Fork( ) System call:

Fork( ) is a system call which Creates new process.

- Functionality of fork ( ) is to duplicate the process by creating **Parent process** and **Child process.**
- After creation of process both parent process and child process starts execution from the next instruction.
- Fork( ) will return values.
    1. Fork( ) will return a value greater than '0' which can be process id of child process for successful creation of process. Fork( )>0.

    2. Fork( ) will return a Negative value in error case Fork( )<0.

    3. Fork( ) will return '0' to child process. Fork( )=0.

Sample Program:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
fork( );
printf("MNJ_S_19\n");
}
```

Output:

MNJ_S_19

MNJ_S_19

Explanation:

- Functionality of fork is duplicating the process by creating parent and child processes.
- It will duplicate the entire process.
- After creation of process the Parent process will start is execution from the next instruction which is **"printf("MNJ_S_19")"**.
- so the parent process will print **MNJ_S_19** because printf is the next instruction.
- The Child process will start is execution from the next instruction which is **"printf("MNJ_S_19")"**.
- so the Child process will print **MNJ_S_19** because printf is the next instruction.

Parent process

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
fork( );
printf("MNJ_S_19\n");
}
```

Child Process

Assuming child process id is 519

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
fork( );
printf("MNJ_S_19\n");
}
```

- The return value of fork( ) in the parent process is 519 because the parent process always holds the id of child process.
- The return value of fork( ) in the child process is zero.

Program:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
if(fork( )==0)
{
printf("MNJ_S_19 \n");
}
else
{
printf("KAJAL \n");
}
}
```

Output:

KAJAL

MNJ_S_19

Assuming child process id is 519

Parent process

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
if(fork( )==0)
{
printf("MNJ_S_19 \n");
}
else
{
printf("KAJAL \n");
}
}
```

Child Process

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main()
{
if(fork( )==0)
{
printf("MNJ_S_19 \n");
}
else
{
printf("KAJAL \n");
}
}
```

Explanation:

- The return value of fork( ) in the parent process is 519 because the parent process always holds the id of child process.
- It will check the **if condition**
  if(fork( ) ==0) i.e., if(519==0)
- As if condition is false it will print the else case which is **“KAJAL”.**
- The return value of fork( ) in the child process is zero.
- It will check the **if condition**
  if(fork( ) ==0) i.e., if(0==0)
- As if condition is true it will print the if case which is **“MNJ_S_19”.**