1. Fork() command

```c
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<sys/wait.h>
#include<fcntl.h>
int main()
{
fork();
printf("Child and Parent Process \n");
return 0;
}
```

Output :

```
root@LAPTOP-92POMHL1:~# gcc code.c
root@LAPTOP-92POMHL1:~# ./a.out
Child and parent process
Child and parent process
root@LAPTOP-92POMHL1:~#
```

2. For multiple fork() commands

```
 GNU nano 6.2                                  terminal_commands.c *
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <fcntl.h>
int main() {
    fork();
fork();
fork();
printf("Child and parent process\n");
    return 0;
}
```

Output :

```
root@LAPTOP-92POMHL1:~# nano fork.c
root@LAPTOP-92POMHL1:~# gcc fork.c
root@LAPTOP-92POMHL1:~# ./a.out
child and parents process
child and parents process
child and parents process
child and parents process
child and parents process
child and parents process
root@LAPTOP-92POMHL1:~# child and parents process
child and parents process
```

3.Printing Parent and child process id through keeping else block on wait()

```
 GNU nano 6.2                                  terminal_commands.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <fcntl.h>
int main() {
    pid_t q;
q=fork();
if(q==-1){
printf("error");}
if(q=0){
printf("child processid=%d\n",getpid());
printf("parent processid=%d\n",getpid());}
else {wait(NULL);
printf("parent id=%d\n",getpid());
printf("Child id=%d\n",q);}

}
```

Output :

```
root@LAPTOP-92POMHL1:~# ./a.out
parent id=6992
child id=0
parent id=6991
child id=0
root@LAPTOP-92POMHL1:~#
```

4.Printing parent and child process id through keeping if block on sleep()

```
 GNU nano 6.2                                  terminal_commands.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <fcntl.h>
int main() {
    pid_t q;
q=fork();
if(q==-1){
printf("error");}
if(q=0){
sleep(10);
printf("child processid=%d\n",getpid());
printf("parent processid=%d\n",getpid());}
else {
printf("parent id=%d\n",getpid());
printf("Child id=%d\n",q);}

printf("similar ids");
}
```

Output :

```
root@LAPTOP-92POMHL1:~# nano fork1.c
root@LAPTOP-92POMHL1:~# gcc fork1.c
root@LAPTOP-92POMHL1:~# ./a.out
parent id=7685
child id=0
similar idsparent id=7686
child id=0
similar idsroot@LAPTOP-92POMHL1:~#
```