P1.c

1 //change the signal no 2 action into isr using sigaction function

2 #include<stdio.h>

3 #include<signal.h>

4 void isr(int n)

5 {

6 printf("in isr...\n");

7 }

8 main()

9 {

10 printf("process under execution...\n");

11 struct sigaction v;

12 v.sa\_handler=isr;

13 sigemptyset(&v.sa\_mask);

14 v.sa\_flags=0;

15 sigaction(2,&v,0);

16 //signal(2,isr);

17 while(1);

18 }

P2.c

1 //wap to find the current action of a signal using sigaction?

2 #include<stdio.h>

3 #include<signal.h>

4 main()

5 {

6 int n;

7 struct sigaction v;

8 signal(2,SIG\_IGN);

9 printf("enter signal number...\n");

10 scanf("%d",&n);

11

12 sigaction(n,0,&v);

13

14 if(v.sa\_handler==SIG\_DFL)

15 printf("default...\n");

16 else if(v.sa\_handler==SIG\_IGN)

17 printf("ignore..\n");

18 else

19 printf("isr...\n");

20 }

21 /\*to find the current action of a signal using signal function first required to set a new dummy action and collect the retun value of signal function,but using sigaction function with the third argument will ge current action of a signal \*/

P3.c

1 /\*SA\_RESETHAND

2 Restore the signal action to the default state once the sig‐

3 nal handler has been called. This flag is only meaningful

4 when establishing a signal handler.\*/

5

6 #include<stdio.h>

7 #include<signal.h>

8 void isr(int n)

9 {

10 printf("in isr...\n");

11 }

12 main()

13 {

14 printf("process is under execution...\n");

15 struct sigaction v;

16 v.sa\_handler=isr;

17 sigemptyset(&v.sa\_mask);

18 v.sa\_flags=SA\_RESETHAND;

19 sigaction(2,&v,0);

20 while(1);

21 }

P4.c

1 /\*

2 any state change in child process(that is child termination/child suspend/child resume)parent process will get notification signal that is SIGCHLD(17) \*/

3 #include<stdio.h>

4 #include<signal.h>

5 void isr(int n)

6 {

7 printf("in isr...\n");

8 }

9 main()

10 {

11

12 if(fork()==0)

13 {

14 printf("child pid:%d\n",getpid());

15 sleep(20);

16 printf("child terminated...\n");

17 exit(0);

18 }

19 else

20 {

21 struct sigaction v;

22 v.sa\_handler=isr;

23 sigemptyset(&v.sa\_mask);

24 v.sa\_flags=0;

25 sigaction(17,&v,0);

26 while(1);

27 }

28

29 }

P5.c

1 /\*

2 any state change in child process(that is child termination/child suspend/child resume)parent process will get notification signal that is SIGCHLD(17)\*/

3

4

5 /\* SA\_NOCLDSTOP

6 If signum is SIGCHLD, do not receive notification when child

7 processes stop (i.e., when they receive one of SIGSTOP,

8 SIGTSTP, SIGTTIN or SIGTTOU) or resume\*/

9 #include<stdio.h>

10 #include<signal.h>

11 void isr(int n)

12 {

13 //wait(0);

14 printf("in isr...\n");

15 }

16 main()

17 {

18

19 if(fork()==0)

20 {

21 printf("child pid:%d\n",getpid());

22 sleep(20);

23 printf("child terminated...\n");

24 exit(0);

25 }

26 else

27 {

28 struct sigaction v;

29 v.sa\_handler=isr;

30 sigemptyset(&v.sa\_mask);

31 v.sa\_flags=SA\_NOCLDSTOP;

32 sigaction(17,&v,0);

33 while(1);

34 }

35

36 }

P6.c

1 /\*with out writing wait (or) wait pid in parent process how to collect the child status so that child not turned as zombiee\*/

2 //ans:using SA\_NOCLDWAIT

3

4 #include<stdio.h>

5 #include<signal.h>

6 void isr(int n)

7 {

8 //wait(0);

9 printf("in isr...\n");

10 }

11 main()

12 {

13

14 if(fork()==0)

15 {

16 printf("child pid:%d\n",getpid());

17 sleep(20);

18 printf("child terminated...\n");

19 exit(0);

20 }

21 else

22 {

23 struct sigaction v;

24 v.sa\_handler=isr;

25 sigemptyset(&v.sa\_mask);

26 v.sa\_flags=SA\_NOCLDSTOP|SA\_NOCLDWAIT;

27 sigaction(17,&v,0);

28 while(1);

29 }

30

31 }

P7.c

1 /\*SA\_NODEFER:Do not prevent the signal from being received from within

2 its own signal handler.\*/

3

4

5 #include<signal.h>

6 void isr(int n)

7 {

8 unsigned int i;

9 printf("in isr...\n");

10 for(i=0;i<4000000000;i++);

11 printf("isr completed..\n");

12 }

13 main()

14 {

15 printf("process under execution...\n");

16 struct sigaction v;

17 v.sa\_handler=isr;

18 sigemptyset(&v.sa\_mask);

19 v.sa\_flags=SA\_NODEFER;

20 sigaction(2,&v,0);

21 while(1);

22 }