

**Name: Sambeg Raj Subedi**

**Date: 03/18/2021**

**Lab 3 Report:**

**Part 1 Screenshot:**

The screenshot displays the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'OPERATING SYSTEM LAB3' with files including 'Part\_3', 'a.out', 'destination1.txt', 'destination2.txt', 'Parent\_Process.c', 'Process\_P1', 'Process\_P1.c', 'Process\_P2', 'Process\_P2.c', 'source.txt', '~\$mbeg LAB3 Report .docx', 'a.out', 'Part\_1.c', 'Part\_2.c', and 'Sambeg LAB3 Report .docx'. The main editor window is open to 'Part\_1.c', showing the following C code:

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <sys/wait.h>
5  #include <sys/types.h>
6
7  int main (int argc, char *argv[])
8  {
9      int status;
10     int child = fork();
11     if (child == 0)
12     {
13         printf ("PID value is , %d\n", getpid());
14
15         execl ("/bin/date", "date", 0, NULL);
16         printf ("Failed\n");
17     }
18     else if (child > 0)
19     {
20         wait (&status);
21     }
22     else
23     {
24         perror ("Error detected");
25     }
26     return 0;
27 }
```

Below the code editor is the TERMINAL panel, which shows the execution of the program:

```
sambeg@MacBook-Pro Operating System Lab3 % gcc Part_1.c
sambeg@MacBook-Pro Operating System Lab3 % ./a.out
PID value is , 86120
Thu Mar 18 14:17:36 EDT 2021
sambeg@MacBook-Pro Operating System Lab3 %
```

The status bar at the bottom indicates the current position is 'Ln 23, Col 6', the tab size is '4', the encoding is 'UTF-8', the line ending is 'LF', and the platform is 'Mac'.

## Part 2 Screenshot:

The screenshot shows the Visual Studio Code editor with the file `Part_2.c` open. The code is a C program that forks a child process. The parent process prints the PID value and then waits for the child to finish. The child process prints its own PID value and then exits. The terminal shows the output of the program, including the PID values and the message "EXECVP process failed".

```
C Part_2.c > main(int, char * [])
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <sys/wait.h>
5  #include <sys/types.h>
6  int main (int argc, char *argv[])
7  {
8      int status;
9      int child = fork();
10     if (child == 0)
11     {
12         printf ("PID value is , %d\n", getpid());
13         char *argv[] = {"ls", "-la", 0};
14         execvp(argv[0], &argv[0]);
15         printf ("EXECVP process failed\n");
16     }
17     else if (child > 0)
18     {
19         wait(&status);
20     }
21     else
22     {
23         perror("Error detected");
24     }
25 }
```

Terminal Output:

```
Sambeg@MacBook-Pro Operating System Lab3 % gcc Part_2.c
Sambeg@MacBook-Pro Operating System Lab3 % ./a.out
PID value is , 86143
total 168
drwxr-xr-x  9 sambeg staff   288 Mar 18 14:18 .
drwxr-xr-x 42 sambeg staff  1344 Mar 18 14:17 ..
-rw-r--r--  1 sambeg staff   6148 Mar 18 11:12 .DS_Store
-rw-r--r--  1 sambeg staff   488 Mar 18 11:16 Part_1.c
-rw-r--r--  1 sambeg staff   433 Mar 18 11:19 Part_2.c
drwxr-xr-x 12 sambeg staff   384 Mar 18 14:13 Part_3
-rw-r--r--  1 sambeg staff  11689 Mar 18 14:15 Sambeg LAB3 Report .docx
-rwxr-xr-x  1 sambeg staff  49768 Mar 18 14:18 a.out
-rw-r--r--  1 sambeg staff   162 Mar 18 14:15 ~$mbeg LAB3 Report .docx
Sambeg@MacBook-Pro Operating System Lab3 %
```

## Part 3 [ Process 1 ] Screenshot:

The screenshot shows the Visual Studio Code editor with the file `Process_P1.c` open. The code is a C program that opens two files, `destination1.txt` and `destination2.txt`, and prints their contents. The program uses `fdopen` to open the files and `fread` to read the contents. The terminal shows the output of the program, including the messages "Sucessfully created destination1.txt" and "Sucessfully created destination2.txt".

```
Part_3 > C Process_P1.c > main(int, char * [])
18     printf("Sucessfully created destination1.txt\n");
19     perror("open");
20     return 1;
21 }
22 else
23 {
24     printf("\n Sucessfully created destination1.txt\n");
25 }
26 int fd2;
27 fd2 = open ("destination2.txt", O_RDWR | O_CREAT, 0644);
28 if (-1 == fd2)
29 {
30     printf("\n Error [%s]\n", strerror(errno));
31     perror("open");
32     return 1;
33 }
34 else
35 {
36     printf("\n Sucessfully created destination2.txt\n");
37 }
38 close (fd1);
39 close (fd2);
40 return 0;
41 }
```

Terminal Output:

```
Sambeg@MacBook-Pro Part_3 % gcc Process_P1.c
Sambeg@MacBook-Pro Part_3 % ./a.out

Sucessfully created destination1.txt

Sucessfully created destination2.txt
Sambeg@MacBook-Pro Part_3 %
```

## Part 3 [ Process 2] Screenshot:

The screenshot shows the Visual Studio Code editor with the file `Process_P2.c` open. The code is a C program that opens `source.txt` and `destination1.txt`. The terminal shows the command `gcc Process_P2.c` and the output `Process completed`.

```
Part_3 > C: Process_P2.c > main(int, char * [])
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/stat.h>
4  #include <fcntl.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <unistd.h>
8
9  int main (int argc, char *argv[])
10 {
11     int fd;
12
13     fd = open("source.txt", O_RDONLY);
14     if (-1 == fd)
15     {
16         printf("\n Error detected [%s]\n", strerror(errno));
17         perror("open");
18         return 1;
19     }
20
21     //-----destination1.txt-----
22     int fd1;    (char [17])"destination1.txt"
23     fd1 = open ("destination1.txt", O_RDWR);
24     if (-1 == fd1)
25     {
26         printf("\n Error detected [%s]\n", strerror(errno));
27         perror("open");
28     }
29 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1: zsh
sambe@MacBook-Pro Part_3 % gcc Process_P2.c
sambe@MacBook-Pro Part_3 % ./a.out

Process completed
sambe@MacBook-Pro Part_3 %
```

## Destination1.txt

The screenshot shows the Visual Studio Code editor with the file `destination1.txt` open. The file contains a single line of text: `55543523523523515252352398325321211145902341841122`. The terminal shows the command `gcc Process_P2.c` and the output `Process completed`.

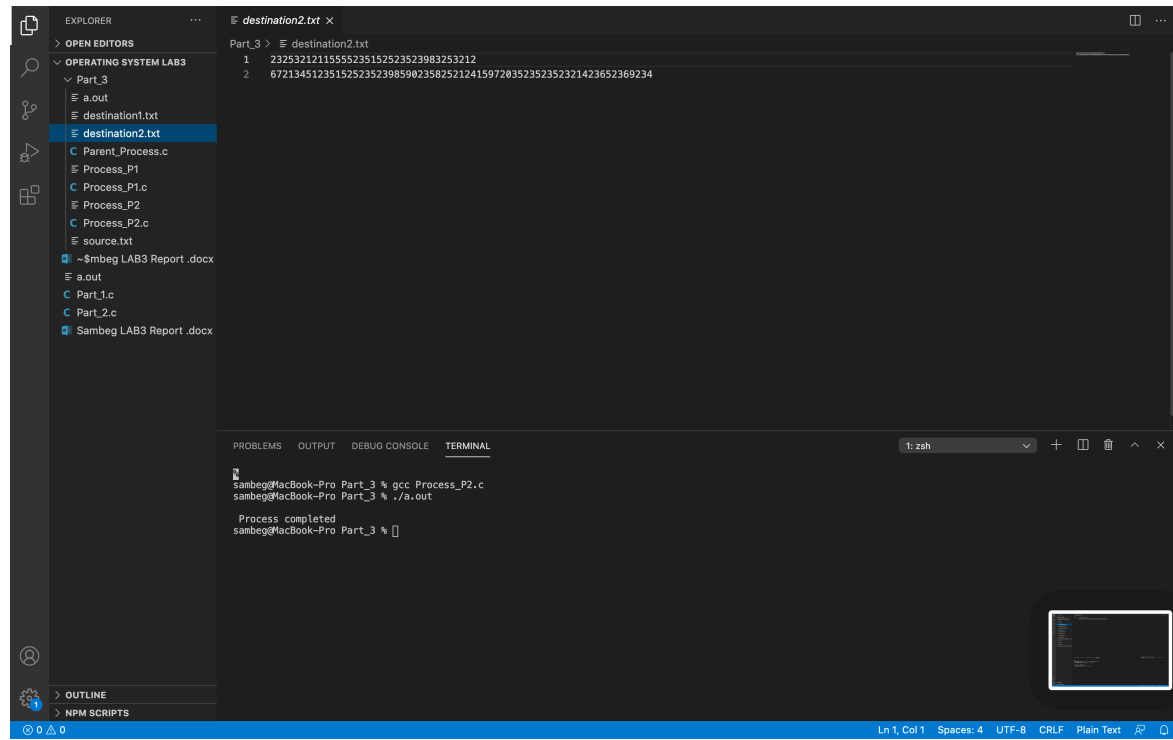
```
Part_3 > destination1.txt
1  55543523523523515252352398325321211145902341841122
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1: zsh
sambe@MacBook-Pro Part_3 % gcc Process_P2.c
sambe@MacBook-Pro Part_3 % ./a.out

Process completed
sambe@MacBook-Pro Part_3 %
```

## Destination2.txt

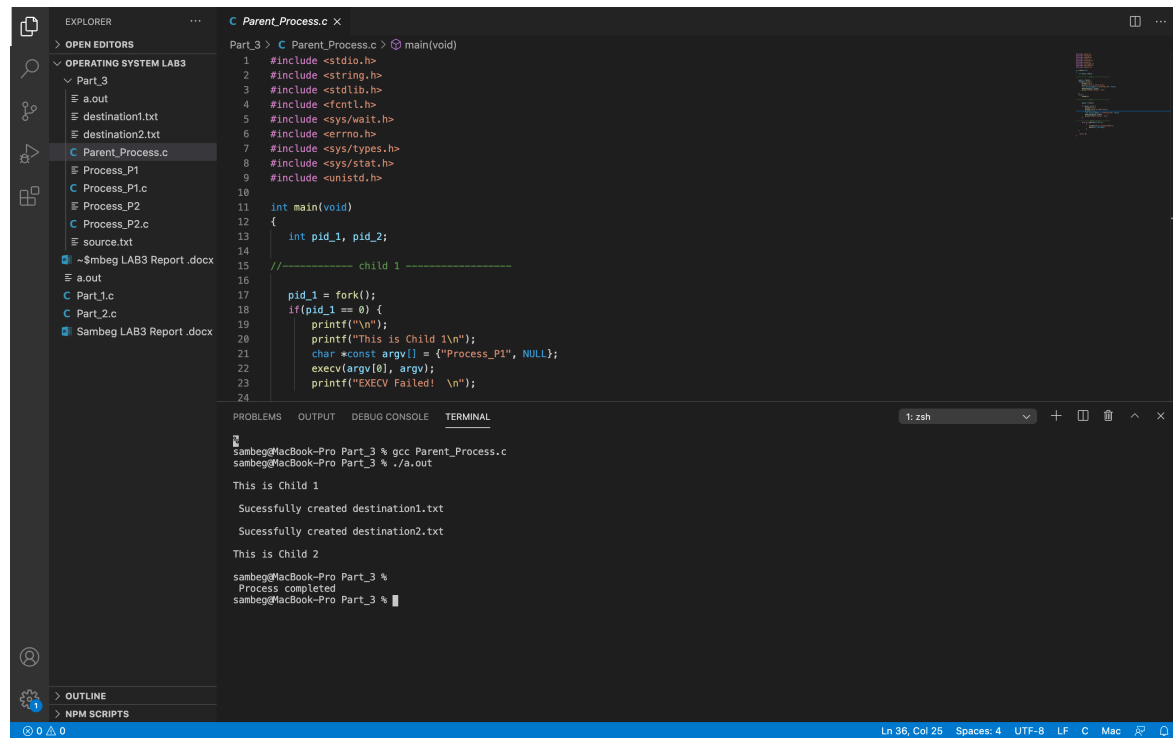


The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files under 'OPERATING SYSTEM LAB3', including 'Part\_3' and its sub-files 'a.out', 'destination1.txt', 'destination2.txt', 'Parent\_Process.c', 'Process\_P1', 'Process\_P1.c', 'Process\_P2', 'Process\_P2.c', 'source.txt', and report documents. The 'destination2.txt' file is selected and open in the editor, displaying two lines of hexadecimal data. The Terminal at the bottom shows the execution of 'gcc Process\_P2.c' and './a.out', with output indicating successful compilation and execution of 'Part\_3'.

```
Part_3 > destination2.txt
1  23253212115555235152523523983253212
2  672134512351525235239859023582521241597203523523212423652369234

Part_3 > C Parent_Process.c > main(void)
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <fcntl.h>
5  #include <sys/wait.h>
6  #include <errno.h>
7  #include <sys/types.h>
8  #include <sys/stat.h>
9  #include <unistd.h>
10
11 int main(void)
12 {
13     int pid_1, pid_2;
14
15     //----- child 1 -----
16
17     pid_1 = fork();
18     if(pid_1 == 0) {
19         printf("\n");
20         printf("This is Child 1\n");
21         char *const argv[] = {"Process_P1", NULL};
22         execv(argv[0], argv);
23         printf("EXECV Failed! \n");
24     }
```

## Part 3 [ Parent Process ] Screenshot:



The screenshot shows the Visual Studio Code interface with 'Parent\_Process.c' open in the editor. The code defines a main function that forks two child processes. The first child, 'child 1', prints a message and executes 'Process\_P1'. The second child, 'child 2', prints a message and executes 'Process\_P2'. The terminal output shows the successful execution of the program, with messages from both child processes and the completion of the parent process.

```
Part_3 > C Parent_Process.c > main(void)
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <fcntl.h>
5  #include <sys/wait.h>
6  #include <errno.h>
7  #include <sys/types.h>
8  #include <sys/stat.h>
9  #include <unistd.h>
10
11 int main(void)
12 {
13     int pid_1, pid_2;
14
15     //----- child 1 -----
16
17     pid_1 = fork();
18     if(pid_1 == 0) {
19         printf("\n");
20         printf("This is Child 1\n");
21         char *const argv[] = {"Process_P1", NULL};
22         execv(argv[0], argv);
23         printf("EXECV Failed! \n");
24     }
```