Lesson 5

Nice:

nice of a process determines the scheduling priority of a process. The *nice* is in the ranges [-20, 20]. The higher the nice, the lower scheduling priority of the process (a process with nice of 20 has the lowest priority).
The nice can only be increased (i.e., to lower the priority of the process). Only the super-user can decrease it.
The default of a process is to begin with nice of 0.

SYNOPSIS (shell command)
nice [-*number*] *command* [*arguments*]

SYNOPSIS (process - program)
nice(int num) // increase (decrease) the nice of the running program in *num*.

Example:
```
> /usr/bin/nice -7 ps -l        // increasing the nice for the process "ps -l" in 7.
```

renice: alter priority of running processes

getpriority, setpriority: get/set program scheduling priority.

SYNOPSIS
```
#include <sys/types.h>
#include <time.h>
#include <sys/resource.h>

int getpriority(int which, int who);
int setpriority(int which, int who, int prio);
```

DESCRIPTION
The scheduling priority of the process, process group, or user, as indicated by *which* and *who* is obtained with the *getpriority()* call and set with the *setpriority()* call.
which is one of: PRIO_PROCESS, PRIO_PGRP, or PRIO_USER
who is interpreted relative to *which* (a process identifier for PRIO_PROCESS, process group identifier for PRIO_PGRP, and a user ID for PRIO_USER).
A zero value of *who* denotes the current process, process group, or user.
The *getpriority()* call returns the highest priority (lowest numerical value) enjoyed by any of the specified processes.

The *setpriority()* call sets the priorities of all of the specified processes to the specified value. Only the super-user may lower priorities.

RETURN VALUES
Since *getpriority()* can legitimately return the value -1, it is necessary to clear the external variable *errno* prior to the call, then check it afterward to determine if a -1 is an error or a legitimate value. The *setpriority()* call returns 0 if there is no error, or -1 if there is.

Examples:
(nice5_1.c)
```
main()
{
        nice(7);
        while(1);
}
1 > gcc -o nice1 nice5_1.c
```

```
2 > nice1 &
[1] 1730
3 > ps -l
UID  PID   PPID   PRI NI STAT TT        TIME COMMAND
8385 1730  21560  0   27 O    pts/0     0:03 nice1
8385 21560 21547  48  20 S    pts/0     0:02 -tcsh
```

(nice5_2.c)

```
#include <sys/resource.h>

main()
{
	int prior;

	prior = getpriority(PRIO_PROCESS, getpid()); // get the nice of the current process
	printf("%d\n", prior);

	nice(7);

	prior = getpriority(PRIO_PROCESS, getpid());
	printf("%d\n", prior);
}
1 > gcc -o nice2 nice5_2.c
2 > nice2
0
7
3 > /usr/bin/nice -14 nice2
14
19
```

(nice5_3.c)

```
#include <sys/resource.h>

main()
{
	int prior;

	setpriority(PRIO_PROCESS, getpid(), 7);
	prior = getpriority(PRIO_PROCESS, getpid());
	printf("%d\n", prior);
}
1 > gcc -o nice3 nice5_3.c
7
```

(nice5_4.c)

```
#include <sys/resource.h>

main()
{
	setpriority(PRIO_USER, getuid(), 1); // set the nice of the shell to 1
}
2 > gcc -o nice4 nice5_4.c
3 > nice4
```

(nice5_5.c)

```
#include <sys/resource.h>

main()
{
	int prior;
```

```
            prior = getpriority(PRIO_PROCESS, getpid());
            printf("%d\n", prior);
}
4 > gcc -o nice5 nice5_5.c
5 > nice5
1
6 > renice -13 1730          // for nice1 above
renice: 1730: setpriority: Permission denied

7 > renice 13 1730

8 > ps -l
UID  PID   PPID   PRI NI STAT TT        TIME COMMAND
8385 1730  21560  0   33 R    pts/0     0:03 nice1
8385 21560 21547  45  21 S    pts/0     0:02 -tcsh

9 > /user/bin/nice -7 ps -l
UID  PID   PPID   PRI NI STAT TT        TIME COMMAND
8385 1730  21560  0   33 R    pts/0     0:03 nice1
8385 21560 21547  45  21 S    pts/0     0:02 -tcsh
8385 21561 21547  45  28 R    pts/0     0:00 ps -l

10 > kill -KILL 1730
[1]     Killed   nice1
```