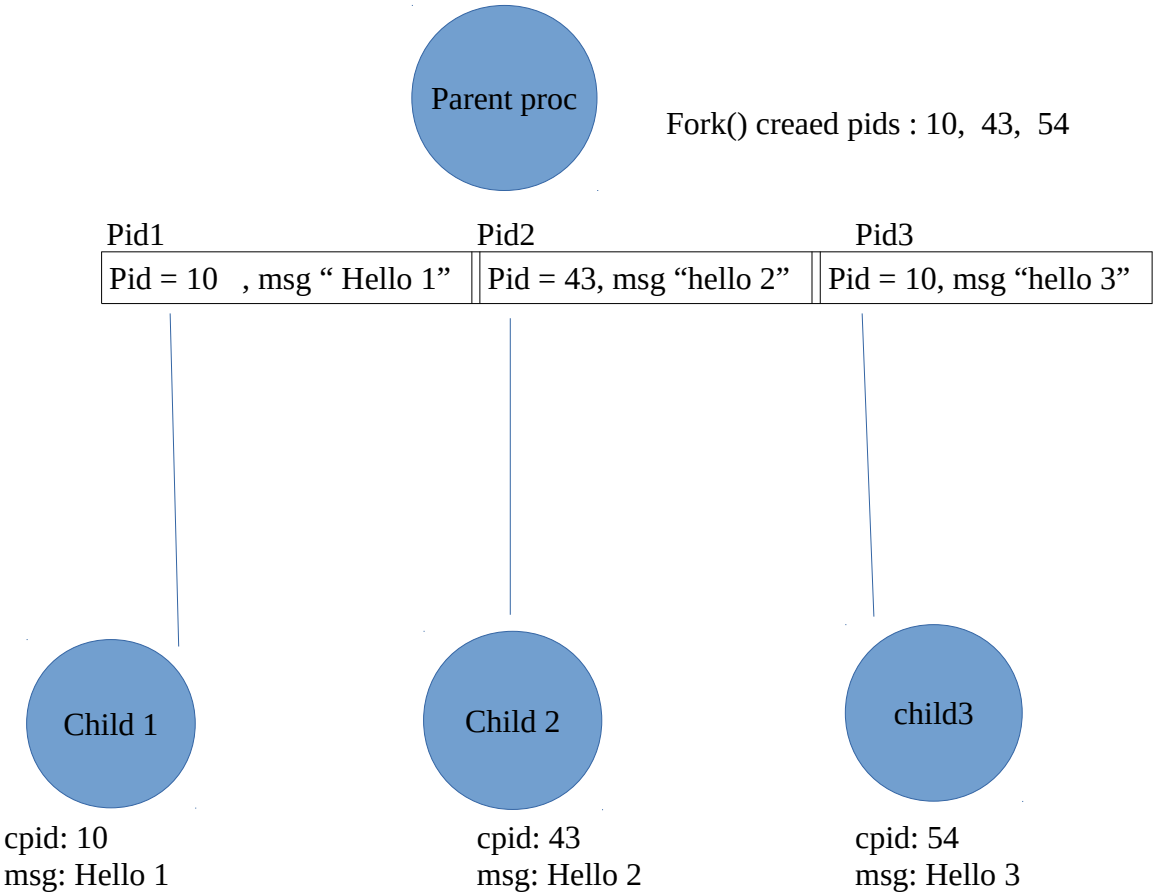


```
struct msgdaga
{
        long mtype;
        char msgbuff[N];
}
```

parent process :

in parent process we create 3 child process and each time we initialize struct with child pid and some text message to perticular child, send msg to message queue.
One that message gets add to message queue, it will be read by such child process which process id is like parent created process id, due to this message forming structure only appropriat chlid can read appropriate message from message queue.

Parent process Application

```
/*
        parent process accept 3 child executable and exec each one after fork
*/
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<sys/wait.h>
```

```
#define MSGSIZE 1024

struct msgdata
{
	long msgtype;
	char msgbuffer[MSGSIZE];
}objs[3];


int main(int argc, char *argv[])
{
	pid_t pids[3];
	key_t key;
	int msgid, status;
	char *msgs[] = {"Hello to child 1","Hello to child 2", "Hello to child 3"};

	key = ftok("/tmp", 0);

	msgid = msgget(key,IPC_CREAT | 0644);
	if(msgid == -1)
	{
		perror("msgget");
		exit(-1);
	}


	for(int i = 0; i < 3; i++)
	{
		 pids[i] = fork();

		 if(pids[i] == 0)
		{
			objs[i].msgtype = getpid();
			strcpy(objs[i].msgbuffer, msgs[i]);
			msgsnd(msgid, &objs[i], sizeof(objs[i]), 0);
			execl(argv[i+1],"",NULL);
		}
	}

	int j =1;
	while (wait(NULL) > 0)
	 {
		  printf("%d child completed\n", j++);
	 }


	return 0;
}
```

child 1 process application

```
/*
        child one application
*/
#include<stdio.h>
#include<stdlib.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<unistd.h>
#include<string.h>


#define MSGSIZE 1024

struct msgdata
{
        long msgtype;
        char msgbuffer[MSGSIZE];
}obj1;


int main()
{


        key_t key;
        int msgid;

        key = ftok("/tmp", 0);

        msgid = msgget(key,IPC_CREAT | 0644);
        if(msgid == -1)
        {
                perror("msgget");
                exit(-1);
        }

        msgrcv(msgid, &obj1, sizeof(obj1), getpid(), IPC_NOWAIT);

        if(strlen(obj1.msgbuffer) == 0)
        {
                printf("there is no msg child 1 in msg q\n");
        }
        else
        {
                printf("data received by child 1 :%s\n",obj1.msgbuffer);
        }

        _exit(0);
}
```

---

child 2 application

```
/*
	child second application
*/
#include<stdio.h>
#include<stdlib.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<unistd.h>
#include<string.h>


#define MSGSIZE 1024

struct msgdata
{
	long msgtype;
	char msgbuffer[MSGSIZE];
}obj1;


int main()
{


	key_t key;
	int msgid;

	key = ftok("/tmp", 0);

	msgid = msgget(key,IPC_CREAT | 0644);
	if(msgid == -1)
	{
		perror("msgget");
		exit(-1);
	}

	msgrcv(msgid, &obj1, sizeof(obj1), getpid(), IPC_NOWAIT);

	if(strlen(obj1.msgbuffer) == 0)
	{
		printf("there is no msg child 2 in msg q\n");
	}
	else
	{
		printf("data received by child 2 :%s\n",obj1.msgbuffer);
	}

	_exit(0);
}
```

child 3 application

```
/*
        child third application
*/
#include<stdio.h>
#include<stdlib.h>
#include<sys/msg.h>
#include<sys/ipc.h>
#include<unistd.h>
#include<string.h>


#define MSGSIZE 1024

struct msgdata
{
        long msgtype;
        char msgbuffer[MSGSIZE];
}obj1;


int main()
{


        key_t key;
        int msgid;

        key = ftok("/tmp", 0);

        msgid = msgget(key,IPC_CREAT | 0644);
        if(msgid == -1)
        {
                perror("msgget");
                exit(-1);
        }

        msgrcv(msgid, &obj1, sizeof(obj1), getpid(), IPC_NOWAIT);

        if(strlen(obj1.msgbuffer) == 0)
        {
                printf("there is no msg child 3 in msg q\n");
        }
        else
        {
                printf("data received by child 3 :%s\n",obj1.msgbuffer);
        }

        _exit(0);
}
```