P1.c

1 #include<signal.h>

2 void isr(int n)

3 {

4 unsigned int i;

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

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

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

8 }

9 main()

10 {

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

12 struct sigaction v;

13 v.sa\_handler=isr;

14 /\*sigemptyset(&v.sa\_mask);//when signal no 2 isr is executing (during isr esecution) same signal

15 not allowed but all other signals are allowed\*/

16 sigfillset(&v.sa\_mask);

17 /\*when signal no 2 isr is executing (during isr execution)same signal not allowed and all other signals also not allowe d)\*/

18 v.sa\_flags=0;

19 sigaction(2,&v,0);

20 while(1);

21 }

P2.c

1 /\*int getrlimit(int resource, struct rlimit \*rlim);

2 int setrlimit(int resource, const struct rlimit \*rlim);

3 The getrlimit() and setrlimit() system calls get and set resource lim‐

4 its respectively. Each resource has an associated soft and hard limit,

5 as defined by the rlimit structure:

6

7 struct rlimit {

8 rlim\_t rlim\_cur; // Soft limit

9 rlim\_t rlim\_max; //Hard limit (ceiling for rlim\_cur)

10 };\*/

11 //RLIMIT\_STACK

12 // The maximum size of the process stack, in bytes

13

14 #include<stdio.h>

15 #include<sys/resource.h>

16 main()

17 {

18 struct rlimit v;

19 printf("process executing...\n");

20 printf("pid:%d ppid:%d\n",getpid(),getppid());

21

22 getrlimit(RLIMIT\_STACK,&v);

23

24 printf("soft:%u\n",v.rlim\_cur);

25 printf("hard:%u\n",v.rlim\_max);

26

27 v.rlim\_cur=10000;

28 setrlimit(RLIMIT\_STACK,&v);

29

30 printf("after setrlimit...\n");

31

32 getrlimit(RLIMIT\_STACK,&v);

33 printf("soft:%u\n",v.rlim\_cur);

34 printf("hard:%u\n",v.rlim\_max);

35

36 }