P1.c

/\*RLIMIT\_CORE

2 Maximum size of core file. When 0 no core dump files are created.

3 When nonzero, larger dumps are truncated to this size.\*/

4 #include<stdio.h>

5 #include<sys/resource.h>

6 main()

7 {

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

9 struct rlimit v;

10 getrlimit(RLIMIT\_CORE,&v);

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

12 printf("soft:%u\n",v.rlim\_max);

13 v.rlim\_cur=1000;

14 setrlimit(RLIMIT\_CORE,&v);

15 printf("after soft limit value change..\n");

16 getrlimit(RLIMIT\_CORE,&v);

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

18 printf("soft:%u\n",v.rlim\_max);

19

20 while(1);

21 }

22 /\* The default action of certain signals is to cause a process to terminate

23 and produce a core dump file, a disk file containing an image of the

24 process's memory at the time of termination. This image can be used in a

25 debugger (e.g., gdb(1)) to inspect the state of the program at the time

26 that it terminated.\*/

P2.c

1 /\* RLIMIT\_FSIZE

2 The maximum size of files that the process may create. Attempts to

3 extend a file beyond this limit result in delivery of a SIGXFSZ

4 signal. By default, this signal terminates a process\*/

5

6 #include<stdio.h>

7 #include<sys/resource.h>

8 main()

9 {

10 struct rlimit v;

11 char a[20]="abcdefghijklmnop";

12 FILE \*fp;

13 getrlimit(RLIMIT\_FSIZE,&v);

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

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

16 v.rlim\_cur=10;

17 setrlimit(RLIMIT\_FSIZE,&v);

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

19 getrlimit(RLIMIT\_FSIZE,&v);

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

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

22 fp=fopen("temp","w");

23 if(fp==NULL)

24 {

25 perror("fopen");

26 return;

27 }

28 fprintf(fp,"%s",a);

29 }

P3.c

1 /\*RLIMIT\_CPU

2 CPU time limit in seconds. When the process reaches the soft

3 limit, it is sent a SIGXCPU signal. The default action for this

4 signal is to terminate the process. \*/

5 #include<stdio.h>

6 #include<sys/resource.h>

7 main()

8 {

9 printf("hello my pid:%d\n",getpid());

10 struct rlimit v;

11 getrlimit(RLIMIT\_CPU,&v);

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

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

14 v.rlim\_cur=20;

15 setrlimit(RLIMIT\_CPU,&v);

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

17 getrlimit(RLIMIT\_CPU,&v);

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

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

20 while(1);

21 }

P4.c

1 /\*time - get time in seconds

2 time\_t time(time\_t \*t);

3 DESCRIPTION

4 time() returns the time as the number of seconds since the Epoch,

5 1970-01-01 \*/

6

7 #include<stdio.h>

8 #include<time.h>

9 main()

10 {

11 time\_t t;

12 while(1)

13 {

14 time(&t);

15 printf("%u\n",t);

16 printf("%s\n",ctime(&t));

17 sleep(1);

18 system("clear");

19 }

20 }