

SYSTEM CALLS

1)

Code:

```
#include <stdio.h>

#include <unistd.h>
int main()
{
    int t1=0,t2=1,n,i,j,temp,count=1,flag,nextTerm,k=2;
    nextTerm=t1+t2;
    printf("Enter size n -----");
    scanf("%d",&n);
    temp=fork();

    if(temp==0)
    {
        printf("(Child)Fibonacci Series:%d,%d,",t1,t2);

        for(i=3;i<=n;++i){
            printf("%d,",nextTerm);
            t1=t2;
            t2=nextTerm;
            nextTerm=t1+t2;
        }

        printf("\n");
    }
    else if(temp>0){
        printf("(Parent)Prime numbers :");
        while(count<=n){
            flag=0;

            for (j = 2; j <= k / 2; j++)
            {
                if (k % j == 0)
                {
                    flag = 1;
```

```

        break;
    }
}
if (flag == 0)
{
    printf("%d\t", k);
    count++;
}
k++;
}
printf("\n");
}
return 0;
}

```

OUTPUT

```

kali@kali: ~/Desktop/Os-prog
$ cc syscallsia.c
$ ./a.out
Enter size n -----7
(Parent)Prime numbers :2    3    5    7    11    13    17
(Child)Fibonacci Series:0,1,1,2,3,5,8,
$

```

2)

CODE:-

```

#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
void main()

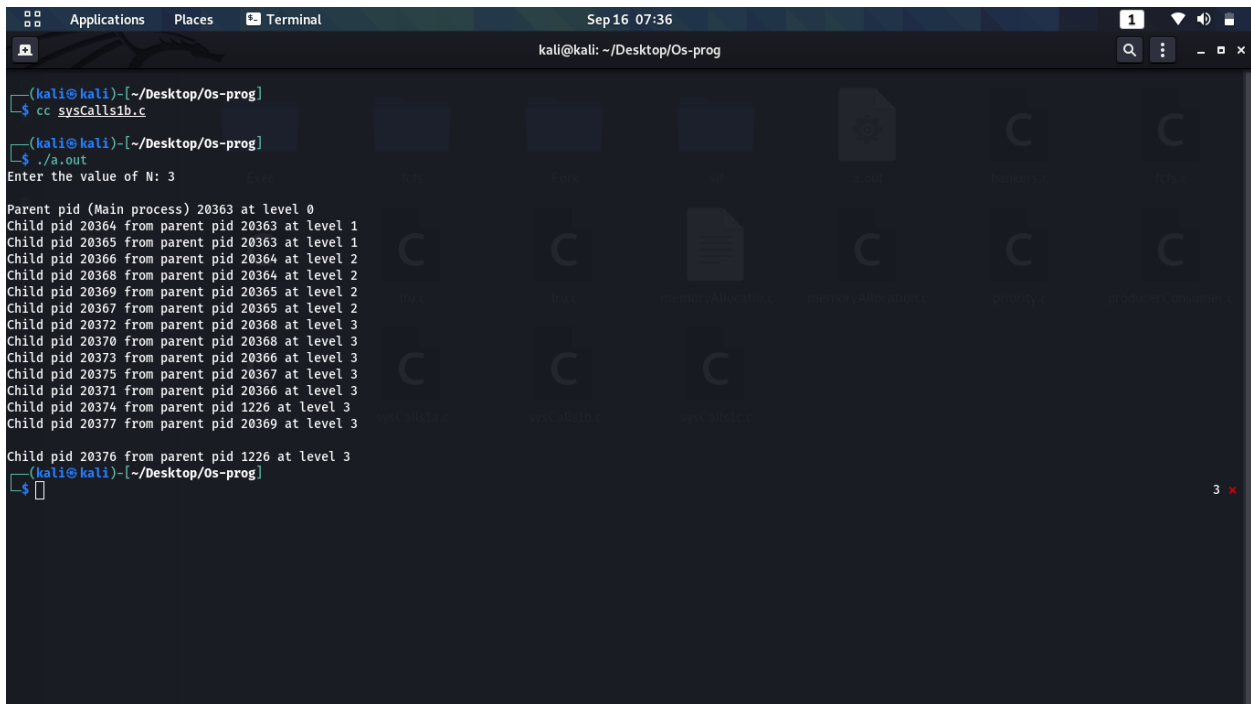
```

```

{
int n;
printf("Enter the value of N: ");
scanf("%d", &n);
printf("\nParent pid (Main process) %d at level 0\n", getpid());
for (int i = 1; i <= n; i++)
if (fork() == 0) // CHILD 1
printf("Child pid %d from parent pid %d at level %d\n",
getpid(), getppid(), i);
else if (fork() == 0) // CHILD 2
printf("Child pid %d from parent pid %d at level %d\n",
getpid(), getppid(), i);
else
{
wait(NULL);
i = n + 1;
}
}
}

```

OUTPUT



```

kali@kali: ~/Desktop/Os-prog
$ cc sysCalls1b.c
$ ./a.out
Enter the value of N: 3

Parent pid (Main process) 20363 at level 0
Child pid 20364 from parent pid 20363 at level 1
Child pid 20365 from parent pid 20363 at level 1
Child pid 20366 from parent pid 20364 at level 2
Child pid 20368 from parent pid 20364 at level 2
Child pid 20369 from parent pid 20365 at level 2
Child pid 20367 from parent pid 20365 at level 2
Child pid 20372 from parent pid 20368 at level 3
Child pid 20370 from parent pid 20368 at level 3
Child pid 20373 from parent pid 20366 at level 3
Child pid 20375 from parent pid 20367 at level 3
Child pid 20371 from parent pid 20366 at level 3
Child pid 20374 from parent pid 1226 at level 3
Child pid 20377 from parent pid 20369 at level 3

Child pid 20376 from parent pid 1226 at level 3
$

```

3)

Input

```

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

```

```

#include <sys/wait.h>
int main()
{
    printf("A: %d\n", getpid());
    if (fork() == 0)
    {
        printf("B: %d forked by %d\n", getpid(), getppid());
        if (fork() == 0)
        {
            printf("D: %d forked by %d\n", getpid(), getppid());
            if (fork() == 0)
            {
                printf("H: %d forked by %d\n", getpid(), getppid());
                if (fork() == 0)
                {
                    printf("I: %d forked by %d\n", getpid(), getppid());
                }
            }
            else
                wait(NULL);
        }
        else
            wait(NULL);
    }
    else if (fork() == 0)
    {
        printf("E: %d forked by %d\n", getpid(), getppid());
    }
    else if (fork() == 0)
    {
        printf("F: %d forked by %d\n", getpid(), getppid());
    }
    else
        wait(NULL);
    else if (fork() == 0)
    {
        printf("C: %d forked by %d\n", getpid(), getppid());
        if (fork() == 0)
        {
            printf("G: %d forked by %d\n", getpid(), getppid());
        }
        else
            wait(NULL);
    }
    else
        wait(NULL);
    return 0;
}

```

OUTPUT

```
Applications  Places  Terminal  Sep 16 07:37  1  [Wi-Fi] [Speaker] [Battery]
kali@kali: ~/Desktop/Os-prog

(kali@kali)-[~/Desktop/Os-prog]
$ cc sysCalls1.c.c
(kali@kali)-[~/Desktop/Os-prog]
$ ./a.out
A: 20431
B: 20432 forked by 20431
C: 20433 forked by 20431
D: 20434 forked by 20432
G: 20435 forked by 20433
F: 20438 forked by 20432
H: 20437 forked by 20434
E: 20436 forked by 20432
I: 20439 forked by 20437
(kali@kali)-[~/Desktop/Os-prog]
$
```