```
/* msg_run4.c - Use System V messages */

#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

typedef struct mymsgbuf {
                  long int mtype;
                  char text[sizeof(int)];
                  } MYMSGBUF;

int sid;
struct msqid_ds buff;

void compute_sqr_msg(long int mesgq)
{
 MYMSGBUF msg = {1, ""};
 int temp;

    msg.mtype = 1;
    msgrcv(mesgq, (struct msgbuf *)&msg, sizeof(int), 1L , MSG_NOERROR);
    memcpy(&temp, msg.text, sizeof(int));
    temp = temp * temp;
    memcpy(msg.text, &temp,sizeof(int));
    msg.mtype = 2;
    msgsnd(mesgq, (struct msgbuf *)&msg,
     sizeof(MYMSGBUF) - sizeof(long), IPC_NOWAIT);


}// compute_sqr_msg


void request_sqr_msg(long int mesgq)
{

    MYMSGBUF msg1 = {1, ""};
    int temp, temp1;

      puts("Enter number:");
      scanf("%d", &temp1);
      fflush(stdin);
      memcpy(msg1.text, &temp1,sizeof(int));

      msg1.mtype = 1;
      msgsnd(mesgq, (struct msgbuf *)&msg1,
         sizeof(MYMSGBUF) - sizeof(long), IPC_NOWAIT);
      msgrcv(mesgq, (struct msgbuf *)&msg1, sizeof(int), 2L , MSG_NOERROR);
      memcpy(&temp, msg1.text, sizeof(int));
      printf("%d * %d = %d \n",temp1, temp1, temp);

}// request_sqr_msg
```

```
int main()
{
  long int mesgq;

  mesgq = msgget(1, IPC_CREAT | 0666);

  if ((sid = fork()))
     request_sqr_msg(mesgq);
  else
     compute_sqr_msg(mesgq);

  msgctl(mesgq, IPC_RMID, &buff);

 return 0;
} /* main */
```

```
% cc msg_run4.c
% ./a.out
Enter number:
-38
-38 * -38 = 1444
%
```

```
/* group4.c - group demo.  */

#include <stdio.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>

void sys_err(char str[])
{
  perror(str);
  exit(1);
} /*  sys_err */

int main()
{
  int grpid, new_grppid, old_grppid, i;

  grpid = getpgrp();

  printf("pid = %d\n", getpid());
  printf("grpid = %d\n\n", grpid);

  switch(fork())
  {
   case -1:
      sys_err("fork");
   case 0:
    for (i=1; i < 50; i++)
     {
       printf("Child 1:... \n");
       sleep(2);
     } /* for */
    exit(0);

  } /* switch */

  switch(fork())
  {
   case -1:
      sys_err("fork");
   case 0:
    for (i=1; i < 50; i++)
     {
       printf("Child 2:... \n");
       sleep(2);
     } /* for */
    exit(0);

  } /* switch */


  switch(fork())
  {
   case -1:
      sys_err("fork");
   case 0:
      old_grppid = getpgrp();
      if (setpgrp() == -1)
             sys_err("setpgrp");

      new_grppid = getpgrp();

     printf("Child 3:pid = %d\n", getpid());
     printf("old_grppid = %d, new_grppid = %d\n\n",
                                old_grppid, new_grppid);

     for (i=1; i < 10; i++)
      {
        printf("Child 3:... \n");
        sleep(2);
      } /* for */
     exit(0);

   } /* switch */


    for (i=0; i < 5; i++)
      {
        printf("Parent Process:... \n");
        sleep(2);
      } /* for */

    kill(0, SIGTERM);

   return 0;

 } /* main */
```

```
% cc group4.c
% ./a.out
pid = 5540
grpid = 5540

Child 1:...
Child 2:...
Child 3:pid = 5543
old_grppid = 5540, new_grppid = 5543

Child 3:...
Parent Process:...
Child 1:...
Child 2:...
Child 3:...
Parent Process:...
Child 2:...
Child 3:...
Parent Process:...
Child 1:...
Child 2:...
Child 3:...
Parent Process:...
Child 1:...
Child 2:...
Child 3:...
Parent Process:...
Child 1:...
Child 2:...
Child 3:...
Terminated
% Child 3:...
Child 3:...
Child 3:...

%
```

```
/*  date7.c - Use of fork() and wait() */

#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
~~#include <sys/wait.h>~~

int main()
{
  int id, wid, status;

  signal(SIGCHLD,SIG_IGN);

  printf("Here comes the date:\n");

  switch (  id =  fork() )
  {  /* select child process */

   case -1:
           perror("fork");
           exit(1);

   case 0:
       execl("/bin/date", "date",0);
       perror("execl");
       exit(1);

   default:
          break;

  } /* switch */

  wid = wait(&status);
  printf("That was the date.\n");

  return 0;

} /* main */
```

```
% cc date7.c
% ./a.out
Here comes the date:
Thu Mar 12 15:12:23 IST 2009
That was the date.
%
```

```
/* copy2.c */

#include <stdio.h>

int main(int argc, char *argv[] )
{
  ~~int buff;~~

   if (argc > 1)
    if ( freopen(argv[1], "rt", stdin) == NULL)
     {
      perror("freopen");
      exit(1);
     } /* if */

   if (argc > 2)
    if ( freopen(argv[2], "wt", stdout) == NULL)
     {
      perror("freopen");
      exit(1);
     } /* if */


   while ( (buff = getchar()) != EOF ) /* read until Ctrl-D */
      putchar(buff);


} /* main */
```

```
/* myutils.h - include declarations of utilities */

extern void fatal();
extern void syserr();
```

```
/* myutils.c - service utility routines */

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

extern int errno, sys_nerr;

/*  print system error messge and  terminate */
void syserr(msg)
char *msg;
{
 perror(msg);
 exit(1);
} /* syserr */

/*  print application error messge and  terminate */
void fatal(msg)
char *msg;
{
 fprintf(stderr, "Application program error: %s.", msg);
 exit(2);
}
```

```
/* env1.c - perform "env > envfile ".  */

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include "myutils.h"

int main()
{
   int fd;

   if ( (fd = open("envfile.txt", O_WRONLY | O_CREAT,0666)) == -1 )
       syserr("open");

   if (close(1) == -1)
        syserr("close");

   if ( dup(fd) != 1 )
         fatal("dup");

   execl("/usr/bin/env", "env", NULL);
   syserr("execl");

  return 0;
} /* main */
```

```
% cc env1.c myutils.c
% ./a.out
% more envfile.txt
PATH=/usr/local/netbeans-6.0.1/bin/:/usr/local/eclipse:/usr/local/bin:/usr/local
/teTeX/bin/x86_64-unknown-linux-gnu/:/usr/local/matlab/bin:/usr/bin:/bin:/usr/sb
in:/sbin:/home3/ronn/bin:/usr/games:/opt/gnome/bin:/opt/kde3/bin:/usr/bin/X11:/u
sr/lib/mit/bin:/usr/lib/mit/sbin:.
SHELL=/bin/tcsh
TERM=xterm
HOSTTYPE=x86_64-linux
VENDOR=suse
OSTYPE=linux
MACHTYPE=x86_64
SHLVL=2
GROUP=users
HOST=sci2
CSHEDIT=emacs
CPU=x86_64
.....
```

```
/* env2.c - perform "env > envfile ", using a new process.  */

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
~~#include "myutils.h"~~


int main()
{
   int fd, status;

   switch(fork())
   {
     case -1:
       syserr("fork");

     case 0:  /* Son process */
      if ( (fd = open("envfile.txt", O_WRONLY | O_CREAT,0600)) == -1 )
          syserr("open");

      if (close(1) == -1)
          syserr("close");

      if ( dup(fd) != 1 )
          fatal("dup");

      execl("/usr/bin/env", "env", NULL);
      syserr("execl");

   } /* switch  */

   wait(&status);
   puts("\n*** Parent process terminating...");

return 0;

} /* main */
```

```
% cc env2.c myutils.c
% ./a.out

*** Parent process terminating...
%
```

```
% cc errlist4.c
/tmp/ccmbHAm0.o: In function 'main':
errlist4.c:(.text+0xa): warning: 'sys_nerr' is deprecated; use
'strerror' or 'strerror_r' instead
% a.out

 Here are the current 132 error messages:

0: Success
1: Operation not permitted
2: No such file or directory
3: No such process
4: Interrupted system call
5: Input/output error
6: No such device or address
7: Argument list too long
8: Exec format error
9: Bad file descriptor
10: No child processes
11: Resource temporarily unavailable
12: Cannot allocate memory
13: Permission denied
14: Bad address
15: Block device required
16: Device or resource busy
17: File exists
18: Invalid cross-device link
19: No such device
20: Not a directory
21: Is a directory
22: Invalid argument
23: Too many open files in system
24: Too many open files
25: Inappropriate ioctl for device
26: Text file busy
27: File too large
28: No space left on device
29: Illegal seek
30: Read-only file system
31: Too many links
32: Broken pipe
33: Numerical argument out of domain
34: Numerical result out of range
35: Resource deadlock avoided
36: File name too long
37: No locks available
38: Function not implemented
39: Directory not empty
40: Too many levels of symbolic links
41: Unknown error 41
42: No message of desired type
43: Identifier removed
44: Channel number out of range
45: Level 2 not synchronized
46: Level 3 halted
47: Level 3 reset
48: Link number out of range
49: Protocol driver not attached
50: No CSI structure available
51: Level 2 halted
52: Invalid exchange
```

```
53: Invalid request descriptor
54: Exchange full
55: No anode
56: Invalid request code
57: Invalid slot
58: Unknown error 58
59: Bad font file format
60: Device not a stream
61: No data available
62: Timer expired
63: Out of streams resources
64: Machine is not on the network
65: Package not installed
66: Object is remote
67: Link has been severed
68: Advertise error
69: Srmount error
70: Communication error on send
71: Protocol error
72: Multihop attempted
73: RFS specific error
74: Bad message
75: Value too large for defined data type
76: Name not unique on network
77: File descriptor in bad state
78: Remote address changed
79: Can not access a needed shared library
80: Accessing a corrupted shared library
81: .lib section in a.out corrupted
82: Attempting to link in too many shared libraries
83: Cannot exec a shared library directly
84: Invalid or incomplete multibyte or wide character
85: Interrupted system call should be restarted
86: Streams pipe error
87: Too many users
88: Socket operation on non-socket
89: Destination address required
90: Message too long
91: Protocol wrong type for socket
92: Protocol not available
93: Protocol not supported
94: Socket type not supported
95: Operation not supported
96: Protocol family not supported
97: Address family not supported by protocol
98: Address already in use
99: Cannot assign requested address
100: Network is down
101: Network is unreachable
102: Network dropped connection on reset
103: Software caused connection abort
104: Connection reset by peer
105: No buffer space available
106: Transport endpoint is already connected
107: Transport endpoint is not connected
108: Cannot send after transport endpoint shutdown
109: Too many references: cannot splice
110: Connection timed out
111: Connection refused
112: Host is down
113: No route to host
```

```
114: Operation already in progress
115: Operation now in progress
116: Stale NFS file handle
117: Structure needs cleaning
118: Not a XENIX named type file
119: No XENIX semaphores available
120: Is a named type file
121: Remote I/O error
122: Disk quota exceeded
123: No medium found
124: Wrong medium type
125: Operation canceled
126: Required key not available
127: Key has expired
128: Key has been revoked
129: Key was rejected by service
130: Owner died
131: State not recoverable
```

```
% cc errlist3.c
/tmp/cc2CNccj.o: In function 'main':
errlist3.c:(.text+0x2f): warning: 'sys_errlist' is deprecated; use
'strerror' or 'strerror_r' instead
errlist3.c:(.text+0xa): warning: 'sys_nerr' is deprecated; use
'strerror' or 'strerror_r' instead
% a.out

 Here are the current 132 error messages:

0: Success
1: Operation not permitted
2: No such file or directory
3: No such process
4: Interrupted system call
5: Input/output error
6: No such device or address
7: Argument list too long
8: Exec format error
9: Bad file descriptor
10: No child processes
11: Resource temporarily unavailable
12: Cannot allocate memory
13: Permission denied
14: Bad address
15: Block device required
16: Device or resource busy
17: File exists
18: Invalid cross-device link
19: No such device
20: Not a directory
21: Is a directory
22: Invalid argument
23: Too many open files in system
24: Too many open files
25: Inappropriate ioctl for device
26: Text file busy
27: File too large
28: No space left on device
29: Illegal seek
30: Read-only file system
31: Too many links
32: Broken pipe
33: Numerical argument out of domain
34: Numerical result out of range
35: Resource deadlock avoided
36: File name too long
37: No locks available
38: Function not implemented
39: Directory not empty
40: Too many levels of symbolic links
41: (null)
42: No message of desired type
43: Identifier removed
44: Channel number out of range
45: Level 2 not synchronized
```

```
46: Level 3 halted
47: Level 3 reset
48: Link number out of range
49: Protocol driver not attached
50: No CSI structure available
51: Level 2 halted
~~52: Invalid exchange~~
53: Invalid request descriptor
54: Exchange full
55: No anode
56: Invalid request code
57: Invalid slot
58: (null)
59: Bad font file format
60: Device not a stream
61: No data available
62: Timer expired
63: Out of streams resources
64: Machine is not on the network
65: Package not installed
66: Object is remote
67: Link has been severed
68: Advertise error
69: Srmount error
70: Communication error on send
71: Protocol error
72: Multihop attempted
73: RFS specific error
74: Bad message
75: Value too large for defined data type
76: Name not unique on network
77: File descriptor in bad state
78: Remote address changed
79: Can not access a needed shared library
80: Accessing a corrupted shared library
81: .lib section in a.out corrupted
82: Attempting to link in too many shared libraries
83: Cannot exec a shared library directly
84: Invalid or incomplete multibyte or wide character
85: Interrupted system call should be restarted
86: Streams pipe error
87: Too many users
88: Socket operation on non-socket
89: Destination address required
90: Message too long
91: Protocol wrong type for socket
92: Protocol not available
93: Protocol not supported
94: Socket type not supported
95: Operation not supported
96: Protocol family not supported
97: Address family not supported by protocol
98: Address already in use
99: Cannot assign requested address
100: Network is down
101: Network is unreachable
```

```
102: Network dropped connection on reset
103: Software caused connection abort
104: Connection reset by peer
105: No buffer space available
106: Transport endpoint is already connected
107: Transport endpoint is not connected
~~108: Cannot send after transport endpoint shutdown~~
109: Too many references: cannot splice
110: Connection timed out
111: Connection refused
112: Host is down
113: No route to host
114: Operation already in progress
115: Operation now in progress
116: Stale NFS file handle
117: Structure needs cleaning
118: Not a XENIX named type file
119: No XENIX semaphores available
120: Is a named type file
121: Remote I/O error
122: Disk quota exceeded
123: No medium found
124: Wrong medium type
125: Operation canceled
126: Required key not available
127: Key has expired
128: Key has been revoked
129: Key was rejected by service
130: Owner died
131: State not recoverable
```

```
/* fdate1.c - perform "date > datefile.txt ".  */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
~~#include "myutils.h"~~

int main()
{
   int fd;

   if ( (fd = open("datefile.txt", O_WRONLY | O_CREAT,0666)) == -1 )
       syserr("open");

   if (close(1) == -1)
        syserr("close");

   if ( dup(fd) != 1 )
         fatal("dup");

   execl("/bin/date", "date", NULL);
   syserr("execl");

  return 0;
} /* main */
```

```
% cc fdate1.c myutils.c
% ./a.out
% cat datefile.txt
Thu Mar 12 13:24:26 IST 2009
%
```

```
/* fdate2.c - perform "date > datefile.txt ", using a new process.  */


#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
~~#include <unistd.h>~~
#include "myutils.h"


int main()
{
   int fd, status;

   switch(fork())
   {
     case -1:
       syserr("fork");

     case 0:  /* Son process */
      if ( (fd = open("datefile.txt", O_WRONLY | O_CREAT,0600)) == -1 )
          syserr("open");

      if (close(1) == -1)
          syserr("close");

      if ( dup(fd) != 1 )
          fatal("dup");

      execl("/bin/date", "date", NULL);
      syserr("execl");

   } /* switch  */

   wait(&status);
   puts("\n*** Parent process terminating...");

 return 0;

} /* main */
```

```
% cc fdate2.c myutils.c
% ./a.out

*** Parent process terminating...
% cat datefile.txt
Thu Mar 12 13:30:52 IST 2009
%
```

```
/*  fiforecv.c */

#include <fcntl.h>
#include <stdio.h>
#include <errno.h>
~~#include <stdlib.h>~~
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#define MAX_MSGSIZE 80

char *fifo = "/tmp/newfifo";

void fatal(char str[])
{
 fprintf(stderr, "%s\n", str);
 exit(0);
} /* fatal */

int main()
{
 int fd;
 char msgbuf[MAX_MSGSIZE];

 /* Create fifo, unless it already exits */

 if(mkfifo(fifo, 0600) == -1)
   if (errno != EEXIST)
      fatal("mkfifo failed");

 if ( (fd = open(fifo, O_RDWR)) < 0)
   fatal("fifo open failed");

 if ( read(fd, msgbuf,MAX_MSGSIZE ) < 0)
   fatal("fifo write failed");

 printf("received: %s\n",  msgbuf);

 if (remove(fifo) < 0)
   perror("remove");

 return 0;

} /* main */
```

```
/*  fifosend.c */

#include <fcntl.h>
#include <stdio.h>
#include <errno.h>
~~#include <stdlib.h>~~
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

char *fifo = "/tmp/newfifo";

void fatal(char str[])
{
 fprintf(stderr, "%s\n", str);
 exit(0);
} /* fatal */

int main()
{
 int fd;
 char msgbuf[] = "Hello World!\n";

 if ( (fd = open(fifo, O_WRONLY | O_NONBLOCK)) < 0)
   fatal("fifo open failed");

 if ((write(fd, msgbuf, strlen(msgbuf)+1)) < 0)
   fatal("fifo write failed");

  return 0;

} /* main */
```

```
% cc fifosend.c -o fifosend
% cc fiforecv.c -o fiforecv
% ./fiforecv
received: Hello World!

%
```

```
% ./fifosend

%
```

```c
/* nice.c - Private version of the nice command.   */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define USAGE "usage: nice [-num] command\n"

int main(argc, argv) /* nice command */
int argc;
char *argv[];
{
 int incr, cmdarg, niceflag;

   if (argc < 2)
   {
    fprintf(stderr, USAGE);
    exit(1);
   } /* if */

   if (argv[1][0] == '-')
   {
    incr = atoi(&argv[1][1]);
    cmdarg = 2;
   } /* if */
   else
   {
    incr = 10;
    cmdarg = 1;
   } /* else  */

   if (cmdarg >= argc)
   {
    fprintf(stderr, USAGE);
    exit(1);
   } /* if */

   niceflag = nice(incr);
   printf("niceflag = %d\n", niceflag);
   execvp(argv[cmdarg], &argv[cmdarg]);
   perror("execvp");

   return 0;

} /* main */
```

```
/* nice2.c - nice command demo.  */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define USAGE "usage: nice [-num] command\n"

int main(argc, argv) /* nice command */
int argc;
char *argv[];
{
 int incr, cmdarg, niceflag, i;

  if (argc < 2)
  {
   fprintf(stderr, USAGE);
   exit(1);
  } /* if */

  if (argv[1][0] == '-')
  {
   incr = atoi(&argv[1][1]);
   cmdarg = 2;
  } /* if */
  else
  {
   incr = 10;
   cmdarg = 1;
  } /* else  */

  if (cmdarg >= argc)
  {
   fprintf(stderr, USAGE);
   exit(1);
  } /* if */

  switch(fork())
  {
   case -1:
      perror("fork");
   case 0:
    niceflag = nice(incr);
    printf("niceflag = %d\n", niceflag);
    execvp(argv[cmdarg], &argv[cmdarg]);
    perror("execvp");
  }

  for (i=1; i < 50; i++)
  {
    printf("Ha Ha Ha Ha \n");
    sleep(2);
   }

return 0;
} /* main */
```

```
% cc nice.c
% ./a.out date
niceflag = 10
Thu Mar 12 14:19:47 IST 2009
% cc -Wall nice2.c
% ./a.out date
Ha Ha Ha Ha
niceflag = 10
Thu Mar 12 14:19:57 IST 2009
Ha Ha Ha Ha

%
```

```
/* pth2.c */

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *print_message_function( void *ptr );

int main()
{
     pthread_t thread1, thread2;
     char *message1 = "Thread 1";
     char *message2 = "Thread 2";
     int  iret1, iret2;

    /* Create independent threads each of which will execute function */

     iret1 = pthread_create ( &thread1, NULL, print_message_function,
(void*) message1);
     iret2 = pthread_create( &thread2, NULL, print_message_function,
(void*) message2);

     /* Wait till threads are complete before main continues. Unless we
*/
     /* wait we run the risk of executing an exit which will terminate
*/
     /* the process and all threads before the threads have completed.
*/

     pthread_join ( thread1, NULL);
     pthread_join( thread2, NULL);

     printf("thread1 = %lu\n",thread1);
     printf("thread2 = %lu\n",thread2);
     printf("Thread 1 returns: %d\n",iret1);
     printf("Thread 2 returns: %d\n",iret2);
     exit(0);
}

void *print_message_function( void *ptr )
{
     char *message;
     message = (char *) ptr;
     printf("%s \n", message);
     return NULL;
}
```

```
% cc -lpthread pth2.c
% ./a.out
Thread 1
Thread 2
thread1 = 1082132800
thread2 = 1090525504
Thread 1 returns: 0
Thread 2 returns: 0
%
```

```
/* pthr2.c */

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *print_message_function( void *ptr );

int main()
{
     pthread_t thread1, thread2;
     int  iret1, iret2;

    /* Create independent threads each of which will execute function */

     iret1 = pthread_create ( &thread1, NULL, print_message_function,
(void*) NULL);
     iret2 = pthread_create( &thread2, NULL, print_message_function,
(void*) NULL);

     /* Wait till threads are complete before main continues. Unless we
*/
     /* wait we run the risk of executing an exit which will terminate
*/
     /* the process and all threads before the threads have completed.
*/

     pthread_join ( thread1, NULL);
     pthread_join( thread2, NULL);

     printf("Thread 1 returns: %d\n",iret1);
     printf("Thread 2 returns: %d\n",iret2);
     exit(0);
}

void *print_message_function( void *ptr )
{
     int i=1;

     i = i *7;

     printf("i = %d, &i = %p  \n", i, &i);
     return NULL;
}
```

```
% cc -lpthread pthr2.c
% ./a.out
i = 7, &i = 0x408001bc
i = 7, &i = 0x410011bc
Thread 1 returns: 0
Thread 2 returns: 0
%
```