P1.c

1 //example for signal generation to process

2 #include<stdio.h>

3 main()

4 {

5 printf("hello...\n");

6 int \*p=0;

7 printf("%d\n",\*p);

8 }

P2.c

1 //example for signal generation for process(ctrl+c)

2 #include<stdio.h>

3 main()

4 {

5 while(1)

6 printf("hello..\n");

7 }

P3.c

1 //example for signal generation to process

2 #include<stdio.h>

3 main()

4 {

5 int a=10,b=0;

6 printf("hello...\n");

7 int res;

8 res=a/b;

9 printf("%d\n",res);

10 }

P4-a

1 #include<stdio.h>

2 main()

3 {

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

5 while(1);

6 }

P4-b

1 /\*kill - send signal to a process

2 int kill(pid\_t pid, int sig);\*/

3 #include<stdio.h>

4 main(int argc,char \*argv[])

5 {

6 int ret;

7 if(argc!=3)

8 {

9 printf("./a.out signalno pid..\n");

10 return;

11 }

12

13 if(ret=kill(atoi(argv[2]),atoi(argv[1]))<0)

14 perror("kill");

15

16 }

P5.c

1 /\*raise - send a signal to the caller

2 int raise(int sig);\*/

3 #include<stdio.h>

4 #include<signal.h>

5 main()

6 {

7 printf("hello...\n");

8 raise(11);

9 printf("by...\n");

10 while(1);

11 }

P6.c

1 /\*pause - wait for signal

2 int pause(void);

3 DESCRIPTION:pause() causes the calling process (or thread) to sleep until a signal is delivered t hat either terminates the process or causes the invocation of a signal-catching function.\*/

4 #include<stdio.h>

5 #include<signal.h>

6 main()

7 {

8 int ret;

9 printf("hello...\n");

10 ret=pause(); //process entered into sleep unti a signal is received

11 printf("ret:%d\n",ret);

12 printf("by..\n");

13 }

14 /\*pause() only returns when a signal was caught and the signal-catching

15 function returned. In this case pause() returns -1\*/