

```
1 // Creacion por copia exacta
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int main(void)
7 {
8     /* code */
9     int i = 10;
10    int id_proc;
11    id_proc = fork();
12    if(id_proc == 0)
13    {
14        i += 10;
15        printf("SOY EL PROCESO HIJO %i\n", i);
16        exit(0);
17    }
18    else
19    {
20        i += 100;
21        printf("SOY EL PROCESO PADRE %i\n", i);
22        exit(0);
23    }
24 }
```

```
1 // Creacion por copia exacta
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <stdlib.h>
5
6 int main(void)
7 {
8     int *i = 10;
9     int id_proc;
10    id_proc = fork();
11    if(id_proc == 0)
12    {
13        *i = *i + 10;
14        printf("SOY EL PROCESO HIJO %i\n", *i);
15        exit(0);
16    }
17    else
18    {
19        *i = *i + 100;
20        printf("SOY EL PROCESO PADRE %i\n", *i);
21        exit(0);
22    }
23 }
```

```

1  #include<unistd.h>
2  #include<stdio.h>
3  #include<stdlib.h>
4  int i = 10;
5  int main(void)
6  {
7      int id_proc = fork();
8      if(id_proc == 0)
9      {
10         i += 10;
11         printf("Soy el proceso hijo %i\n", i);
12     }
13     else
14     {
15         i += 100;
16         printf("Soy el proceso padre %i\n", i);
17     }
18     i += 20;
19     printf("Valor de i= %i\n", i);
20     // Delimita hasta donde el proceso padre o hijo ejecutan
21     exit(0);
22 }

```

```

1  #include<unistd.h>
2  #include<stdio.h>
3  #include<stdlib.h>
4  int i = 10;
5  int main(void)
6  {
7      int id_proc = fork();
8      if(id_proc == 0)
9      {
10         i += 10;
11         printf("Soy el proceso hijo %i\n", i);
12         // Para generar un proceso del hijo
13     }
14     else
15     {
16         i += 100;
17         printf("Soy el proceso padre %i\n", i);
18         // Para generar un proceso del hijo Padre
19     }
20     i += 20;
21     printf("Valor de i= %i\n", i);
22     // Delimita hasta donde el proceso padre o hijo ejecutan
23     exit(0);
24 }

```

```

1 // 04 de Octubre 2018
2 // Para generar muchos procesos HIJO
3 #include<unistd.h>
4 #include<stdio.h>
5 #include<stdlib.h>
6 int i = 10;
7 int main(void)
8 {
9     int id_proc;
10    for(i = 0; i<100; i++)
11    {
12        id_proc = fork();
13        if(id_proc == 0)
14        {
15            printf("Soy el proceso hijo %i\n", i);
16        } break; //exit(0); // Para que no cree hijos de hijos
17        else
18        {
19            // Para que el padre haga otra cosa, sino se omite
20            printf("Soy el proceso padre %i\n", i);
21        }
22    }
23 }

```

```

1 // 04 de Octubre 2018
2 // Para generar muchos procesos HIJO
3 #include<unistd.h>
4 #include<stdio.h>
5 #include<stdlib.h>
6 int i = 10;
7 int main(void)
8 {
9     int id_proc;
10    for(i = 0; i<100; i++)
11    {
12        id_proc = fork();
13        if(id_proc == 0)
14        {
15            printf("Soy el proceso hijo %i\n", i);
16        } break; //exit(0); // Para que no cree hijos de hijos
17        else
18        {
19            // Para que el padre haga otra cosa, sino se omite
20            printf("Soy el proceso padre %i\n", i);
21        }
22    }
23 }

```

```
1 // Para generar muchos procesos seguidos
2 #include<unistd.h>
3 #include<stdio.h>
4 #include<stdlib.h>
5 int i = 10;
6 int main(void)
7 {
8     /* Mantiene vivos los procesos hijos que se
9     van convirtiendo en padres si usamos exit vamos
10    matando los procesos padre y eso no queremos. */
11    int id_proc;
12    for(i = 0; i<100; i++)
13    {
14        id_proc = fork();
15        // Cuando es hijo el fork siempre devuelve 0.
16        if(id_proc == 0)
17            printf("Soy el proceso hijo %i\n", i);
18        else
19        {
20            break; // Sale del ciclo
21            // Para que el padre haga otra cosa, sino se omite
22            printf("Soy el proceso padre %i\n", i);
23        }
24    }
25 }
```