Laboratory Assignment 4

On

Design Principles of Operating System

(CSE 3249)

Submitted by

Name	**: Dinanath Dash**
Reg. No.	**: 2241004161**
Semester	**: 5th**
Branch	**: CSE**
Section	**: 2241026**
Session	**: 2024-2025**
Admission Batch	**: 2022**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

FACULTY OF ENGINEERING & TECHNOLOGY (ITER)

SIKSHA 'O' ANUSANDHAN DEEMED TO BE UNIVERSITY

BHUBANESWAR, ODISHA – 751030

Assignment 4: Familiarization with Process Management in Linux environment.

Objective of this Assignment:

- To trace the different states of a process during its execution
- To learn the use of different system calls such as (fork(),vfork(),wait(),execl()) for process handling in Unix/Linux environment.

1. Write a C program to create a child process using fork () system call. The child process will print the message “Child” with its process identifier and then continue in an indefinite loop. The parent process will print the message “Parent” with its process identifier and then continue in an indefinite loop.
a) Run the program and trace the state of both processes.
b) Terminate the child process. Then trace the state of processes.
c) Run the program and trace the state of both processes. Terminate the parent process. Then trace the state of processes.
d) Modify the program so that the parent process after displaying the message will wait for child process to complete its task. Again, run the program and trace the state of both processes.
e) Terminate the child process. Then trace the state of processes.

Output -

a)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
int main() {
    pid_t pid = fork();

    if (pid < 0) {
        // Fork failed
        perror("Fork failed");
        return 1;
    } else if (pid == 0) {
        // Child process
        printf("Child: PID = %d\n", getpid());
        while (1) {
            // Infinite loop for child process
        }
    } else {
        // Parent process
        printf("Parent: PID = %d\n", getpid());
        while (1) {
            // Infinite loop for parent process
        }
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q1.c&
[2] 6381
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q1.c
[2]+  Done                    gedit q1.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Parent: PID = 6395
Child: PID = 6396
^C
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ps -al
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000     541     436  0  80   0 -  1518 core_s pts/1    00:00:00 bash
0 R  1000    6404    6307  0  80   0 -  2078 -      pts/0    00:00:00 ps
[1]+  Done                    gedit q1.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

b)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int main() {
    pid_t pid = fork();

    if (pid < 0) {
        // Fork failed
        perror("Fork failed");
        return 1;
    } else if (pid == 0) {
        // Child process
        printf("Child: PID = %d\n", getpid());
        sleep(5); // Simulate some work in the child process
        printf("Child: Completed task\n");
        exit(0);  // Child process terminates
    } else {
        // Parent process
        printf("Parent: PID = %d\n", getpid());
        wait(NULL); // Parent waits for the child process to complete
        printf("Parent: Child has completed\n");
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q1d.c&
[2] 589
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q1d.c
[2]+  Done                    gedit q1d.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Parent: PID = 600
Child: PID = 601
Child: Completed task
Parent: Child has completed
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

2. Trace the output of the following codes:

a)

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    if (fork() == 0)
        printf("1");
    else
        printf("2");
    printf("3");
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2a.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2a.c
[1]+  Done                    gedit q1.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
2313dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

b)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    if (vfork() == 0) {
        printf("1");
        _exit(0);
    } else {
        printf("2");
    }
    printf("3");
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2b.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2b.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
123dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

c)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    pid_t pid;
    int i = 5;
    pid = fork();
    i = i + 1;
    if (pid == 0) {
        printf("Child: %d\n", i);
    } else {
        wait(NULL);
        printf("Parent: %d\n", i);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2c.c&
[1] 726
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2c.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child: 6
Parent: 6
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

d)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    pid_t pid;
    int i = 5;
    pid = vfork();
    i = i + 1;
    if (pid == 0) {
        printf("Child: %d\n", i);
        _exit(0);
    } else {
        printf("Parent: %d\n", i);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2d.c&
[2] 755
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2d.c
[2]+  Done                    gedit q2d.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child: 6
Parent: 7
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

e)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    pid_t pid;
    int i = 5;
    pid = fork();
    if (pid == 0) {
        i = i + 1;
        printf("Child: %d\n", i);
    } else {
        wait(NULL);
        printf("Parent: %d\n", i);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2e.c&
[2] 775
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2e.c
[2]+  Done                    gedit q2e.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child: 6
Parent: 5
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

f)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    pid_t pid;
    int i = 5;
    pid = vfork();
    if (pid == 0) {
        i = i + 1;
        printf("Child: %d\n", i);
        _exit(0);
    } else {
        printf("Parent: %d\n", i);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2f.c&
[2] 619
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2f.c
[2]+  Done                    gedit q2f.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child: 6
Parent: 6
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

g)

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    int i = 5;
    if (fork() == 0) {
        printf("Child: %d\n", i);
    } else {
        printf("Parent: %d\n", i);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2g.c&
[2] 662
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2g.c
[2]+  Done                    gedit q2g.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Parent: 5
Child: 5
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

h)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    int i = 5;
    if (vfork() == 0) {
        printf("Child: %d\n", i);
        _exit(0);
    } else {
        printf("Parent: %d\n", i);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2h.c&
[2] 678
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2h.c
[2]+  Done                    gedit q2h.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child: 5
Parent: 5
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

i)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    if (fork() == 0) {
        printf("1");
    } else {
        wait(NULL);
        printf("2");
    }
    printf("3");
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2i.c&
[2] 706
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2i.c
[2]+  Done                    gedit q2i.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
1323dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

j)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    if (vfork() == 0) {
        printf("1");
        _exit(0);
    } else {
        printf("2");
    }
    printf("3");
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2j.c&
[2] 727
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2j.c
[2]+  Done                    gedit q2j.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
123dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

k)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    pid_t c1;
    int n = 10;
    c1 = fork();
    if (c1 == 0) {
        printf("Child\n");
        n = 20;
        printf("n=%d\n", n);
    } else {
        wait(NULL);
        printf("Parent\n");
        printf("n=%d\n", n);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2k.c&
[2] 743
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2k.c
[2]+  Done                    gedit q2k.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child
n=20
Parent
n=10
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

l)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    pid_t c1;
    int n = 10;
    c1 = vfork();
    if (c1 == 0) {
        printf("Child\n");
        n = 20;
        printf("n=%d\n", n);
        _exit(0);
    } else {
        printf("Parent\n");
        printf("n=%d\n", n);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2l.c&
[2] 760
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2l.c
[2]+  Done                    gedit q2l.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child
n=20
Parent
n=20
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

m)

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    int i = 5;
    fork();
    i = i + 1;
    fork();
    printf("%d\n", i);
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2m.c&
[2] 776
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2m.c
[2]+  Done                    gedit q2m.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
6
6
6
6
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

n)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    pid_t pid;
    int i = 5;
    pid = vfork();
    if (pid == 0) {
        printf("Child: %d\n", i);
        _exit(0);
    } else {
        i = i + 1;
        printf("Parent: %d\n", i);
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2n.c&
[2] 818
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2n.c
[2]+  Done                    gedit q2n.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Child: 5
Parent: 6
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

o)

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    int i = 5;
    if (fork() == 0) {
        i = i + 1;
    } else {
        i = i - 1;
    }
    printf("%d\n", i);
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2o.c&
[2] 1252
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2o.c
[2]+  Done                    gedit q2o.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
4
6
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

p)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main() {
    int i = 5;
    if (vfork() == 0) {
        i = i + 1;
        _exit(0);
    } else {
        i = i - 1;
    }
    fprintf(stderr, "%d\n", i);
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2p.c&
[2] 1272
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2p.c
[2]+  Done                    gedit q2p.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
5
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

q)

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int j, i = 5;
    for (j = 1; j < 3; j++) {
        if (fork() == 0) {
            i = i + 1;
            break;
        } else {
            wait(NULL);
        }
    }
    printf("%d\n", i);
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2q.c&
[2] 1289
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2q.c
[2]+  Done                    gedit q2q.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
6
6
5
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

r)

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    int j, i = 5;
    for (j = 1; j < 3; j++) {
        if (fork() != 0) {
            i = i - 1;
            break;
        }
    }
    fprintf(stderr, "%d\n", i);
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2r.c&
[2] 1307
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2r.c
[1]-  Done                    gedit q2f.c
[2]+  Done                    gedit q2r.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
4
4
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ 5
```

s)

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    if (fork() == 0) {
        if (fork()) {
            printf("1\n");
        }
    }
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2s.c&
[1] 1351
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2s.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
1
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

t)

```c
#include <stdio.h>
#include <unistd.h>

void fun1() {
    fork();
    fork();
    printf("1\n");
}

int main() {
    fun1();
    printf("1\n");
    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q2t.c&
[2] 1369
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q2t.c
[2]+  Done                    gedit q2t.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
1
1
1
1
1
1
1
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ 1
```

3. Write a C program that will create three child process to perform the following operations respectively:

- First child will copy the content of file1 to file2
- Second child will display the content of file2
- Third child will display the sorted content of file2 in reverse order.
- Each child process being created will display its id and its parent process id with appropriate message.
- The parent process will be delayed for 1 second after creation of each child process. It will display appropriate message with its id after completion of all the child processes.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <sys/wait.h>

void copy_file() {
    int src = open("file1.txt", O_RDONLY);
    int dest = open("file2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    char buf[1024];
    ssize_t bytes;
```

```
    while ((bytes = read(src, buf, sizeof(buf))) > 0)
        write(dest, buf, bytes);
    close(src);
    close(dest);
}

void display_file(const char *filename) {
    char buf[1024];
    int fd = open(filename, O_RDONLY);
    while (read(fd, buf, sizeof(buf)) > 0)
        write(STDOUT_FILENO, buf, strlen(buf));
    close(fd);
}

void display_sorted_reverse() {
    FILE *file = fopen("file2.txt", "r");
    char *lines[100];
    size_t n = 0;
    char buf[1024];

    while (fgets(buf, sizeof(buf), file))
       lines[n++] = strdup(buf);
   fclose(file);

   for (size_t i = 0; i < n - 1; i++)
       for (size_t j = 0; j < n - i - 1; j++)
           if (strcmp(lines[j], lines[j + 1]) < 0) {
               char *temp = lines[j];
               lines[j] = lines[j + 1];
               lines[j + 1] = temp;
           }

   for (size_t i = 0; i < n; i++) {
       printf("%s", lines[i]);
       free(lines[i]);
   }
}

int main() {
    if (fork() == 0) {
        printf("First Child: Copying file content\n");
        copy_file();
        exit(0);
    }
    wait(NULL);

    if (fork() == 0) {
        printf("Second Child: Displaying file content\n");
        display_file("file2.txt");
        exit(0);
    }
    wait(NULL);
     if (fork() == 0) {
         printf("Third Child: Displaying sorted reverse content\n");
         display_sorted_reverse();
         exit(0);
     }
     wait(NULL);

     printf("Parent: All tasks completed.\n");
     return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q3.c&
[2] 1428
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q3.c
[2]+  Done                    gedit q3.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
First Child: Copying file content
Second Child: Displaying file content
10
20
30
◆◆◆Third Child: Displaying sorted reverse content
30
20
10
Parent: All tasks completed.
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```

4. Write a C program that will create a child process to generate a Fibonacci series of specified length and store it in an array. The parent process will wait for the child to complete its task and then display the Fibonacci series and then display the prime Fibonacci number in the series along with its position with appropriate message.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

// Function to check if a number is prime
int is_prime(int num) {
    if (num <= 1) return 0;
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) return 0;
    }
    return 1;
}

// Function to generate Fibonacci series
void generate_fibonacci(int *fib, int length) {
    fib[0] = 0;
    fib[1] = 1;
    for (int i = 2; i < length; i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
}

int main() {
    int length;
    printf("Enter the length of the Fibonacci series: ");
    scanf("%d", &length);

    if (length < 2) {
        printf("Length should be at least 2.\n");
        return 1;
    }

    int fib[length];
    pid_t pid = fork();

    if (pid == 0) {
        // Child Process: Generate Fibonacci series
        generate_fibonacci(fib, length);
        printf("Child Process: Fibonacci series generated.\n");
        exit(0);
    } else if (pid > 0) {
        // Parent Process: Wait for child to complete
        wait(NULL);
        printf("Parent Process: Fibonacci series:\n");
        generate_fibonacci(fib, length);

        for (int i = 0; i < length; i++) {
            printf("%d ", fib[i]);
        }
        printf("\n");

        // Identify and display prime Fibonacci numbers
        printf("Parent Process: Prime Fibonacci numbers:\n");
        for (int i = 0; i < length; i++) {
            if (is_prime(fib[i])) {
                printf("Prime: %d at position %d\n", fib[i], i);
            }
        }
    } else {
        perror("Fork failed");
        return 1;
    }

    return 0;
}
```

```
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gedit q4.c&
[2] 1444
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ gcc q4.c
[2]+  Done                    gedit q4.c
dinanath@DINANATH:~/DOS_2241004161/DOSass4$ ./a.out
Enter the length of the Fibonacci series: 5
Child Process: Fibonacci series generated.
Parent Process: Fibonacci series:
0 1 1 2 3
Parent Process: Prime Fibonacci numbers:
Prime: 2 at position 3
Prime: 3 at position 4
dinanath@DINANATH:~/DOS_2241004161/DOSass4$
```