דגמאות נוספות

```
/* forkhello.c */

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

int main()
{

  printf("Hello ");

  fork();

  printf("World!\n");

  return 0;

} /* main */
```

```
% cc forkhello.c
% ./a.out
Hello World!
Hello World!
% ./a.out > fhello.txt
% cat fhello.txt
Hello World!
Hello World!
%
```

```c
/* forkhello1.c */


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

```c
int main()
{

  printf("Hello \n");

  fork();

  printf("World!\n");

  return 0;

} /* main */
```

```
% cc forkhello1.c
% ./a.out
Hello
World!
World!
% ./a.out > fhello.txt
% cat fhello.txt
Hello
World!
Hello
World!
%
```

```
/* forkhello2.c */

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

int main()
{

  printf("Hello \n");

  if( fork())
     printf("World1!\n");
  else
     printf("World2!\n");

  return 0;

} /* main */
```

```
% cc forkhello2.c
% ./a.out
Hello
World2!
World1!
% ./a.out > fhello.txt
% cat fhello.txt
Hello
World2!
Hello
World1!
%
```

```
/* twait.c - Use the system command */

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

int time_restricted_wait(int secs, int *status)
{
 int id, result;

  if ( (id = fork()) == 0)
  {
   sleep(secs);
   exit(0);
  }
  else /* paraent process */
  {
   result = wait(status);
   if (id == result)
     return 0;
   else
    return result;
  }

}

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

  if ((id = fork()) == 0)
      {
        sleep(8);
        exit(0);
      }
   else
    child_id = time_restricted_wait(4, &status);

  printf("\ngetpid = %d, id = %d\n", getpid(), id);
  printf("\nchild_id = %d, status = %d\n", child_id, status);

  child_id = time_restricted_wait(8, &status);

  printf("\nchild_id = %d, status = %d\n", child_id, status);

 return 0;

}
```

```
% cc twait.c
% ./a.out

getpid = 22403, id = 22404

child_id = 0, status = 0

child_id = 22404, status = 0
%
```

```
/*  errlist2.c - display system error messages */

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

main()
{
 int i;

 printf(" Here are the current %d error messages:\n\n",
                                                  sys_nerr);

 for (i=0; i < sys_nerr; i++)
    printf("%d: %s \n", i, sys_errlist[i]);
} /* main */
```

```
% 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
```

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

```
/*  errlist4.c - display system error messages */

#include <unistd.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

main()
{
 int i;
 char *cptr;

 printf(" Here are the current %d error messages:\n\n",
                                                  sys_nerr);

 for (i=0; i < sys_nerr; i++)
  {
    cptr = strerror(i);
    printf("%d: %s \n", i, cptr);
  } /* for */

} /* main */
```

```
% cc errlist4.c
/tmp/ccmbEAmO.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
```

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

```
% ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000fc8 229379     ronn       666        4          0

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

% ipcrm -m 229379
% ipcs

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

%
```

```
/* freesh.c -  free segment with nickname 5656 */


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

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>


void sys_err(char s[])
{
   perror(s);
   exit(1);
}


int main()
{
  int memid;
  struct shmid_ds buff;


     if ( (memid = shmget(5656, sizeof(int), 0666)) < 0)
         sys_err("Cannot shmget");



                              /* Release the shared segment */
     if ( shmctl(memid, IPC_RMID, &buff) < 0 )
        sys_err("Cannot shmctl");

  return 0;

} /* main */
```

```
% cc freesh.c
% ./a.out
Cannot shmget: No such file or directory
%
```

```
/* async1.c -  prod2, cons2 */

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

volatile int    *n;


/*------------------------------------------------------------------------
 *  main  --  producer and consumer processes synchronized with semaphores
 *------------------------------------------------------------------------
 */
int  main()
{
        int     prod2(), cons2();
        int     semid, shmid;
        struct shmid_ds buff;

        shmid = shmget(0, sizeof(int), 0666);
        n = (int*)shmat(shmid,0,0);
        *n = 0;


        if ( fork() )
          /* Parent process    */
            cons2();
        else /* Child process */
         prod2();

        shmdt((char *)n);

        shmctl(shmid, IPC_RMID, &buff);

   return 0;
}

/*------------------------------------------------------------------------
 *  prod2  --  increment n 20 times
 *------------------------------------------------------------------------
 */

int prod2()
{
        int     i;


        for (i=1; i<=20; i++) {
                (*n)++;
        }
}

/*------------------------------------------------------------------------
 *  cons2  --  print n 20 times
```

```
 *-------------------------------------------------------------------
 */

int cons2()
{
        int     i;
```

```
        for (i=1; i<=20; i++) {
                printf("n is %d\n", *n);
        }
}
```

```
% cc async1.c
% ./a.out
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
n is 20
%
```

```
/* async2.c -  prod2, cons2 */

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

volatile int    *n;


/*------------------------------------------------------------------------
 *  main  --  producer and consumer processes synchronized with semaphores
 *------------------------------------------------------------------------
 */
int  main()
{
        int     prod2(), cons2();
        int     semid, shmid;
        struct  shmid_ds buff;

        shmid = shmget(IPC_PRIVATE, sizeof(int), 0666);
        n = (int*)shmat(shmid,0,0);
        *n = 0;


        if ( fork() == 0 )
          /* Child process    */
            cons2();
        else /* Parent process */
         prod2();

        shmdt((char *)n);
        shmctl(shmid, IPC_RMID, &buff);

   return 0;

}

/*------------------------------------------------------------------------
 *  prod2  --  increment n 20 times
 *------------------------------------------------------------------------
 */

int prod2()
{
        int     i;


        for (i=1; i<=20; i++) {
                (*n)++;
        }
}

/*------------------------------------------------------------------------
 *  cons2  --  print n 20 times
```

```
 *-----------------------------------------------------------------------
 */

int cons2()
{
        int     i;
```

```
        for (i=1; i<=20; i++) {
                printf("n is %d\n", *n);
        }
}
```

```
% cc async2.c
% ./a.out
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
n is 0
%
```

```
/* shmq2.c -  prod2, cons2 */

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


int     *n;             /* external variables are shared by all processes */
int     childpid;

typedef struct string_rec
{
   int busy_flag;
   char data[80];

} STRING_REC, *STRING_REC_PTR;


STRING_REC_PTR string_create(key)
int key;
{
   int result;
    STRING_REC_PTR semrec;


   result = shmget( (key_t) key,  sizeof(STRING_REC), 0666|IPC_CREAT);
   semrec = (STRING_REC_PTR) shmat(result, 0, 0);

   (*semrec).busy_flag = 0;
   (*semrec).data[0] = 0;


   return semrec;

}

void read_string(char dest[], STRING_REC_PTR  sptr )
{
   while ((*sptr).busy_flag == 1 )
     ;

 (*sptr).busy_flag = 1;
 strcpy(dest,(*sptr).data);
 (*sptr).busy_flag = 0;

}

void write_string(STRING_REC_PTR  sptr, char source[] )
{
   while ((*sptr).busy_flag == 1 )
     ;
 (*sptr).busy_flag = 1;
  strcpy((*sptr).data, source);
 (*sptr).busy_flag = 0;

}



int main()
{
```

```
   int id;
   STRING_REC_PTR sptr;
   char str[80];

   sptr = string_create(11);

   if ( ( id =  fork() ) != 0 )
   {  /* select parent process */
      puts("**************  parent process ***********\n");
      write_string(sptr,"Hello World");
      read_string(str, sptr);
      printf("string is %s\n", str);
      puts("\n**************  parent process ***********\n");

      sleep(6);
      puts("**************  parent process ***********\n");
      read_string(str, sptr);
      printf("string is %s\n", str);
      puts("\n**************  parent process ***********\n");
      exit(0);
   }


      sleep(5);
      puts("**************  child process ***********\n");
      read_string(str, sptr);
      printf("string is %s\n", str);
      write_string(sptr,"Goodbye World");
      read_string(str, sptr);
      printf("string is %s\n", str);
      puts("\n**************  child process ***********\n");

  return 0;
}
```

```
% cc shmq2.c
% ./a.out
**************  parent process ***********

string is Hello World

**************  parent process ***********

**************  child process ***********

string is Hello World
string is Goodbye World

**************  child process ***********

**************  parent process ***********

string is Goodbye World

**************  parent process ***********

%
```

```
/* demo_sig_par.c -- traps a signal -  */

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

void handler(int signum)
{
   if(signum == SIGINT)
   {
    signal(SIGINT, handler);
    printf("I wont be interrupted!\n");
   } /* if */
   else
     if(signum == SIGTSTP)
     {
      signal(SIGTSTP, handler);
      printf("I wont be suspended!\n");
     } /* if */

} // handler


int main()
{
  int i, pid;

  signal(SIGINT, handler);
  signal(SIGTSTP, handler);

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

  return 0;

} /* main */
```

```
% cc demo_sig_par.c
% ./a.out
Ha Ha Ha Ha
Ha Ha Ha Ha
Ha Ha Ha Ha
I wont be suspended!
Ha Ha Ha Ha
Ha Ha Ha Ha
I wont be interrupted!
Ha Ha Ha Ha
Ha Ha Ha Ha
Quit
%
```

```
// uncrt3.c

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


int create(char path[], char argv0[], char parm1[], char parm2[],
char parm3[])
{
  int pid;

  if ( (pid = fork()) == 0  )
  {
    kill(getpid(), SIGSTOP);
    execl(path, argv0, parm1, parm2, parm3, 0 );
    exit(0);
  } /* child */
  else /* parent */
   return pid;

} /* create */


int resume(int pid)
{
  return (kill(pid, SIGCONT));
} /* resume */

int main(void)
{
  int pid;


  printf("Before Create\n");

  pid = create("/bin/ping", "ping", "-c", "2","study.haifa.ac.il");

  sleep(3);

  printf("Before Resume\n");

  resume(pid);

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

```
% cc uncrt3.c
% ./a.out
Before Create
Before Resume
% PING study.haifa.ac.il (132.74.1.26) 56(84) bytes of data.
```

```
64 bytes from study.haifa.ac.il (132.74.1.26): icmp_seq=1 ttl=254
time=0.445 ms
64 bytes from study.haifa.ac.il (132.74.1.26): icmp_seq=2 ttl=254
time=0.390 ms

--- study.haifa.ac.il ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.390/0.417/0.445/0.034 ms

%
```

```
/* sync1.c -  prod2, cons2 */

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

typedef struct n
{
  int n;
  int flag1;
  int flag2;

} N, *NPTR;

volatile NPTR n;



/*------------------------------------------------------------------------
 *  main  --  producer and consumer processes synchronized with semaphores
 *------------------------------------------------------------------------
 */
int  main()
{
        int     prod2(), cons2();
        int     shmid;
        struct shmid_ds buff;


        shmid = shmget(IPC_PRIVATE, sizeof(N), 0666);
        n = (NPTR)shmat(shmid,0,0);
        n->n = 0;
        n->flag1 = 0;
        n->flag2 = 1;


        if ( fork() )
          /* Parent process     */
            cons2();
        else /* Child process */
         prod2();

        shmdt((char *)n);
        shmctl(shmid, IPC_RMID, &buff);

   return 0;
}

/*------------------------------------------------------------------------
 *  prod2  --  increment n 20 times
 *------------------------------------------------------------------------
 */
```

```
int prod2()
{
        int     i;


        for (i=1; i<=20; i++) {
             while(n->flag1 == 0)
                  ;
             (n->n)++;
             n->flag2 = 1;
             n->flag1 = 0;
        }
  return 0;
}

/*-------------------------------------------------------------------------
 *  cons2  --  print n 20 times
 *-------------------------------------------------------------------------
 */

int cons2()
{
        int     i;



        for (i=1; i<=20; i++) {
             while(n->flag2 == 0)
                  ;
             printf("n is %d\n", n->n);
             n->flag1 = 1;
             n->flag2 = 0;
        }
  return 0;
}
```

```
% cc sync1.c
% a.out
n is 0
n is 1
n is 2
n is 3
n is 4
n is 5
n is 6
n is 7
n is 8
n is 9
n is 10
n is 11
n is 12
n is 13
n is 14
n is 15
```

```
n is 16
n is 17
n is 18
n is 19
%
```