Objectives

- To create parent-child processes using fork() system call.
- To create a process chain and prevent child processes from becoming zombie using wait() system call.
- To create a process fan and prevent child processes from becoming zombies using a wait() system call.
- To create a process tree and prevent child processes from becoming zombie using wait() system call.
- To execute a new process from a process using an exec() system call.

Pre-Lab Tasks

1. Testing of gcc compiler

i) type a source file with printf "helloworld" msg ii) compilation statement : *$gcc -o helloworld.out helloworld.c*

1. Display of command line arguments

Type the code below in the file cmdargs.c, and then compile and execute with arguments from the command line and write the output

```
int main(int argc,char* argv[]){
int i; for(i=0;i<argc;i++)
printf("%s\n",argv[i]);
}
```

In-Lab Tasks

header file*: unistd.h, stdio.h*

System call: *fork(), getpid(), getppid(),wait()*

Task1: Compile and run the following program and study its behavior a)

Write the output of the child process

b) Write the output of the parent process

```
└$ gcc q1.c -o q1.o

┌──(tayyab123㉿kali)-[~/Desktop/faizan]
└─$ ./q1.o
I am parent: my process id is 2992 and my child process id is 2993
I am child: my process id is 2993 and my parent process id is 2992
```

c) Mention any unusual thing noticed in the output of child process

```
#include <stdio.h>
#include <unistd.h>
 int main() {
int pid; pid
= fork();
if(pid > 0)
```

```
printf("I am parent: my process id is %d and my child
process id is %d\n", getpid(),pid);


else if (pid == 0)

printf("I am child: my process id is %d and my parent
process id is %d\n",getpid(), getppid()); else

printf("ERROR in executing fork()");

 return

0;

}
```

Task2a: Compile and run the following code and write the output. From the output Also draw the process interconnected diagram to show process chain or fan or tree.

```
#include <stdio.h>

#include <unistd.h>


int main(int argc,char* argv[]) {

int pid; int i; for(i=0;i<4;i++){

pid = fork();      if(pid > 0)

         break;

     else if(pid == 0)

     continue;     else

         printf("ERROR: In fork()");

}//end of for

printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());
```

```
}//end of main
```

```
└─$ ./q2a.o
My process id is 3825 and my Parent process id is1833
My process id is 3826 and my Parent process id is1047

My process id is 3827 and my Parent process id is1047
My process id is 3828 and my Parent process id is3827
My process id is 3829 and my Parent process id is3828
```

Task 2b: Modify the code to get the number of process value from the command line argument. Compile and run the program with number of process value other than 4 and write the output.

```
#include <stdio.h>

#include <unistd.h>


int main(int argc,char* argv[]) {

int pid; int i;




printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());

}//end of main
```

Output:

```
└─$ ./q2b.c 3
My process id is 4166 and my Parent process id is 1833
I am parent: my process id is 4166 and my child process id is 4167
I am parent: my process id is 4166 and my child process id is 4168
I am child: my process id is 4168 and my parent process id is 4166
I am parent: my process id is 4166 and my child process id is 4169
I am child: my process id is 4167 and my parent process id is 4166
I am child: my process id is 4169 and my parent process id is 1047
```

```
argv[]) {

ser provided exactly one argument (the number of processes
<number_of_processes>\n", argv[0]);
urn error code 1 for improper usage
```

Task 2c: Write the modified code to prevent the parent process from terminating before the child processes through wait() system call and also write the output system call: wait()

```
#include <stdio.h>

#include <unistd.h>


int main(int argc,char* argv[]) {

int pid; int i;




printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());

}//end of main
```

Output:

Task 3a: Compile and run the following code and write the output. From the output Also draw the process interconnected diagram to show process chain or fan or tree.

```
#include <stdio.h>

#include <unistd.h>
```

```
int main(int argc,char* argv[]) {
int pid; int i; for(i=0;i<4;i++){
pid = fork();      if(pid > 0)
continue;
     else if(pid == 0)
          break;
else
          printf("ERROR: In fork()");
}//end of for
printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());
}//end of main
```

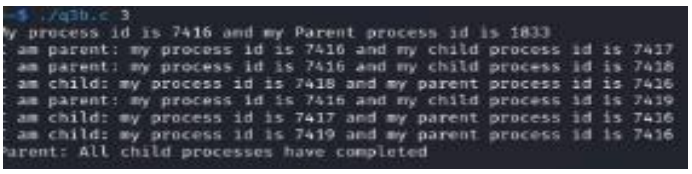
Output:

```
└$ ./q3a.c
My process id is 6876 and my Parent process id is6875
My process id is 6875 and my Parent process id is1833

My process id is 6879 and my Parent process id is1047
My process id is 6878 and my Parent process id is1047
My process id is 6877 and my Parent process id is1047
```

Task 3b: Write the modified code to get the number of process value from the command line argument. Compile and run the program with a number of process value other than 4 and write the output.

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

```
int main(int argc,char* argv[]) {
int pid; int i;

printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());
}//end of main
```



Output:

Task 3c: Write the modified code to prevent the parent process from terminating before the child processes through wait() system call and also write the output system call: wait()

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

int main(int argc,char* argv[]) {
int pid; int i;

printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());
}//end of main
```

Output:

```
└$ ./q3c.c
Parent: Created child with process id 7602
Parent: Created child with process id 7603
Child: My process id is 7603 and my Parent process id is 7601
Parent: Created child with process id 7604
Child: My process id is 7602 and my Parent process id is 7601
Parent: Created child with process id 7605
Child: My process id is 7604 and my Parent process id is 7601
Child: My process id is 7605 and my Parent process id is 7601
Parent: All child processes have terminated. My process id is 7601
```

Task 4a: Creating the Process Tree, write the code and output similarly as in process chain and process fan with n=4

```
#include <stdio.h>

#include <unistd.h>


int main(int argc,char* argv[]) {

int pid; int i;




printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());

}//end of main
```

Output:

```
└$ ./q4a.c
Parent: Created child with process id 7760
Parent: Created child with process id 7761
Child: My process id is 7760 and my Parent process id is 7759
Parent: Created child with process id 7762
Child: My process id is 7761 and my Parent process id is 7759
Parent: Created child with process id 7763
Child: My process id is 7762 and my Parent process id is 7759
Child: My process id is 7763 and my Parent process id is 7759
```

Task4c: In Process Tree with n=4, Prevent the parent process from terminating before child processes and write the modified code and output system call: wait()

hint: `while(wait()>0);`

```
#include <stdio.h> #include
<unistd.h> int main(int
argc,char* argv[]) { int pid; int
i;
```

```
printf("My process id is %d and my Parent process id is
%d\n",getpid(),getppid());
```

}//end of main *Output:*

```
└$ ./q4c.c
Parent: Created child with process id 8375
Child: My process id is 8375 and my Parent process id is 8374
Parent: Created child with process id 8376
Child: My process id is 8376 and my Parent process id is 8375
Parent: Created child with process id 8377
Child: My process id is 8377 and my Parent process id is 8376
Parent: Created child with process id 8378
Child: My process id is 8378 and my Parent process id is 8377
```

Task5: write the code to create a child process and execute ls -l command in the child process system call: exec() hint: explore the variants exec() call

```
└$ ./q5.c
Child: Executing 'ls -l' command
total 164
-rw-r--r-- 1 tayyab123 tayyab123   797 Sep 25 22:19 q1.c
-rwxr-xr-x 1 tayyab123 tayyab123 16104 Sep 25 21:09 q1.o
-rwxr-xr-x 1 tayyab123 tayyab123 16104 Sep 25 21:13 q2a.o
-rwxr-xr-x 1 tayyab123 tayyab123 16200 Sep 25 21:16 q2b.c
-rwxr-xr-x 1 tayyab123 tayyab123 16296 Sep 25 21:20 q2c.c
-rwxr-xr-x 1 tayyab123 tayyab123 16104 Sep 25 21:49 q3a.c
-rwxr-xr-x 1 tayyab123 tayyab123 16296 Sep 25 21:58 q3b.c
-rwxr-xr-x 1 tayyab123 tayyab123 16200 Sep 25 22:01 q3c.c
-rwxr-xr-x 1 tayyab123 tayyab123 16200 Sep 25 22:04 q4a.c
-rwxr-xr-x 1 tayyab123 tayyab123 16200 Sep 25 22:12 q4c.c
-rwxr-xr-x 1 tayyab123 tayyab123 16144 Sep 25 22:19 q5.c
Parent: Child process finished
```