

Lab Exercise 2

Process Forking

Name: Abhinav Pandey

Roll No. : AM.EN.U4AIE21088

1. What is the output of the following code that you are getting?

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

int main ()
{
    int id, ret;

    ret = fork();
    id = getpid();
    printf("\n My identifier is ID = [%d]\n", id);
```

Answer :

```
1
2
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 int main ()
7 {
8     int id, ret;
9     ret = fork () ;
10    id = getpid();
11    printf ("\n My identifier is ID =[%d]\n",id);
12 }
13
14
```



```
My identifier is ID =[675]
```

2. What is the output of the following code? How many processes are being created including the parent process? Draw the process graph to trace the fork calls.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int
main ()
{
    int id, ret;
    ret = fork ();
    ret = fork ();
    id = getpid ();
    printf ("\n My identifier is ID =[%d] \n", id);
    wait(NULL);
    return 0;
}
```

Answer:

main.c

```
1 //Abhinav Pandey
2
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 int main ()
7 {
8     int id, ret;
9     ret = fork ();
10    ret = fork ();
11    id = getpid ();
12    printf ("\n My identifier is ID =[%d] \n", id);
13    wait (NULL);
14    return 0;
15 }
16
```



My identifier is ID =[1193]

My identifier is ID =[1194]

3. What is the output of the following code?

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

void fork3()
{
    int ret;

    ret = fork();
    if (ret == 0)
        printf("\n [%d] Hello from child", getpid());
    else
        printf("\n [%d] Hello from parent", getpid());
}

int main ()
{
    fork3();
    return 0;
}
```

Answer:

```

1 //Abhinav Pandey
2
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 void fork3 ()
7 {
8     int ret;
9     ret = fork () ;
10    if (ret == 0)
11        printf ("\n [%d] Hello from child", getpid());
12    else printf ("\n [%d] Hello from parent", getpid());
13
14 }
15 int main (){
16     fork3();
17     return 0;
18 }
19

```



```
[321] Hello from parent
```

4. What is the output of the following code? How many processes are being created including the parent process? Draw the process graph to trace the fork calls.

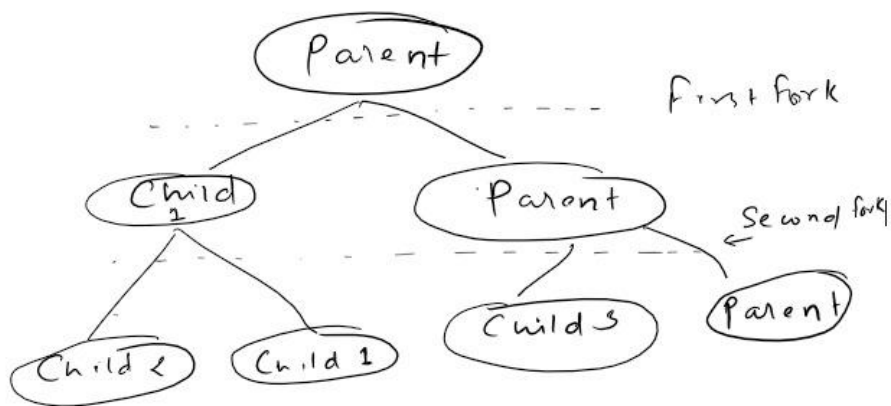
```

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
void
fork4 ()
{
    printf ("\n[%d] L0 \n", getpid ());
    fork ();
    printf ("\n[%d] L1 \n", getpid ());
    fork ();
    printf ("\n[%d] bye \n", getpid ());
}

int
main ()
{
    fork4 ();
    return 0;
}

```

Answer:



```

main.c
1 //Abhinav Pandey
2
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <sys/types.h>
6 void fork4 ()
7 {
8     printf ("\n[%d] L0 \n", getpid ());
9     fork ();
10    printf ("\n[%d] L1 \n", getpid ());
11    fork ();
12    printf ("\n[%d] bye \n", getpid ());
13 }
14 int
15 main ()
16 {
17     fork4();
18     return 0;
19 }
20

```

```

[422] L0
[422] L1
[426] L1
[422] bye

```


5. Write a C program called `sumfact.c` that does the following:
1. Takes an integer argument (say, $N1$) from the command line.
 2. Forks two children processes
 - a. First child computes $1+2+\dots+N1$ (sum of positive integers up to $N1$) and prints out the result and its own process ID.
 - b. Second child computes $1*2*\dots*N1$ (the factorial of $N1$) and prints out the result and its own process ID.
 3. Parent waits until both children are finished, then prints out the message "Done" and its own process ID.

Answer:

```
main.c
1  #include <unistd.h>
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  int main(int argc, char *argv[])
6  {
7      int N1 = atoi(argv[1]);
8      int sum1=0,prod2=1,flag1,flag2;
9      int c1_pid, c2_pid,fork1,fork2;
10     fork1 = fork();
11     fork2 = fork();
12     if(fork1 > 0){
13         for(int check=1;check <= N1;check++){
14             sum1 += check;
15         }
16         printf("Sum is %d",sum1);
17         flag1 =1;
18     }
19     if (fork2 > 0)
20     {
21         for (int check = 1; check <= N1; check++)
22         {
23             prod2 *= check;
24         }
25         printf("Product is %d", prod2);
26         flag2=1;
27     }
28     if(flag1&&flag2==1) printf("\nDone");
29     return 0;
30 }
```