```c
// Creacion por copia exacta
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    /* code */
    int i = 10;
    int id_proc;
    id_proc = fork();
    if(id_proc == 0)
    {
        i += 10;
        printf("SOY EL PROCESO HIJO %i\n", i);
        exit(0);
    }
    else
    {
        i += 100;
        printf("SOY EL PROCESO PADRE %i\n", i);
        exit(0);
    }
}
```

```c
// Creacion por copia exacta
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int *i = 10;
    int id_proc;
    id_proc = fork();
    if(id_proc == 0)
    {
        *i = *i + 10;
        printf("SOY EL PROCESO HIJO %i\n", *i);
        exit(0);
    }
    else
    {
        *i = *i + 100;
        printf("SOY EL PROCESO PADRE %i\n", *i);
        exit(0);
    }
}
```

```c
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
int i = 10;
int main(void)
{
    int id_proc = fork();
    if(id_proc == 0)
    {
        i += 10;
        printf("Soy el proceso hijo %i\n", i);
    }
    else
    {
        i += 100;
        printf("Soy el proceso padre %i\n", i);
    }
    i += 20;
    printf("Valor de i= %i\n", i);
    // Delimita hasta donde el proceso padre o hijo ejecutan
    exit(0);
}
```

```c
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
int i = 10;
int main(void)
{
    int id_proc = fork();
    if(id_proc == 0)
    {
        i += 10;
        printf("Soy el proceso hijo %i\n", i);
        // Para generar un proceso del hijo
    }
    else
    {
        i += 100;
        printf("Soy el proceso padre %i\n", i);
        // Para generar un proceso del hijo Padre
    }
    i += 20;
    printf("Valor de i= %i\n", i);
    // Delimita hasta donde el proceso padre o hijo ejecutan
    exit(0);
}
```

```c
// 04 de Octubre 2018
// Para generar muchos procesos HIJO
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
int i = 10;
int main(void)
{
    int id_proc;
    for(i = 0; i<100; i++)
    {
        id_proc = fork();
        if(id_proc == 0)
        {
            printf("Soy el proceso hijo %i\n", i);
        } break; //exit(0); // Para que no cree hijos de hijos
        else
        {
            // Para que el padre haga otra cosa, sino se omite
            printf("Soy el proceso padre %i\n", i);
        }
    }
}
```


```c
// 04 de Octubre 2018
// Para generar muchos procesos HIJO
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
int i = 10;
int main(void)
{
    int id_proc;
    for(i = 0; i<100; i++)
    {
        id_proc = fork();
        if(id_proc == 0)
        {
            printf("Soy el proceso hijo %i\n", i);
        } break; //exit(0); // Para que no cree hijos de hijos
        else
        {
            // Para que el padre haga otra cosa, sino se omite
            printf("Soy el proceso padre %i\n", i);
        }
    }
}
```

```c
// Para generar muchos procesos seguidos
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
int i = 10;
int main(void)
{
    /* Mantiene vivos los procesos hijos que se
    van convirtiendo en padres si usamos exit vamos
    matando los procesos padre y eso no queremos. */
    int id_proc;
    for(i = 0; i<100; i++)
    {
        id_proc = fork();
        // Cuando es hijo el fork siempre devuelve 0.
        if(id_proc == 0)
            printf("Soy el proceso hijo %i\n", i);
        else
        {
            break; // Sale del ciclo
            // Para que el padre haga otra cosa, sino se omite
            printf("Soy el proceso padre %i\n", i);
        }
    }
}
```