ΕΡΓΑΣΤΗΡΙΟ 9

Διαδιεργασιακή Επικοινωνία

Να πληκτρολογηθεί και να εκτελεστεί ο κώδικας των επόμενων παραδειγμάτων που επιδεικνύουν τους διάφορους τρόπους επικοινωνίας μεταξύ διεργασιών.

Παράδειγμα 1. Επικοινωνία διεργασιών με κλείδωμα κοινόχρηστου αρχείου (αρχεία producer.c και consumer.c).

Αρχείο producer.c

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define FileName "data.dat"
#define DataString "Now is the winter of our discontent\n"

int main() {
 struct flock lock;
 lock.l_type = F_WRLCK;    /* read/write (exclusive versus shared) lock */
 lock.l_whence = SEEK_SET; /* base for seek offsets */
 lock.l_start = 0;         /* 1st byte in file */
 lock.l_len = 0;           /* 0 here means 'until EOF' */
 lock.l_pid = getpid();    /* process id */

 int fd; /* file descriptor to identify a file within a process */
 if ((fd = open(FileName, O_RDWR | O_CREAT, 0666)) < 0)  {
     perror("open failed..."); exit(-1); }
 if (fcntl(fd, F_SETLK, &lock) < 0) {
     perror("fcntl failed to get lock..."); exit(-2); }
 else {
   write(fd, DataString, strlen(DataString)); /* populate data file */
   fprintf(stderr, "Process %d has written to data file...\n", lock.l_pid); }

 /* Now release the lock explicitly. */
 lock.l_type = F_UNLCK;
 if (fcntl(fd, F_SETLK, &lock) < 0) {
     perror("explicit unlocking failed..."); exit(-1); }

 close(fd);
 return 0; }
```

Αρχείο consumer.c

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
```

```
#include <unistd.h>
#define FileName "data.dat"

int main() {
 struct flock lock;
 lock.l_type = F_WRLCK;    /* read/write (exclusive) lock */
 lock.l_whence = SEEK_SET; /* base for seek offsets */
 lock.l_start = 0;         /* 1st byte in file */
 lock.l_len = 0;           /* 0 here means 'until EOF' */
 lock.l_pid = getpid();    /* process id */

 int fd; /* file descriptor to identify a file within a process */
 if ((fd = open(FileName, O_RDONLY)) < 0) {
     perror("open to read failed..."); exit(-1); }

 /* If the file is write-locked, we can't continue. */
 fcntl(fd, F_GETLK, &lock); /* sets lock.l_type to F_UNLCK if no write lock */
 if (lock.l_type != F_UNLCK) {
     perror("file is still write locked..."); exit(-1); }

 lock.l_type = F_RDLCK; /* prevents any writing during the reading */
 if (fcntl(fd, F_SETLK, &lock) < 0) {
     perror("can't get a read-only lock..."); exit(-1); }

 /* Read the bytes (they happen to be ASCII codes) one at a time. */
 int c; /* buffer for read bytes */
 while (read(fd, &c, 1) > 0)    /* 0 signals EOF */
   write(STDOUT_FILENO, &c, 1); /* write one byte to the standard output */

 /* Release the lock explicitly. */
 lock.l_type = F_UNLCK;
 if (fcntl(fd, F_SETLK, &lock) < 0) {
     perror("explicit unlocking failed..."); exit(-1); }

 close(fd);
 return 0; }
```

Παράδειγμα 2. Επικοινωνία διεργασιών με χρήση κοινόχρηστης μνήμης (αρχεία memwriter.c και memreader.c).

Αρχείο memwriter.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <semaphore.h>
```

```
#include <string.h>
#include "shmem.h"

#define ByteSize 512
#define MemContents "This is the way the world ends...\n"

int main() {


int fd = shm_open("/shMemEx", O_RDWR | O_CREAT, 0644);
if (fd < 0) { perror ("Can't open shared mem segment..."); exit (-1); }
ftruncate(fd, ByteSize);

caddr_t memptr = mmap(NULL, ByteSize, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
if ((caddr_t)-1==memptr) { perror ("Can't get segment..."); exit (-2); }

sem_t* semptr = sem_open("sem", O_CREAT, 0644, 0);
if (semptr==(void*) -1) { perror ("sem_open"); exit (-2); }
strcpy(memptr, MemContents);

/* increment the semaphore so that memreader can read */
if (sem_post(semptr) < 0) { perror ("sem_post"); exit (-3); }
sleep(12); /* give reader a chance */

munmap(memptr, ByteSize); /* unmap the storage */


close(fd);
sem_close(semptr);
shm_unlink("/shMemEx");

return 0; }
```

Αρχείο memreader.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <semaphore.h>
#include <string.h>
#include "shmem.h"

#define ByteSize 512
#define MemContents "This is the way the world ends...\n"

int main() {

  int fd = shm_open("/shMemEx", O_RDWR, 0644);
  if (fd < 0) { perror ("Can't get file descriptor..."); exit (-1); }

  caddr_t memptr = mmap(NULL, ByteSize, PROT_READ | PROT_WRITE,
                        MAP_SHARED, fd, 0);
  if ((caddr_t)-1==memptr) { perror ("Can't access segment..."); exit (-1); }
```

```
 sem_t* semptr = sem_open("sem", O_CREAT, 0644, 0);
 if (semptr == (void*)-1) { perror ("sem open"); exit (-1); }

 if (!sem_wait(semptr)) { /* wait until semaphore != 0 */
    int i;
    for (i = 0; i < strlen(MemContents); i++)
          write(STDOUT_FILENO, memptr + i, 1);
    sem_post(semptr); }

 munmap(memptr, ByteSize);
 close(fd);
 sem_close(semptr);
 unlink(BackingFile);
 return 0; }
```

Παράδειγμα 3. Επικοινωνία διεργασιών με χρήση ουράς μηνυμάτων (αρχείο mqExample.c).

```
#include <mqueue.h>
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <signal.h>
#include <string.h>

#define MSG_SIZE 16384

void handler (int sig_num) { printf ("Received sig %d.\n", sig_num); }

int main (int argc, char * argv []) {
 struct mq_attr attr, old_attr;
 struct sigevent sigevent;
 mqd_t mqdes, mqdes2;
 char * message = "Hello world";
 char buf [MSG_SIZE];
 unsigned int prio=0, i;
 mqdes = mq_open ("/mqueue1", O_RDWR | O_CREAT, 0664, NULL);
 mq_getattr (mqdes, &attr);
 printf ("Max number of messages on the queue ==> %ld messages.\n", attr.mq_maxmsg);
 printf ("Size of messages on the queue ==> %ld bytes.\n", attr.mq_msgsize);
 printf ("%ld messages are currently on the queue.\n", attr.mq_curmsgs);
 if (attr.mq_curmsgs != 0) {
  attr.mq_flags = O_NONBLOCK;
  mq_setattr (mqdes, &attr, &old_attr);
  while (mq_receive (mqdes, &buf[0], MSG_SIZE, &prio) != -1)
       printf ("Received a message with priority %d.\n", prio);
  if (errno != EAGAIN) { perror ("mq_receive()"); exit (EXIT_FAILURE); }
  mq_setattr (mqdes, &old_attr, 0); }
 signal (SIGUSR1, handler);
 sigevent.sigev_signo = SIGUSR1;
```

```
if (mq_notify (mqdes, &sigevent) == -1) {
  if (errno == EBUSY)
     printf ("Another process has registered for notification.\n");
     exit (EXIT_FAILURE); }
for (i= 0; i < attr.mq_maxmsg; i++) {
    printf ("Writing a message with priority %d.\n", prio);
    if (mq_send (mqdes, message, strlen(message)+1, prio) == -1) perror ("mq_send()");
    prio += 5;}
mq_close (mqdes);
return (0); }
```