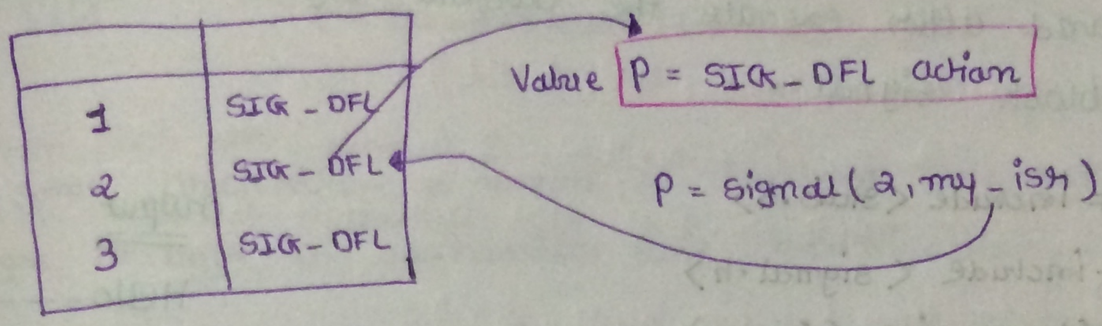
(*) disadvantages & advantages of signal

prog-1 write a program to find the current action of the given signal, given the signal number.

Note :- signal function return the previous default action.




```
#include <stdio.h>
#include <signal.h>
main()
{
   Void (*P)(int);
   int num;
       printf("Enter the num---\n");
       scanf("%d", &num);
   Signal (2, SIG_IGN) ;
    P= Signal (num, SIG_IGN);
    Signal (num, P) ;
 if ( P== SIG_DFL)
     Printf (" Default ----\n") ;
 else if ( P== SIG_IGN)
     Printf (" Ignore ----\n") ;
 else
  Printf (" my_isr ----\n") ;
}
```

output :-

Enter the num = 2
Ignore ---
Enter the num = 3
Default ----

① Without setting the new action it is not possible to find out what is the current action is parent in the signal table corresponding to the signal value.

② disadvantage - 2

⇒ because of 1 signal if isr is executing before completion of that if another signal came than another signal is served. means it is not possible to block other signal because of one signal if ISR is executing.

⇒ if same signal is came than block the new signal and after execute the current signal it execute the block signal.

```
# include <stdio.h>
# include <signal.h>
Void my_isr (int n)
{
   Printf (" In isr ---\n");
   Sleep (10);
   printf (" In isr after sleep ----\n");
}
main()
{
    Signal (2, my_isr);
    Printf (" Hello ---\n");
    while (1);
}
```

Output

```
Hello ----
^C In isr ----
(between 10 sec)
^C
(after 10 second)
 In isr ---
 In isr after sleep.
```

To overcome this disadvantages to use SIGACTION Function.

$ man Sigaction :-

Sigaction → examine and change a signal action.

```
#include <signal.h>
int sigaction (int signum , const struct sigaction *act ,
                    struct sigaction *oldact);
```

struct sig
{
Void
Void
sigse
int
Void
};

⇒ on some
assign

⇒ The Sa
be used

POSI

⇒ Some S
Created
any on

```
#include
#include
Void my
{
    Printf
}
main()
{
    Str
    // S
    V.
    Si
    V
    S
    P
```

```
struct sigaction
{
    Void  (*sa-handler) (int) ;
    Void  (*sa-sigaction) (int, siginfo-t *, void *) ;
    sigset-t        sa_mask ;
    int             sa-flags ;
    Void            (*sa-restores) (Void) ;
} ;
```

→ on some architecture a union is involved : do not assign to both a sa-handler and sa-sigaction.

→ The sa restores element is obsolute and should not be used.

POSIX :- Portable Operating System Interface

→ Some architecture, inside this structure a union is created which involves 1st and 2nd we can consider any one. we consider 1st.

```
#include <stdio.h>
#include <signal.h>
Void my-isr (int n)
{
   Printf (" In isr ---\n");
}
main()
{
    Struct sigaction  V ;
    // Signal (2, my-isr) ;
    V.sa-handler  = my-isr ;
    sigemptyset ( &V.sa-mask) ;
    V.sa_flags = 0 ;
    Sigaction (2, &V, 0) ;
    Printf (" Hello ---\n"); while(1) ;
}
```

Output

```
Hello----
^C In isr ----
^C In isr ----
^C In isr ----
```

✱ 3 Ways we can use sigaction :-

① sigaction (num, &V, NULL); → New action

② sigaction (num, NULL, &V); → old action

③ sigaction (num, &V, &V);

① sigaction (num, &V, NULL);

⇒ In 1st method Interested to set new method but not collecting a old method.

② sigaction (num, NULL, &V);

⇒ we are not setting the new method but collecting the old action (current action).

③ sigaction (num, &V, &V);

⇒ we are setting a new action as well as you collecting the old action.

```
#include <stdio.h>
#include <signal.h>
main()
{
        Struct sigaction V;
        int num;
            Printf(" Enter the num---\n");
            Scanf("%d", &num);
       Signal (2, SIG-IGN);
     Sigaction (num, NULL, &V);
if ( V.sa-handler == SIG-DFL)
      Printf (" Default----\n");
else if ( V.sa-handler == SIG-IGN)
        Printf (" Ignore----\n");
```

```
else
  printf (" my-isr ----\n");
}
```

Sigaction for the SIGCHLD signal.

```
#include <stdio.h>
#include <signal.h>
Void my-isr (int n)
{
  printf (" In isr----\n");
  wait(0);
}
main()
{
    if (fork()==0)
    {
       sleep(5);
    }
    else
    {
       struct sigaction V;
       V.sa_handler = my_isr;
       sigemptyset (&V.sa-mask);
       V.sa_flags =0;
       Sigaction (17, &V, NULL);
       while(1);
    }
}
```

→ Sa flags specific a set of flags which modify the behaviours of the signal. It is formed by the bitwise or ~~not~~ of zero or more of the following.

① SA_NOCLDSTOP :-

do not receive notification when child process is stop or resume. The parent will receive the SIGCHLD signal when

1) Child terminating.
2) Child process suspended.
3) Child process resumed.

```
# include <stdio.h>
# include <signal.h>
Void myisr (int n)
{
    printf ("In isr ----\n");
}
main()
{
    if (fork()==0)
    {
        printf ("In child Pid = %d \n", getpid());
        while(1);
    }
    else
    {
        Struct sigaction V;
        V.sa_handler = my_isr;
        Sigemptyset (&V.sa_mask);
      // V.sa_flags = 0
        V.sa_flags = SA_NOCLDSTOP;
        Sigaction (17, &V, NULL);
        while(1);
    }
}
```

② SA_NOCLDWAIT :-

→ If signum is SIGCHLD.
→ do not transform child into zombies when they terminate.

```
#include <stdio.h>
#include <signal.h>
Void my_isr (int n)
{
   Printf (" In isr ---\n");
}
main()
{
   if (fork() == 0)
   {
      Printf (" In child Pid = %d\n", getpid());
      while(1);
   }
   else
   {
      struct Sigaction V;
      V.sa_handler = my_isr;
      Sigemptyset (&v.sa_mask);
      V.sa_flags = SA_NOCLDWAIT;
      Sigaction (17, &V, NULL);
      while(1);
   }
}
```

③ SA_NODEFER :-

→ Do not prevent the signal from being received from within its own signal handler.

→ This is flag is only meaningful when establishing a signal handler - SA_NOMASK.

Program :-

```c
#include <stdio.h>
#include <signal.h>
Void my_ish (int n)
{
        Printf (" In ish ----\n ");
        Sleep (10);
        Printf (" In ish after Sleep---\n");
}
main()
{
        struct Sigaction V;
        V.sa_handler = my_ish;
        sigemptyset ( &V.sa_mask);
        V.sa_flags = SA_NODEFER;
        Sigaction (2, &V, 0);
        while(1);
}
```

(4) SA_RESETHAND :-

⇒ Restore the signal action to the default state once the signal handler has been called.

```c
#include <stdio.h>
#include <signal.h>
Void my_ish ( int n)
{
    Printf (" In ish ----\n");
}
main()
{
        struct Sigaction V;
        V.sa_handler = my_ish;
        sigemptyset ( &V.sa_mask);
        V.sa_flags = SA_RESETHAND;
```

```c
Sigaction
while(1)
}
```

Sa

⇒ Sa-ma

be block d

```c
#include <s
# include <
Void my_is
{
    Printf
    Sleep(1
    Printf
}
main()
{
    Struct
    V.sa
    Sig fl
    V.sa
    Siga
    whil
}
```

⇒ Sigen

Sig f

$

1) int