thirdSelection."
288. echo -e "${noColour}"
289. chooseOne
290. fi
291. fi
292. done
293. }
294. #Program exit Function
295. function endFunction() {
296. echo -e "Do you want to continue(Yes/No): \c "
297. read opt
299. while [ -z $opt ]
300. do
301. echo -e "${redColour}"
302. echo "-------------------Invalid-Input---------------------"
303. echo "Type "Yes" for Continue and "No" for exit the program"
304. echo -e "${noColour}"
305. endFunction
306. done
307. if [ $opt == "Yes" ]
308. then
309. musicBand
310. elif [ $opt == "No" ]
311. then
312. echo -e "${greenColour}"
313. figlet "Program Exit"
314. echo -e "${noColour}"
315. exit
316. else
317. echo -e "${redColour}"
318. echo "-------------------Invalid-Input---------------------"
319. echo "Type "Yes" for Continue and "No" for exit the program"
320. echo -e "${noColour}"
321. endFunction
322. fi
323. }
324. validUser

## 1.3) Testing

Testing is a way of knowing whether the system which we have developed either correct or not. Testing is an internal part of programming. It helps to identify the error so the developer can correct it or replace one with a better one.

Test No 1: Running without a username

Objective: To check whether a program can execute without a username.

Table 1: Table of test 1

|  |  |
| --- | --- |
| **Test No.** | **1** |
| Input | * Open ubuntu and open file using bash filename without username and ID (bash 20049086cw2ii) |
| Expected Output | Should Display massage “Program can't execute without username and Id” |
| Actual Output | Massage display as “Program can't execute without username and Id” |
| Test Result | Test results passed successfully. |

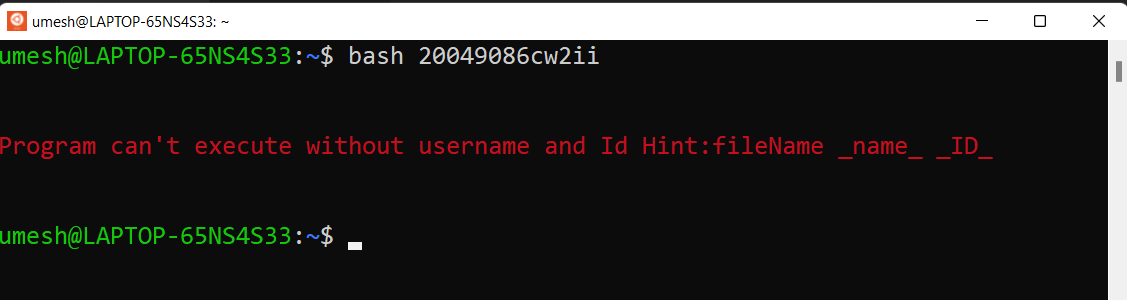


Figure 1: Displaying error massage "Program can't execute without username and Id"

Test 2: run with username and id

Objectives: to check whether the program will run with username and Id.

Table 2: Table of test 2

|  |  |
| --- | --- |
| **Test No.** | **2** |
| Input | * Open ubuntu and open file using bash filename with username and ID |
| Expected Output | The program should ask for Secrete key from the user for program access. |
| Actual Output | The program asks to secrete a key from the user for program access. |
| Test Result | Test results passed successfully. |

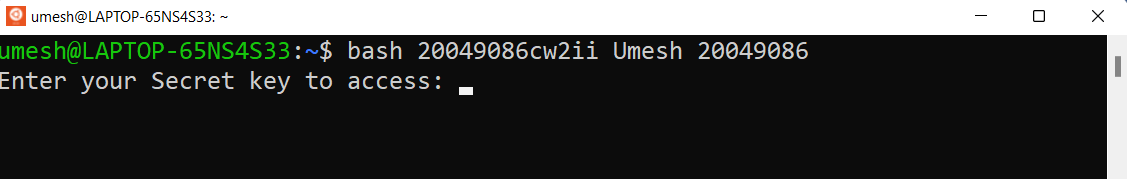


Figure 2: run with username and id

Test 3: Running an incorrect password three times.

Objective: To check where the program gives the user more than three chances.

Table 3: Table of test 3

|  |  |
| --- | --- |
| **Test No.** | **3** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the random secret key. * Attempt of program decrease. * When several attempts are equal to 0 then the program terminates. |
| Expected Output | The program should terminate. |
| Actual Output | The program terminates. |
| Test Result | Test results passed successfully. |

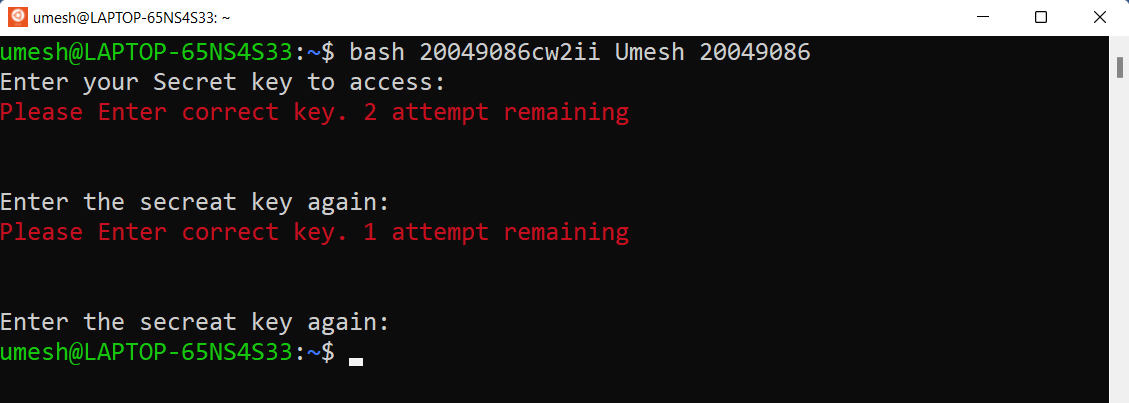


Figure 3: Running an incorrect password three times

Test 4: Running correct password.

Objective: To check whether the program gives the user more than three chances.

Table 4: Table of test 4

|  |  |
| --- | --- |
| **Test No.** | **4** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. |
| Expected Output | The program should welcome a user by mentioning the user Id with the username and displaying the access date. And a program should be asked users to choose the programmer’s favorite band. |
| Actual Output | The program welcomes a user by mentioning the user Id with the username and displaying the access date. And a program asked users to choose the programmer’s favorite band. |
| Test Result | Test results passed successfully. |

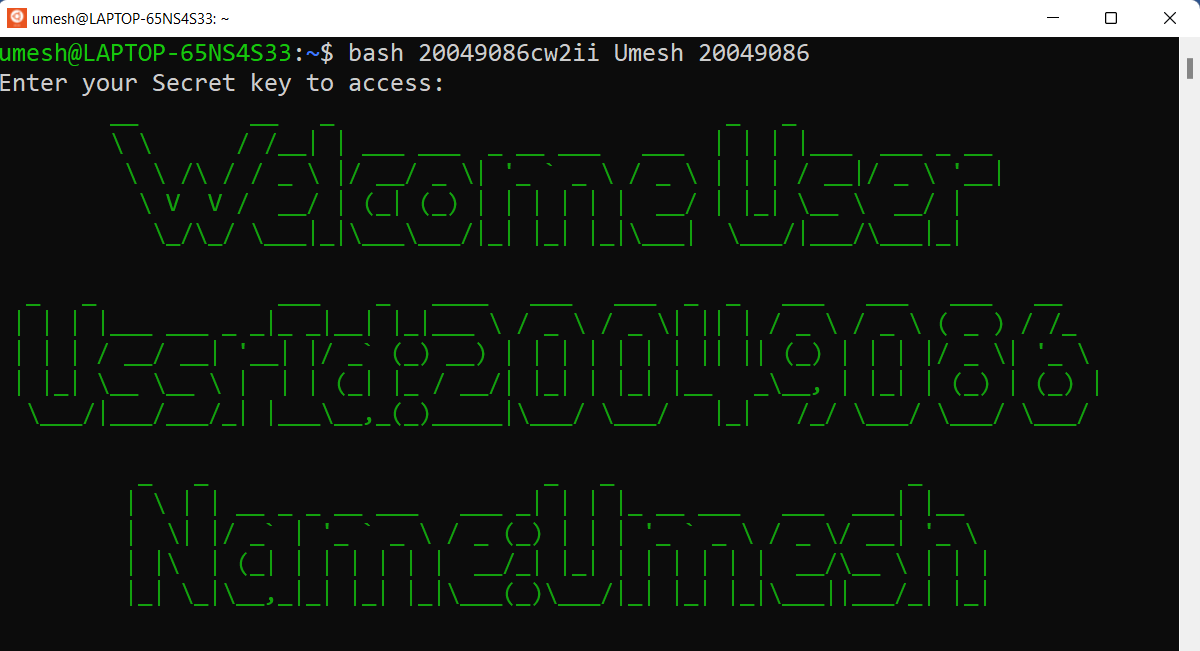


Figure 4: Running correct password part 1

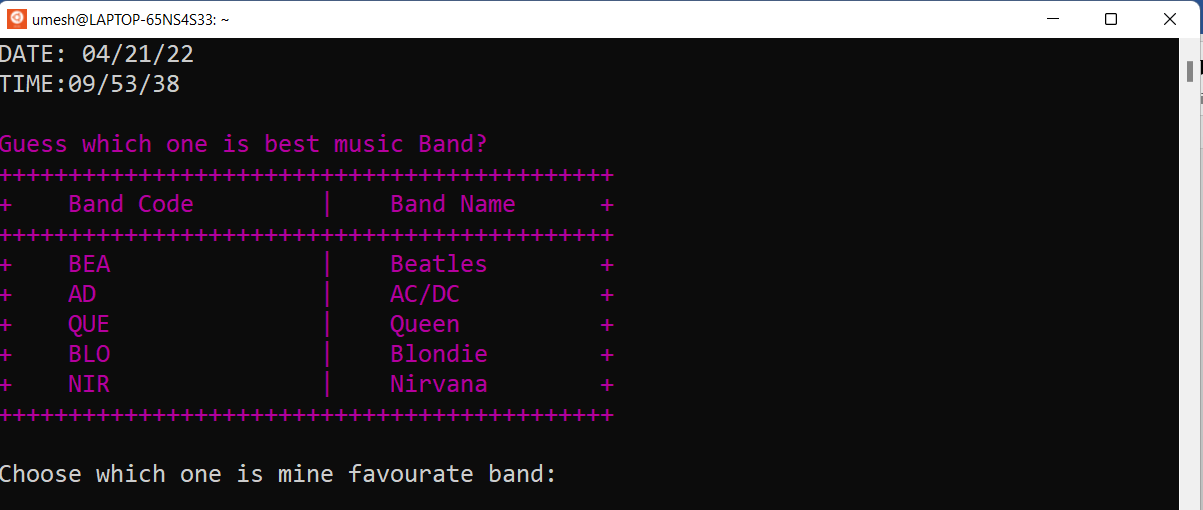


Figure 5: Running correct password part 2

Test 5: Displaying band name.

Objective: To check whether a program displays a band name with code or not.

Table 5: Table of Test 5

|  |  |
| --- | --- |
| **Test No.** | **5** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. |
| Expected Output | The program should display a band name with code. And a program should be asked users to choose the programmer’s favorite band. |
| Actual Output | The program displays a band name with a code. And a program asked users to choose the programmer’s favorite band |
| Test Result | Test results passed successfully. |



Figure 6: Displaying band name.

Test 6: Running incorrect band Code.

Objective: To check whether a program will execute with incorrect band code.

Table 6: Table of Test 6

|  |  |
| --- | --- |
| **Test No.** | **6** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering incorrect band Code. |
| Expected Output | When a user enters a random word or empty text then a program should display an error message “Please enter valid keyword” and if a user enters a valid but unfavourite band code then the program should display a message “This is not my favorite band choose correct one”. |
| Actual Output | When a user enters a random word or empty text then a program displays an error message “Please enter valid keyword” and if a user enters a valid but unfavourite band code then the program displays a message “This is not my favorite band choose correct one”. |
| Test Result | Test results passed successfully. |

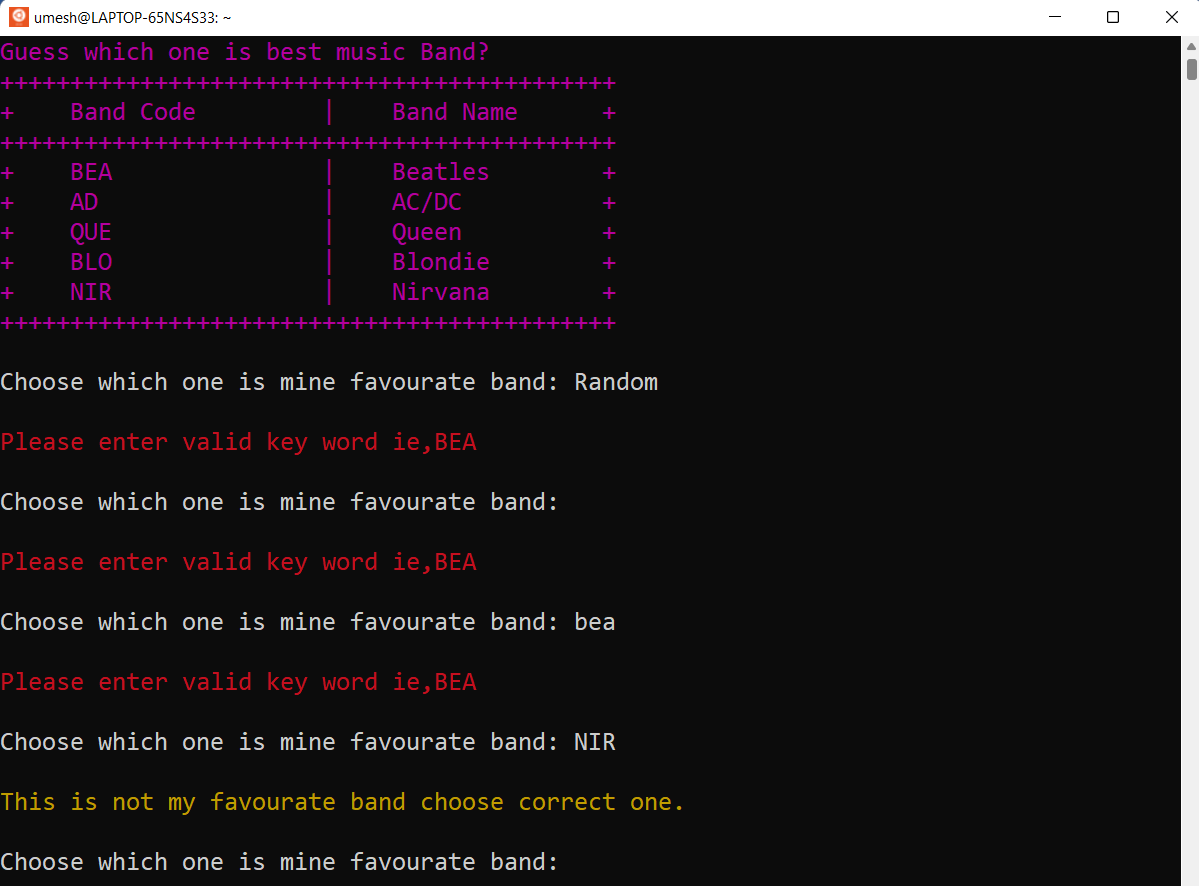


Figure 7: Running incorrect band Code part 1.

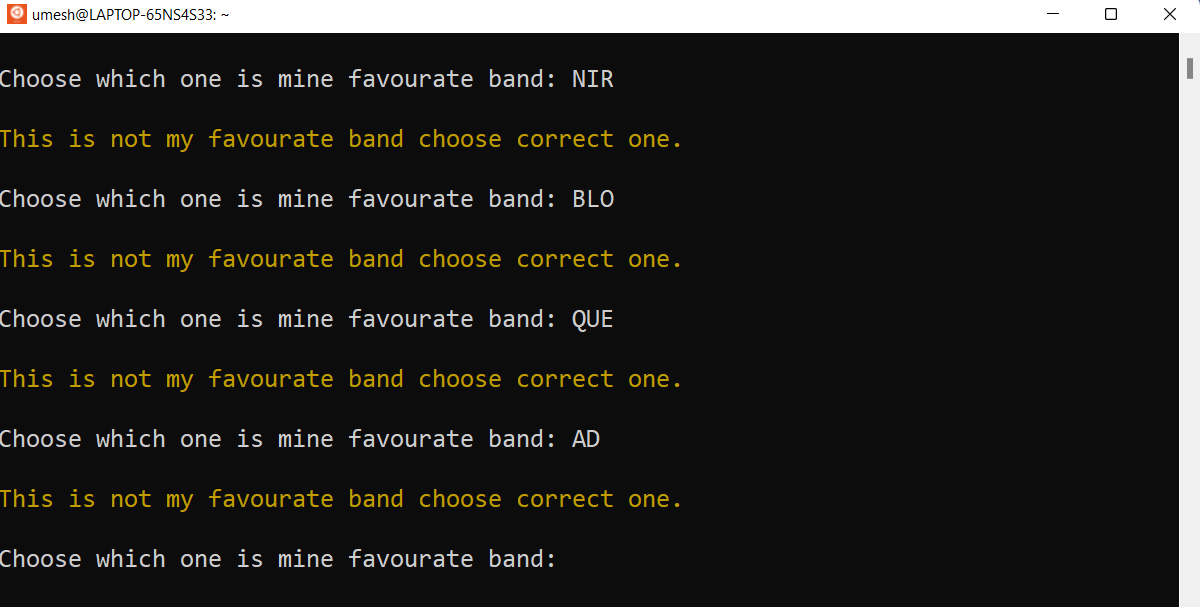


Figure 8: Running incorrect band Code part 2.

Test 7: Running correct band code

Objective: To check whether the program will execute with the correct band code.

Table : Table of test 7

|  |  |
| --- | --- |
| **Test No.** | **7** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering correct band Code. |
| Expected Output | When a user chooses the correct band member then a program should display a description of the band and the program should ask a user to choose three band members. |
| Actual Output | When a user chooses the correct band member then a program displays a description of the band should display and the program asks a user to choose three band members. |
| Test Result | Test results passed successfully. |

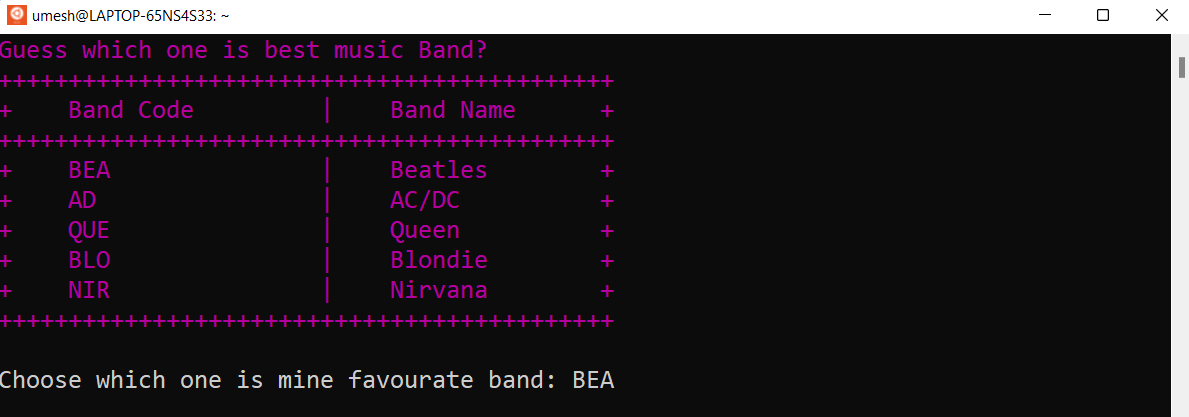


Figure 9: Running correct band code part 1.

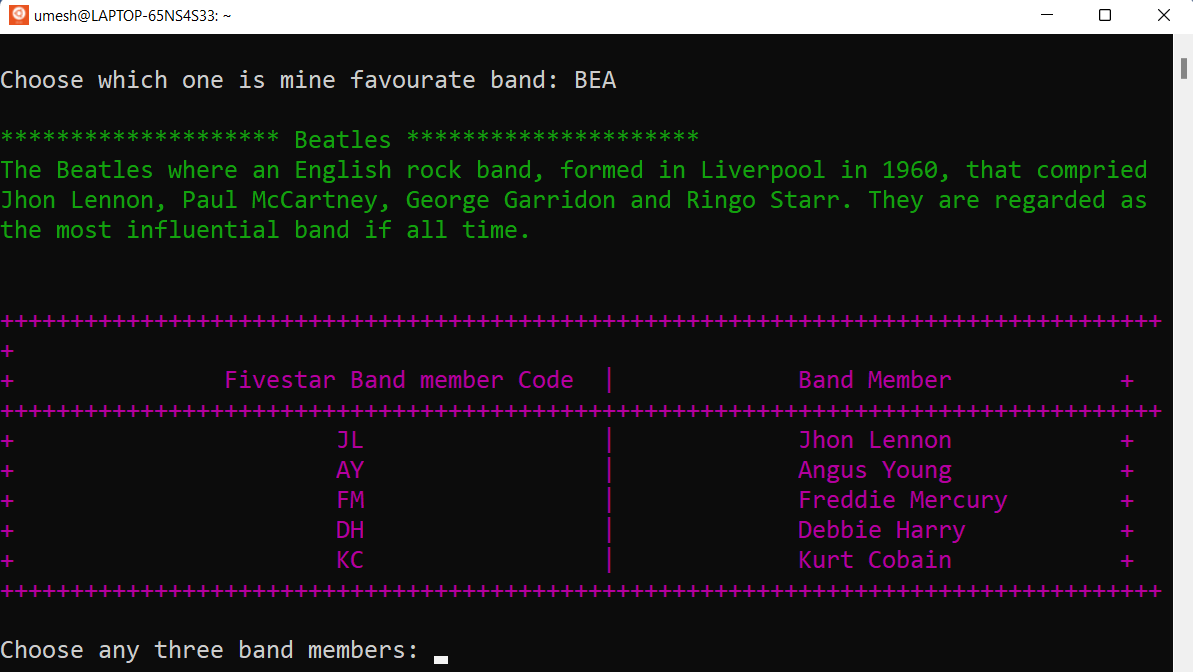


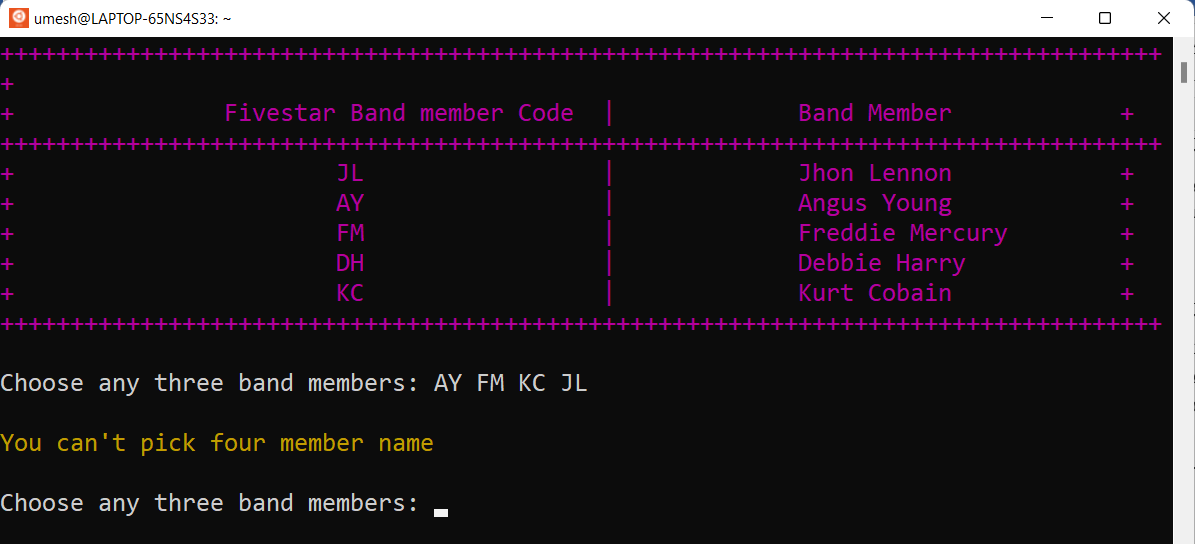
Figure 10: Running correct band code part 2.

Test 8: Picking a 4-member name.

Objective: To check whether a program allows a user to pick 4 members.

Table 8: Table of test 8

|  |  |
| --- | --- |
| **Test No.** | **8** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Entering the correct secret key. * Entering correct band Code. * Picking 4 valid band members from the list. |
| Expected Output | When a user picks a four-member from the list then a program should display an error message “You can’t pick four members.”. And again, a program should ask a user to choose the band’s member’s code. |
| Actual Output | When a user picks a four-member from the list then a program displays an error message “You can’t pick four members.”. And again, a program asks a user to choose the band’s member’s code. |
| Test Result | Test results passed successfully. |



Test 9: Picking the same band member’s name

Objective: To check whether a program allows a user to choose the same band member twice.

Table 9: Table of test 9

|  |  |
| --- | --- |
| **Test No.** | **9** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering correct band Code. * Picking the same band member code twice. |
| Expected Output | When a user picks the same band member code twice then the program should display a message in which selection a user has selected the band code twice. And again, a program should ask a user to choose bands members. |
| Actual Output | When a user picks the same band member code twice then the program displays a massage in which selection a user has selected the band code twice. And again, a program asks a user to choose bands members. |
| Test Result | Test results passed successfully. |

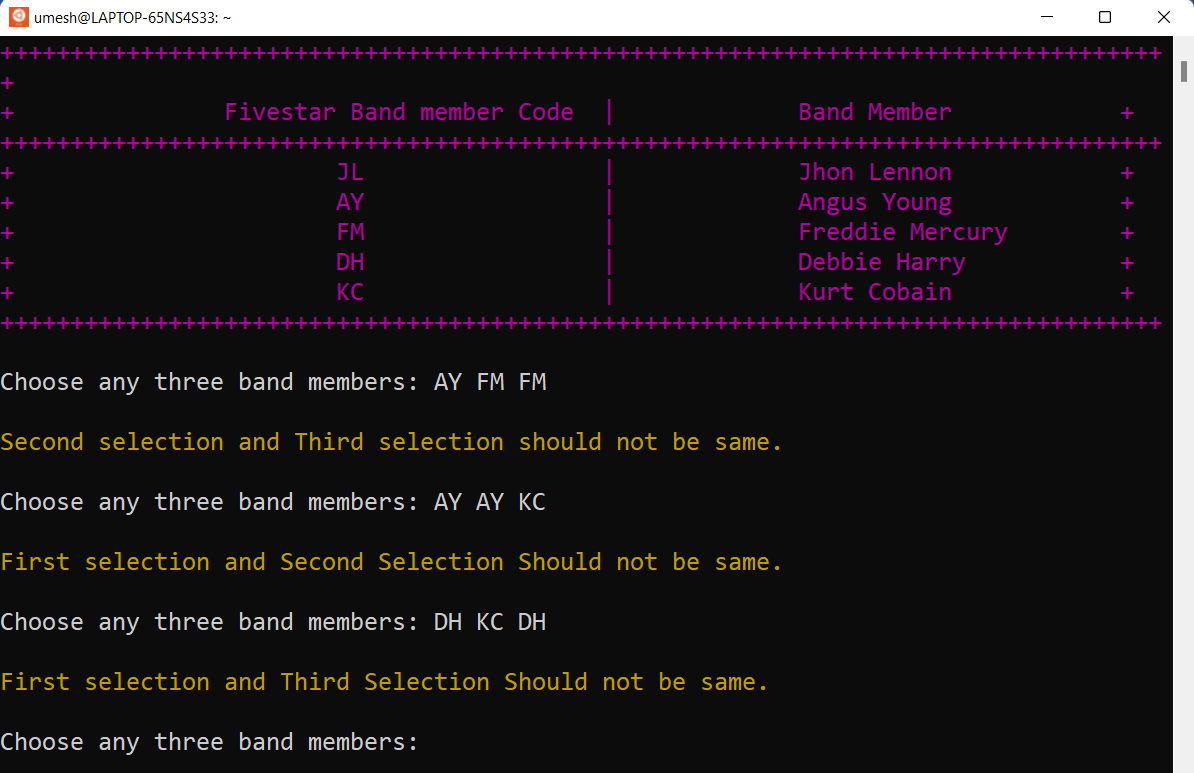


Figure 11: Picking a 4-member name.

Test 10: Running wrong user id. (Parameter validation)

Objective: To check whether a program will run with the wrong user Id.

Table : Table of test 10

|  |  |
| --- | --- |
| **Test No.** | **10** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering correct band Code. * Picking different band member codes. * Inserting invalid user Id. |
| Expected Output | When a user inserts the wrong user id then a program should display an error message “Please insert valid input.”. And again, a program should ask a user to insert a user Id. |
| Actual Output | When a user inserts the wrong user id then a program displays an error message “Please Enter valid input.”. And again, a program asks a user to insert a user Id. |
| Test Result | Test results passed successfully. |

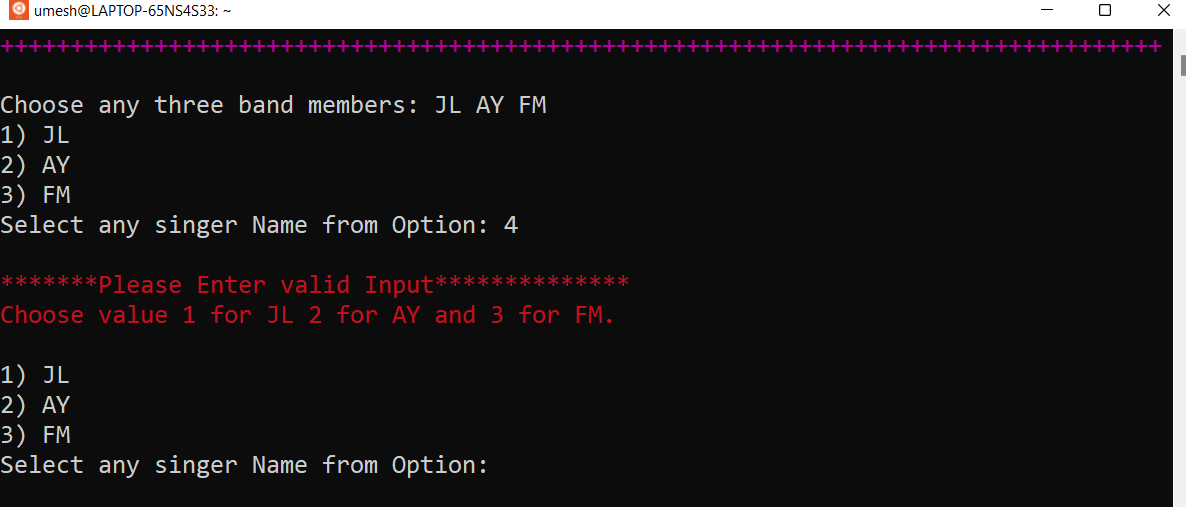


Figure 12: Running wrong user id. (Parameter validation)

Test 11: Inserting the right user Id which has an external file of a member.

Objective: To check whether a program run with a valid user Id that has an external file of a member.

Table 11: Table of test 11

|  |  |
| --- | --- |
| **Test No.** | **11** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering correct band Code. * Picking different band member codes. * Inserting valid user Id which has an external file of a member. |
| Expected Output | When a user inserts the valid user id which has an external file of a member then a program should display some information about band members and a program should ask a user to exit. |
| Actual Output | When a user inserts the valid user id which has an external file of a member then a program displays some information about band members and a program asks a user to exit. |
| Test Result | Test results passed successfully. |

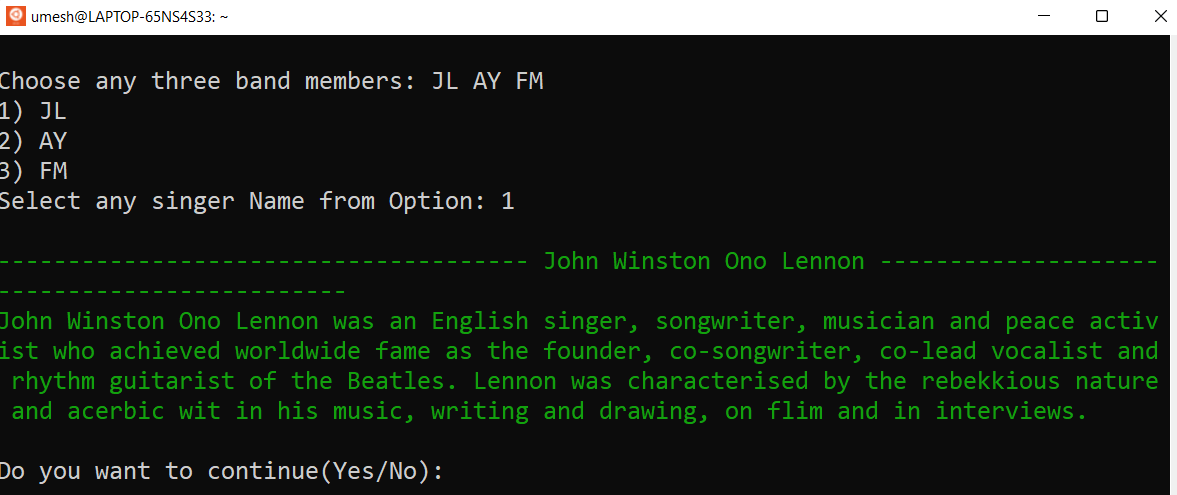


Figure 13: Inserting the right user Id which has an external file of a member.

Test 12: Inserting the right user Id which has no external file of a member.

Objective: To check whether a program run with a valid user Id that has no external file of a member.

Table 12: Table of test 12

|  |  |
| --- | --- |
| **Test No.** | **12** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering correct band Code. * Picking different band member codes. * Inserting valid user Id which has no external file of a member. |
| Expected Output | When a user inserts the valid user id which has no external file of a member then a program should display the message “Band member File not available”. And again, the program should ask a user to choose the best music band. |
| Actual Output | When a user inserts the valid user id which has no external file of a member then a program displays the message “Band member File not available”. And again, the program asks a user to choose the best music band. |
| Test Result | Test results passed successfully. |

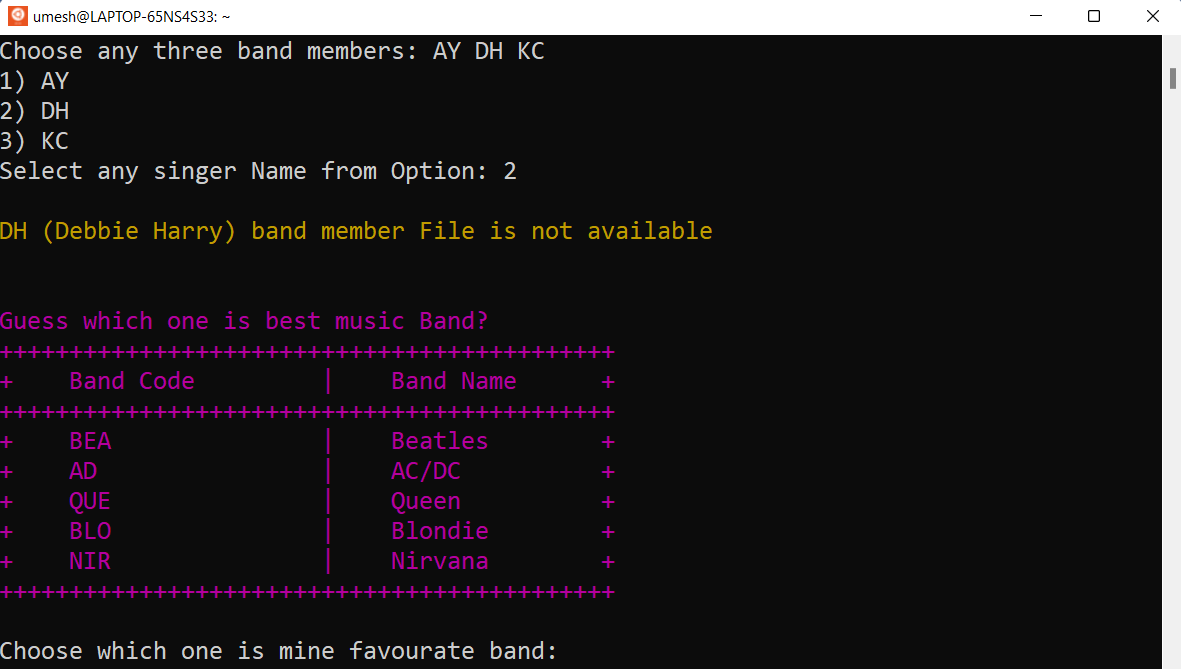


Figure 14: Inserting the right user Id which has no external file of a member.

Test 13: Entering Yes to continue the program.

Objective: To check whether a program will continue when the user Enters yes.

Table 13: Table of test 13

|  |  |
| --- | --- |
| **Test No.** | **13** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering correct band Code. * Picking different band member codes. * Inserting valid user Id which has an external file of a member. * Entering Yes on do you want to continue. |
| Expected Output | When a user inserts Yes. And again, the program should ask a user to choose the best music band. |
| Actual Output | When a user inserts Yes. And again, the program asks a user to choose the best music band. |
| Test Result | Test results passed successfully. |

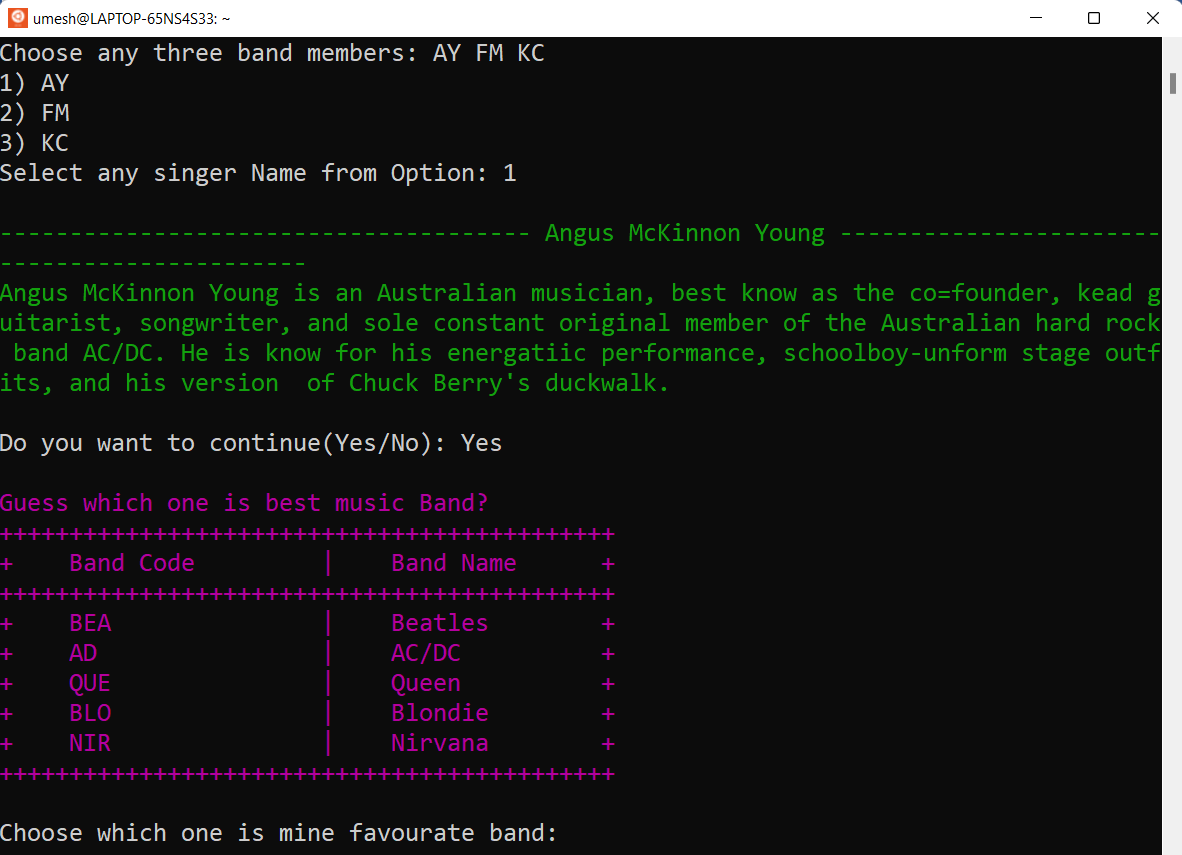


Figure 15: To check whether a program will continue when the user Enters yes.

Test 14: Entering No to continue the program.

Objective: To check whether a program will continue when the user Enters No.

Table 14: Table of test 14

|  |  |
| --- | --- |
| **Test No.** | **14** |
| Input | * Open ubuntu and open file using bash filename with username and ID * Enter the correct secret key. * Entering correct band Code. * Picking different band member codes. * Inserting valid user Id which has an external file of a member. * Entering No on do you want to continue. |
| Expected Output | When a user inserts No. The program should exit. |
| Actual Output | When a user inserts No. The program excited the user. |
| Test Result | Test results passed successfully. |



Figure 16: Entering No to continue the program.

## Contents of three files.

### JL

John Winston Ono Lennon was an English singer, songwriter, musician, and peace activist who achieved worldwide fame as the founder, co-songwriter, co-lead vocalist, and rhythm guitarist of the Beatles. Lennon was characterized by the rebellious nature and acerbic wit in his music, writing, and drawing, on film and in interviews.

### AY

Angus McKinnon Young is an Australian musician, best known as the co=founder, lead guitarist, songwriter, and sole constant original member of the Australian hard rock band AC/DC. He is known for his energetic performance, schoolboy-unform stage outfits, and his version of Chuck Berry's duckwalk.

### FM

Freddie Mercury was a British singer-songwriter, who was best known as the vocalist of the rock band Queen. Regarded as one of the greatest singers in the history of rock music, he was known for his flamboyant stage persona and four-octave vocal range.

## Conclusion

This assignment was finally completed after a lot of hard work and several research projects on related topics like Unix, bash scripting, and many more. The coursework assignments were not simple at all.

The course work also required a properly documented report of all tasks completed during the program's development. The development and testing of the program are completed and documentation of the program is completed. Development of coursework required the scripting of Unix commands using Ubuntu to be completed. During the command scripting, many valuable lessons were learned. Overall, this course work provided a glimpse into what the life of a future programmer might be like. It is a lovely experience while bash scripting.

# Task B

## Introduction

2306We are living in a technological era. In this era, process management is involved in every aspect of human beings. Process management is one of the most important and relevant tasks in operating system design. Process management involved the execution of various tasks such as the creation of processes, scheduling of processes, termination of processes, management of deadlocks so on. The operating system handles changes in the system's running processes as well as tasks such as scheduling and resource allocation. Let’s say a person named ashrin use system Windows XP or Mac OS and he is working on MS Word. In the background, has decided to play some music. He has some online articles opened. He is surfing the net and so on. So the operating system needs to manage the network performance disk and memory management. This is why process management is needed. All of the applications in the background are processes that require resources and processor time to complete (Codingninjas, 2021).

Aims and Objectives

The assignment aims to know about how process management of an operating system works. To know how states of process states interact with each other, to understand how PCB works, and understand how process states work.

1. To know the concept of process management.
2. To know about changes in technology.
3. To know how process management works.
4. To understand how process states work.
5. To understand how PCB works on the operating system.
6. To know how processes are implemented on the operating system.

## Background

Earlier there used to be only single-task performing systems. There used to be just one process running at a time in this way. It also had complete freedom to use all of the system's resources. Today, however, we have multitasking and multiprocessing operating systems, which allow multiple processes to operate simultaneously.

### 2.2.1 Process Architecture

Process architecture refers to the hierarchal design of processes and systems that are applied when transforming inputs into outputs. There are five states of process Architecture. They are shown in the figure.

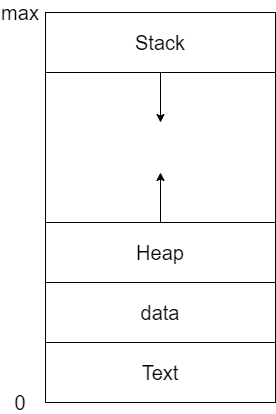


Figure 17: Process in memory.

1. Stack

The stack contains temporary data, such as function parameters, returns addresses, and local variables.

1. Heap

Dynamically allocated memory to process during its run time.

1. Data

Contains the global variable.

1. Test

A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the **Program Counter** (Silberschatz, et al., 2009).

Overall, these are the five states of Process Architecture.

### 2.2.2 Process Control Blocks

The operating system executes various activities in creating a process. It assigns a process identification number (PID) to each process to identify it. Because multi-programming is supported by the operating system, it must keep track of all processes. The process control block (PCB) is used to track the process's execution status for this task. Each memory block holds information on the current state of the process, the program number, the stack pointer, the status of open files, scheduling algorithms, and so on. It contains many pieces of information associated with a specific process (Silberschatz, et al., 2009).

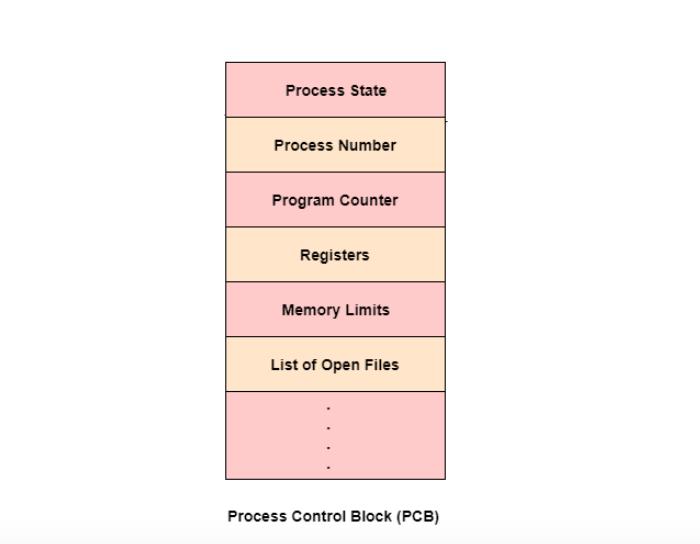


Figure 18: Phases of Process Control Block.

(Source: https://www.tutorialspoint.com/what-is-process-control-block-pcb)

These are the data items of Process Control Block

1. Process State

The state may be new, ready, running, waiting, halted, and so on (Onsman, 2018).

1. Program Counter

The counter indicates the address of the new instruction to be executed for this process (Onsman, 2018).

1. CPU registers

The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward (Onsman, 2018).

1. CPU-scheduling Information

This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters (Onsman, 2018).

1. Memory-management information

This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system (Onsman, 2018).

1. Accounting information

This information includes the amount of CPU and real-time used, time limits, account numbers, job or process numbers, and so on (Onsman, 2018).

1. I/O state information

This information includes the list of I/O devices allocated to the process, a list of open files, and so on (Onsman, 2018).

In brief, the PCB serves as the repository for any information that may vary from process to process.

### 2.2.3 the Process States

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. The process may be in one of the following states.



Figure 19:: Process States.

(Image Source:

https://www.researchgate.net/publication/322254868\_Understanding\_Enforcement\_of\_Flow\_of\_Processes\_in\_Operating\_Systems)

1. New/Entry

A new process is a program that will be loaded into the main memory by the operating system (Shaikh, 2017).

1. Ready

When a process is created, it immediately goes into the ready state, where it waits for the CPU to be assigned to it. The operating system selects new processes from secondary memory and places them all in the main memory. Ready state processes are processes that are ready for execution and sit in the main memory. Many processes may be active in the ready state (Shaikh, 2017).

1. Running

The OS will choose one of the processes from the ready state based on the scheduling mechanism. As a result, if our system only has one CPU, the number of running processes at any given time will always be one. If the system has n processors, we can have n processes executing at the same time (Shaikh, 2017).

1. Ready Suspended

The ready suspended state refers to a process in the ready state that is transferred to secondary memory from the main memory due to a lack of resources (mostly primary memory). If the main memory is full and a higher priority process is scheduled for execution, the OS must free up space in the main memory by moving the lower priority process to the secondary memory. Suspend ready processes are kept in secondary memory until the main memory is free (Shaikh, 2017).

1. Blocked Suspended

Depending on the scheduling method or the intrinsic behavior of the process, a process can migrate from the Running state to the Block or Wait for states. When a process waits for a resource to be allotted or for user input, the OS puts it in a block or waits for state and gives the CPU to other processes (Shaikh, 2017).

1. Termination

A process enters the termination state when it has completed its execution. The process's context (Process Control Block) will be removed as well, and the process will be terminated by the operating system (Shaikh, 2017).

1. Blocked

Depending on the scheduling method or the intrinsic behavior of the process, a process can migrate from the Running state to the Blocked state. When a process waits for a resource to be allotted or for user input, the OS puts it in a blocked state and gives the CPU to other processes (Shaikh, 2017).

Hence, these are the major stages of the process state.

### 2.2.4 Process Hierarchies

When a process generates a child process, the parent and child processes tend to associate with each other in specific ways. If necessary, the child process can also create other processes. This parent-child-like structure of processes forms a hierarchy, called Process Hierarchy. Similarly, when one process generates another, the parent and child are still linked in certain ways. The youngster can develop new processes on his or her own, forming a process hierarchy. Unlike plants and animals that reproduce sexually, a process only has one parent (Tanenbaum & Woodhull, 2012).

A process, its children, and further descendants together may form a process group. This is signal-dependent. When a signal is given to a group, each process has the option of catching the signal, ignoring it, or taking the default action of being killed by the signal (Tanenbaum & Woodhull, 2012).

### 2.2.5. Implementation of Processes

The operating system uses a table (an array of structures) called the process table to implement the process model, with one entry per process. (These items are also known as process control blocks by some writers.) This entry contains information about the process's state, program counter, stack pointer, memory allocation, the status of open files, accounting and scheduling information, alarms and other signals, and everything else about the process that must be saved when it is switched from running to ready state so that it can be restarted later as if it had never been stopped

(Tanenbaum & Woodhull, 2012).

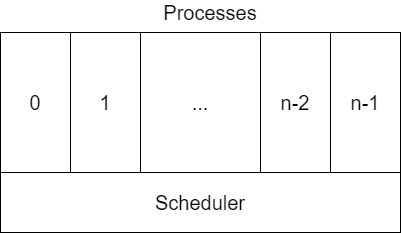


Figure 20: The lowest approves-structured operating system handles interrupts and scheduling.

### 2.2.6 Threads

Within a process, a thread is a single sequence stream. Threads are frequently referred to as lightweight processes since they share some of the properties of processes. A thread is a basic unit of CPU utilization, consisting of a program counter, a stack, and a set of registers (Sharma, et al., 2014).

Threads are a common approach to improving program parallelism in many ways. The CPU swiftly shifts between the threads, providing the impression that they are executing in parallel. A thread can be in any of numerous states, much like in a traditional process with one thread. There is a separate stack for each thread. Because threads typically call separate procedures, they will have a different execution history. It is for this reason that each thread needs its stack. A thread is a basic unit of CPU utilization in an operating system with thread support (Sharma, et al., 2014).

Threads are crucial in the development of an operating system. A server, such as a printer server, is made up of numerous threaded processes. Threads do not require interprocess communication because they can share common data. Threads can take advantage of multiprocessors by their very nature. Threads consume extremely few resources from the operating system in which they run. Threads, in other words, don't require any more address space, global data, program code, or operating system resources (Sharma, et al., 2014).

## Conclusion

This assignment was finally completed after a lot of hard work and several research on relative topics like operating systems, process management, PCB of operating systems, and so on. The assignments were not simple at all. The assignment carries out lots of research and dedication.

In conclusion, the operating system's process underlying structure holds a lot of information, including not only the code we wrote, but also the status information on how the CPU, RAM, and other hardware are working right now. It's the same as stating the work we want to do and then handing it over to the operating system, which will coordinate resources to complete the task. The entire procedure is abstracted into a procedure. When a process is created, it cannot be executed right away because the operating system has its own set of scheduling principles, known as process scheduling.

# References

Codingninjas, 2021. Importance Of Processor Management In Operating System. *Importance Of Processor Management In Operating System*, 2 August, pp. 1-3.

guru99, 2022. *Process Management in OS: PCB in Operating System.* [Online]   
Available at: https://www.guru99.com/process-management-pcb.html  
[Accessed 18 4 2022].

McHoes, A. M. & Ida, M. F., 2012. *Understanding Operating Systems.* 6th ed. Canada: Nelson Education, Ltd.

Onsman, A., 2018. *What is Process Control Block (PCB)?.* [Online]   
Available at: https://www.tutorialspoint.com/what-is-process-control-block-pcb  
[Accessed 18 April 2022].

Shaikh, Z. A., 2017. Understanding Enforcement of Flow of Processes in Operating Systems. *Understanding Enforcement of Flow of Processes in Operating Systems,* 5(5), pp. 79-81.

Sharma, K., Saini, H. & Mehta, G., 2014. Threads in Operating System. *Threads in Operating System,* 2(IX), pp. 220-222.

Silberschatz, A., Galvin, P. B. & Gagne, G., 2009. *Operating System Concepts.* 8th ed. jefferson city: R.R Donnelley.

Tanenbaum, A. S. & Woodhull, A. S., 2012. *Operating Systems Design and Implemntation.* 3rd ed. New Jersey: Pearson Education Inc,.

Thakur, D., 2013. *What is Operating System and its types? Definition.* [Online]   
Available at: https://ecomputernotes.com/fundamental/disk-operating-system/what-is-operating-system  
[Accessed 14 March 2022].

Wang, Y., 2004. Operating Systems. *Operating Systems,* I(2nd), pp. 5-6.

# Appendix

# Appendix -A(Glossary)

Interprocess: mechanism provided by the operating system that allows processes to communicate with each other

Parallelism: the process of decomposing a problem into smaller tasks that can be executed at the same time using multiple compute resources.

Intrinsic: belonging naturally

Multitasking: the performance of more than one task at the same time.

Multiprocessing: the running of two or more programs or sequences of instructions simultaneously by a computer with more than one central processor.

## Appendix – B (Process Scheduling)

Process scheduling is a basic operating system feature that allows you to get the most out of your CPU. Multiprogram and multithread approaches are used to keep the CPU running many processes or threads at the same time. Multiprogramming and multithreaded operating systems are built on the foundation of process scheduling (Wang, 2004).

## Appendix – C (Priority Scheduling)

This algorithm assigns different priorities to individual processes. On this basis, CPU scheduling will be carried out by selecting the highest-priority process. The priority algorithm's flaw is hunger, which refers to the indefinite stalling of low priority processes due to heavy CPU load. To avoid famine, an aging mechanism could be used, which raises the priority levels of low-priority processes regularly, causing their execution priorities to rise automatically while they wait in the ready queue (Wang, 2004).