

Task - 1

The screenshot shows the Visual Studio Code interface with a C program named `task1.c` open in the editor. The program is designed to open a file named `anyfile.txt` for read/write access. It then enters a loop where it prompts the user to write something. If the user enters a non-empty string, it prints the string and prompts for another string. If the user enters an empty string, it prints a message and prompts for a string to write. The terminal output shows the program's execution, including the file opening message and the user's input/output.

```
task1.c
1 // ----- Task - 1 -----
2 #include <stdio.h>
3 #include <fcntl.h>
4 #include <unistd.h>
5 #include <string.h>
6
7 int main() {
8     int fd;
9     char str[100], str1[100];
10
11     fd = open("anyfile.txt", O_RDWR | O_CREAT, 0666);
12     printf("anyfile.txt opened for read/write access\n");
13
14     while (1) {
15         printf("Type '-1' to write something");
16         scanf("%s", str);
17
18         if (strcmp(str, "-1") == 0) {
19             printf("Please enter a string to write: ");
20             scanf("%s", str1);
21         }
22     }
23 }
```

Terminal Output:

```
d: command not found
omi@omi-ASUS:~/Desktop/codes/cse321/lab_assignments/assignment_1$ cd "/home/omi/Desktop/codes/cse321/lab_assignments/assignment_1/" && gcc task1.c -o task1 && "
/home/omi/Desktop/codes/cse321/lab_assignments/assignment_1/"task1
anyfile.txt opened for read/write access
Type '-1' to write something-1
Please enter a string to write: Omi
"Omi" was written to anyfile.txt
Type '-1' to write something-1
Please enter a string to write: Hello
"Hello" was written to anyfile.txt
Type '-1' to write something-1
Please enter a string to write: Haque
"Haque" was written to anyfile.txt
Type '-1' to write something0
File closed.
omi@omi-ASUS:~/Desktop/codes/cse321/lab_assignments/assignment_1$
```

Task - 2

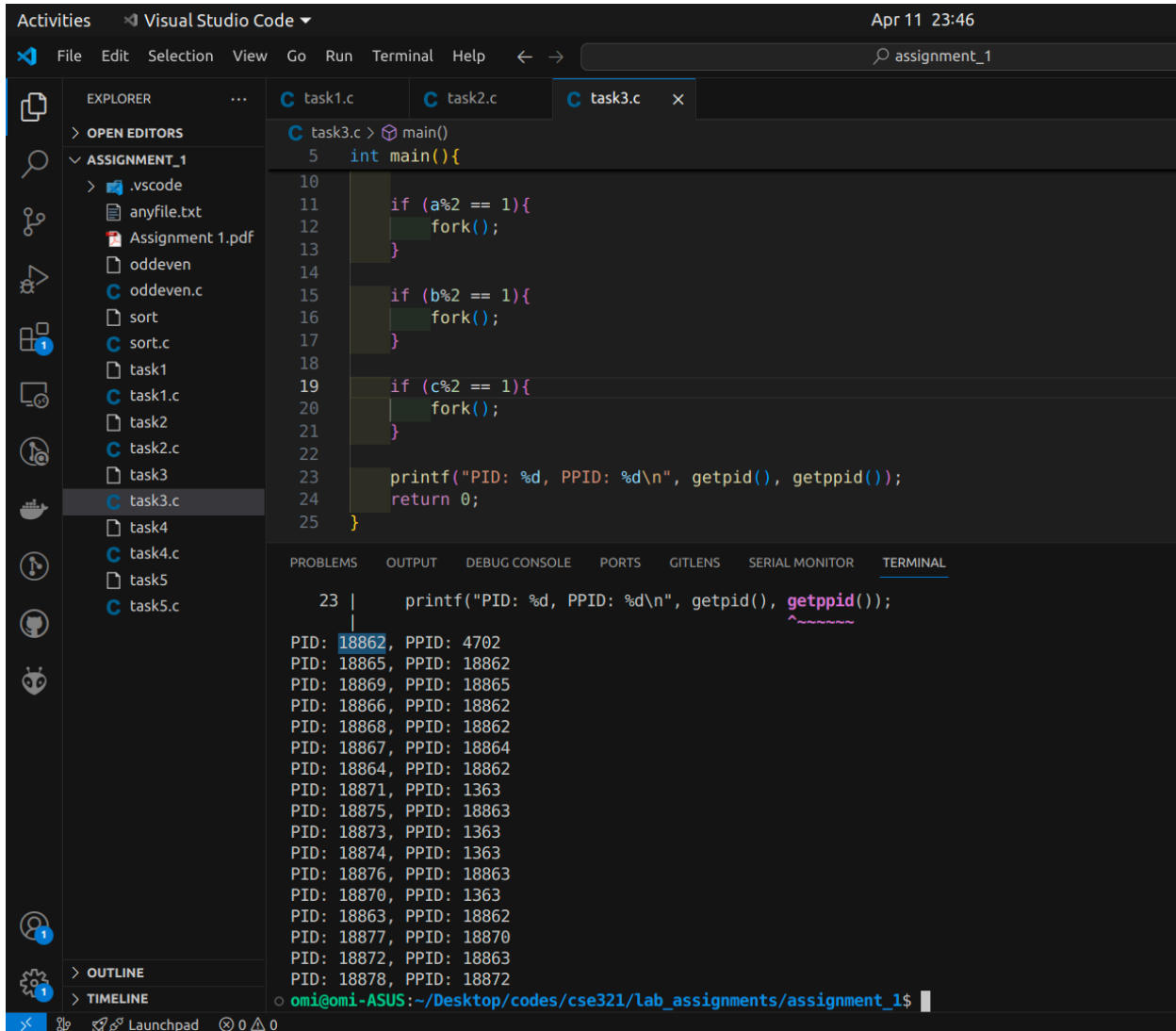
The screenshot shows the Visual Studio Code interface with a C program in `task2.c` and its terminal output. The Explorer sidebar on the left shows a project named `ASSIGNMENT_1` with various files, including `task2.c`. The main editor displays the code for `task2.c`, which implements a grandchild process using `fork()` and `wait()`. The terminal at the bottom shows the execution output, including compiler warnings and the program's runtime behavior.

```
1 // ----- Task - 1 -----
2
3 #include<stdio.h>
4 // #include<fcntl.h>
5
6
7 int main() {
8     int pid, pid2, status;
9     pid = fork();
10
11     if (pid < 0){
12         printf("Fork Failed\n");
13         exit(1);
14     }
15
16     if(pid == 0){
17         // Child Process
18         pid2 = fork();
19
20         if (pid2 == 0){
21             // Grandchild process
22             printf("I am grandchild \n");
23             exit(0);
24             // wait(&status);
25         }
26     }
27     wait(&status);
28
29     exit(0);
30 }
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS SERIAL MONITOR TERMINAL

```
task2.c:23:13: note: include '<stdlib.h>' or provide a declaration of 'exit'
task2.c:27:13: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
27 |     wait(&status);
   |     ^~~~~
task2.c:29:13: warning: incompatible implicit declaration of built-in function 'exit'
29 |     exit(0);
   |     ^~~~~
task2.c:29:13: note: include '<stdlib.h>' or provide a declaration of 'exit'
I am grandchild
I am child
I am parent
omi@omi-ASUS:~/Desktop/codes/cse321/lab_assignments/assignment_1$
```

Task - 3



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays a project named 'ASSIGNMENT_1' with several files, including 'task3.c'. The main editor window shows the code in 'task3.c', which defines a `main()` function that forks three child processes based on odd/even checks for variables `a`, `b`, and `c`. The terminal at the bottom shows the output of the program, displaying the PID and PPID for the parent and the three child processes.

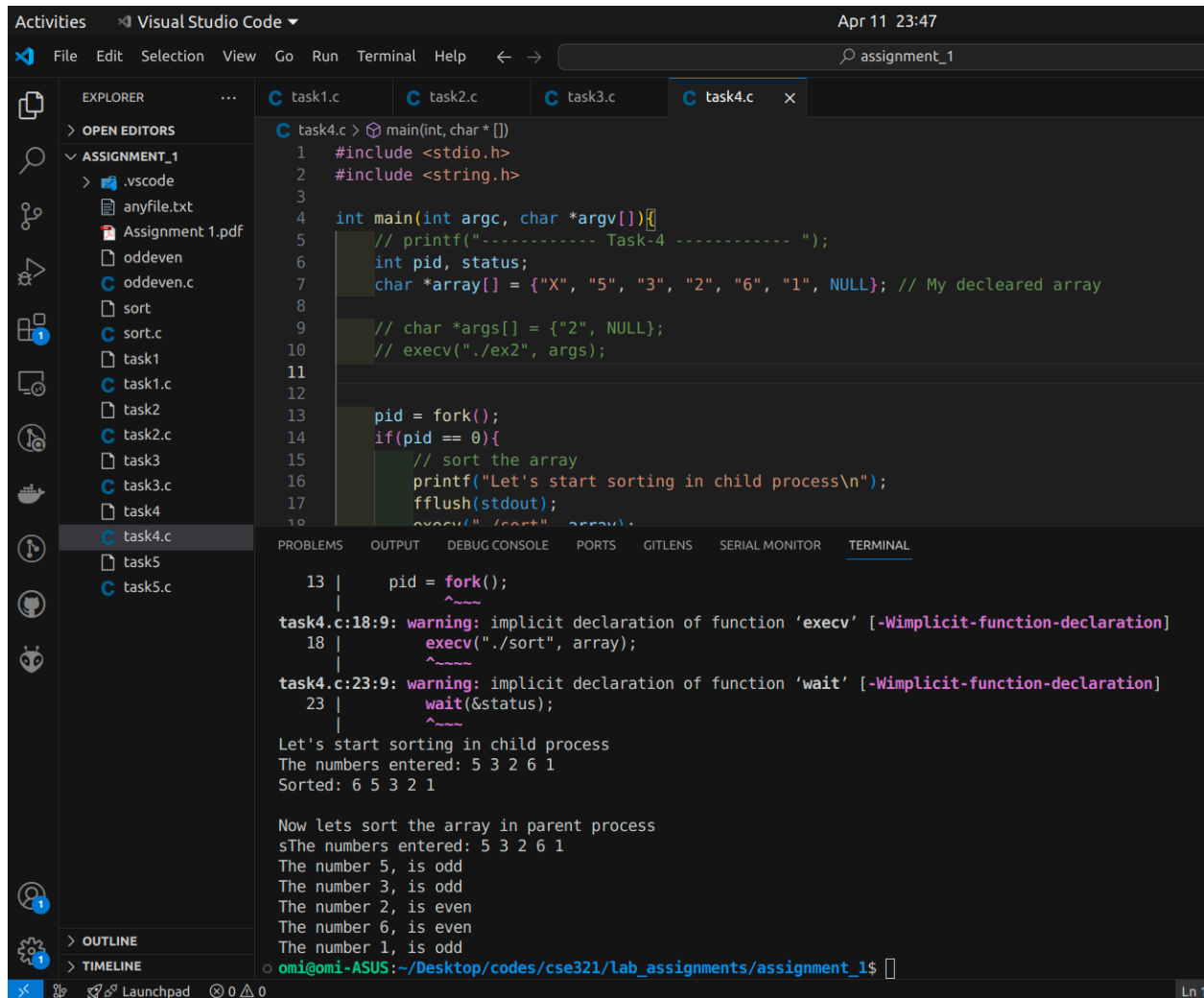
```
task3.c > main()
5  int main(){
10
11      if (a%2 == 1){
12          fork();
13      }
14
15      if (b%2 == 1){
16          fork();
17      }
18
19      if (c%2 == 1){
20          fork();
21      }
22
23      printf("PID: %d, PPID: %d\n", getpid(), getppid());
24      return 0;
25  }
```

Terminal Output:

```
23 | printf("PID: %d, PPID: %d\n", getpid(), getppid());
PID: 18862, PPID: 4702
PID: 18865, PPID: 18862
PID: 18869, PPID: 18865
PID: 18866, PPID: 18862
PID: 18868, PPID: 18862
PID: 18867, PPID: 18864
PID: 18864, PPID: 18862
PID: 18871, PPID: 1363
PID: 18875, PPID: 18863
PID: 18873, PPID: 1363
PID: 18874, PPID: 1363
PID: 18876, PPID: 18863
PID: 18870, PPID: 1363
PID: 18863, PPID: 18862
PID: 18877, PPID: 18870
PID: 18872, PPID: 18863
PID: 18878, PPID: 18872
```

More than 8 Process will be created, In My case 17 process here.

Task - 4



Activities Visual Studio Code Apr 11 23:47

File Edit Selection View Go Run Terminal Help ← → assignment_1

task1.c task2.c task3.c task4.c

task4.c

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main(int argc, char *argv[])
5 {
6     // printf("----- Task-4 ----- ");
7     int pid, status;
8     char *array[] = {"X", "5", "3", "2", "6", "1", NULL}; // My declared array
9
10    // char *args[] = {"2", NULL};
11    // execv("./ex2", args);
12
13    pid = fork();
14    if(pid == 0){
15        // sort the array
16        printf("Let's start sorting in child process\n");
17        fflush(stdout);
18        execv("./sort", array);
19    }
20
21    wait(&status);
22
23    printf("Let's start sorting in parent process\n");
24    printf("The numbers entered: 5 3 2 6 1\n");
25    printf("Sorted: 6 5 3 2 1\n");
26
27    printf("Now lets sort the array in parent process\n");
28    printf("The numbers entered: 5 3 2 6 1\n");
29    printf("The number 5, is odd\n");
30    printf("The number 3, is odd\n");
31    printf("The number 2, is even\n");
32    printf("The number 6, is even\n");
33    printf("The number 1, is odd\n");
34}
```

task4.c:18:9: warning: implicit declaration of function 'execv' [-Wimplicit-function-declaration]

task4.c:23:9: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]

Let's start sorting in child process

The numbers entered: 5 3 2 6 1

Sorted: 6 5 3 2 1

Now lets sort the array in parent process

sThe numbers entered: 5 3 2 6 1

The number 5, is odd

The number 3, is odd

The number 2, is even

The number 6, is even

The number 1, is odd

omi@omi-ASUS:~/Desktop/codes/cse321/lab_assignments/assignment_1\$

Task - 5

Activities Visual Studio Code Apr 11 23:48

File Edit Selection View Go Run Terminal Help ← → assignment_1

task1.c task2.c task3.c task4.c task5.c

task5.c > main()

```
1 // ----- Task-5 -----
2 #include <stdio.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6
7 int main(){
8     int pid, pid2, status;
9
10    pid = fork();
11
12    if (pid < 0){
13        printf("Fork Failed\n");
14    }
15
16    if (pid == 0){
17        printf("2. Child Process ID: %d\n", getpid());
18        for(int i = 0; i < 2; i++){
19            // ...
20        }
21    }
22
23    // ...
24
25    // ...
26
27    // ...
28    exit(0);
29
30    // ...
31    wait(NULL);
32
33    // ...
34    return 0;
35 }
```

PROBLEMS OUTPUT DEBUG CONSOLE PORTS GITLENS SERIAL MONITOR TERMINAL

omi@omi-ASUS:~/Desktop/codes/cse321/lab_assignments/assignment_1\$ cd "/home/omi/Desktop/codes/cse321/lab_assignments/assignment_1/" && gcc task5.c -o task5 && ./task5

task5.c: In function 'main':

task5.c:28:17: warning: implicit declaration of function 'exit' [-Wimplicit-function-declaration]

```
28 |         exit(0);
   |         ^~~~~
```

task5.c:28:17: warning: incompatible implicit declaration of built-in function 'exit'

task5.c:6:1: note: include '<stdlib.h>' or provide a declaration of 'exit'

```
5 | #include <sys/types.h>
+++ |+#include <stdlib.h>
6 |
```

task5.c:31:17: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]

```
31 |     wait(NULL);
   |     ^~~~~
```

1. Parent Process ID: 19027
2. Child Process ID: 19028
3. Grandchild Process ID: 19029
4. Grandchild Process ID: 19030
5. Grandchild Process ID: 19031

omi@omi-ASUS:~/Desktop/codes/cse321/lab_assignments/assignment_1\$

Ln 8, Co