communication interprocessus

Novembre 2008

On examine deux autres moyens de communication interprocessus offerts par les systèmes autres que les *tubes* : *file de message*, *mémoire partagés*. Un dernier moyen trés puissant concernant les communications processus distants (*socket*) sera vu dans le cours de programmation réseau.

1 File de message

L'exécution des commandes de la cible `demo1` est illustrée par la figure (1). Trois messages sont envoyés dans une file de messages identifiées par le nom de l'exécutable, puis lus avec des priorités différentes. La commande `ipcs` est utilisée pour tracer l'évolution de la file de messages.

(a) Expliquer le rôle des appels systèmes.

(b) Commenter la gestion du *type* des messages.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <sys/ipc.h>
5  #include <sys/msg.h>
6  #include <string.h>
7  #include <unistd.h>
8  #define MAX 256
9  int CTRL=0, SEND=0, RECV = 0;
10
11 typedef unsigned short int ushort;
12
13 typedef struct {
14     long  type;
15    char data[MAX];
16 } message;
17
18
19
20
21 int main(int argc, char* argv[] )
```

```
22 { int opt, type;
23   char *optlist ="cd:r:s:";
24   key_t key;
25   message msg;
26   int file;
27   while ( ( opt = getopt( argc, argv , optlist) ) > 0  )
28           switch( opt ){
29           case 'r': RECV = 1;        type = atoi( optarg ); break;
30           case 's': SEND = 1; msg . type = atoi( optarg ); break;
31           case 'c': CTRL = 1;                               break;
32           case 'd': strncpy( msg . data, optarg, MAX );  ; break;
33           default : printf("\nusage %s : %s", argv[0], optlist);
34                     exit ( 1 );
35           }
36
37   if ( ( key = ftok( argv[0], 0 )) == -1  )
38     perror("ftok"), exit( 1 );
39
40   if ( ( file = msgget( key, IPC_CREAT | 0600 ) ) == -1 )
41     perror("msget"), exit( 1 );
42
43   if ( SEND ) {
44           if ( msgsnd( file , ( void*) & msg, MAX , 0 ) < 0 )
45              perror("msgsnd"), exit(1);
46   }
47   if ( RECV ) {
48     if ( msgrcv( file , ( void*) & msg, MAX , type , 0 ) >= 0 )
49        printf("type=%ld data=%s", msg . type, msg . data);
50     else perror("msgrecv"), exit(1);
51   }
52
53   if ( CTRL )
54        msgctl( file , IPC_RMID, NULL);
55   printf("\n");
56   return 0;
57 }
```

2 Mémoire partagée

L'exécution des commandes de la cible `demo2` est illustrée par la figure (2). Une zone de 26 octets est partagée par deux processus. La commande `ipcs` est utilisée pour tracer la mémoire partagée.

(a) Expliquer le rôle des appels systèmes.

(b) Commenter l'exécution.

(c) L'ensemble est-t-il correct ?

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <limits.h>
#include <unistd.h>
#define MAX 26
void ptable( int s, char * t)
{ int i;
  if ( s ) return;
  printf("\n%d : ", s);
  for(i = 0; i < MAX; i++)
    printf(" %c", t[i] );
}
void itable( char * t)
{ int i;
  for(i = 0; i < MAX; i++)
    t[ MAX - i - 1] = 65 + i;
}
int test( char *t )
{ int i;
  for( i = 0; i < MAX - 1; i++)
    if ( t[i] >= t[i+1] ) return 1;
  return 0;
}
void maps( void )
{
  char bfr[256];
  sprintf(bfr, "cat /proc/%d/maps | grep rw-s", getpid() );
  system( bfr );
}


int main(int argc, char *argv[])
{
  key_t key;
  int       shm;
  char * t = NULL;
  int i, s = 0, tmp, change = 1, cpt;

  if ( ( key = ftok( argv[0], 0 ) ) == -1  )
    perror("ftok"), exit( 1 );

  if ( ( shm =shmget( key, MAX, IPC_CREAT | 0600) ) < 0 )
    perror("shmget"), exit( 1 );

  if ( ( t = shmat( shm, NULL, 0 )) == NULL )
    perror("shmat"), exit ( 1 );

```

```
51    s = *(argv[1]) - '0';
52    if ( s ) itable( t );
53    else system("./shm.exe 1 &");
54
55    maps();
56
57    cpt = 1 << 25;
58
59    while ( cpt-- ) {
60      if ( change) ptable( s , t );
61      change = 0;
62      for(i = s; i + 1 < MAX; i+=2)
63        if ( t[i] > t[i+1] ) {
64          tmp    = t[i];
65          t[i]   = t[i+1];
66          t[i+1] = tmp;
67          change = 1;
68        }
69    }
70    putchar('\n');
71    return 0;
72  }
```

(a)

(b)

3 Makefile

```
cible = msg.exe shm.exe

all : $(cible) ipc.pdf

$(cible):%.exe:%.c
        gcc -Wall -g $< -o $@

mklistings: mklistings.lex
        flex mklistings.lex
        gcc -Wall lex.yy.c -omklistings -lfl

ipc.pdf: listings.tex ipc.tex
        pdflatex ipc.tex
        pdflatex ipc.tex

listings.tex : $(cible)
        ./mklistings msg.c     > listings.tex
        ./mklistings shm.c    >> listings.tex
```

```
demo1:
	./msg.exe -s1 -dhello_1
	./msg.exe -s2 -dhello_2
	./msg.exe -s3 -dhello_3
	./msg.exe -s4 -dhello_4
	ipcs -q
	./msg.exe -r2
	./msg.exe -r-3
	./msg.exe -r0
	ipcs -q
	./msg.exe -c
	ipcs -q

demo2:
	./shm.exe 0
	ipcs -m | head -3
	ipcs -m | grep 26

output:
	make demo1 | grep -v make > msg.out
	make demo2 | grep -v make > shm.out
```

```
./msg.exe -s1 -dhello_1

./msg.exe -s2 -dhello_2

./msg.exe -s3 -dhello_3

./msg.exe -s4 -dhello_4

ipcs -q

------ Queues de messages --------
cl         msqid      propri taire perms
octets-utilis s messages
0x0000f598 557056     langevin   600        1024          4

./msg.exe -r2
type=2 data=hello_2
./msg.exe -r-3
type=1 data=hello_1
./msg.exe -r0
type=3 data=hello_3
ipcs -q

------ Queues de messages --------
cl         msqid      propri taire perms
octets-utilis s messages
0x0000f598 557056     langevin   600        256           1

./msg.exe -c

ipcs -q

------ Queues de messages --------
cl         msqid      propri taire perms
octets-utilis s messages
```

Figure 1: msg.out

```
./shm.exe 0
b7ffa000-b7ffb000 rw-s 00000000 00:08 2654233
/SYSV0000f558 (deleted)
b7f8f000-b7f90000 rw-s 00000000 00:08 2654233
/SYSV0000f558 (deleted)


0 :  Z X Y V W T U R S P Q N O L M J K H I F G D E B C A
0 :  X Z V Y T W R U P S N Q L O J M H K F I D G B E A C
0 :  X V Z T Y R W P U N S L Q J O H M F D K B I A G C E
0 :  V X T Z R Y P W N U L S J Q H O F M D K B I A G C E
0 :  V T X R Z P Y N W L U J H S F Q D O B M A K C I E G
0 :  T V R X P Z N Y L W J U H S F Q D O B M A K C I E G
0 :  T R V P X N Z L Y J W H U F S D Q B A O C M E K G I
0 :  R T P V N X L Z J Y H W F U D S B Q A O C M E K G I
0 :  R P T N L V J X H Z F Y D W B U A S C Q E O G M I K
0 :  P R N T L V J X H Z F Y D W B U A S C Q E O G M I K
0 :  P N R L T J H V F X D Z B Y A W C U E S G Q I O K M
0 :  N P L R J T H V F X D Z B Y A W C U E S G Q I O K M
0 :  N L P J R H T F V D X B Z A C Y E W G U I S K Q M O
0 :  L N J P H R F T D V B X A Z C Y E W G U I S K Q M O
0 :  L J N H P F R D B T A V C X E Z G Y I W K U M S O Q
0 :  J L H N F P D R B T A V C X E Z G Y I W K U M S O Q
0 :  J H L F N D P B R A T C E V G X I Z K Y M W O U Q S
0 :  H J F L D N B P A R C T E V G X I Z K Y M W O U Q S
0 :  F H D J B L A N C P E R G T I V K X M Z O Y Q W S U
0 :  D F B H A J C L E N G P I R K T M V O X Q Z S Y U W
0 :  D B A F C H E J G L I N K P M R O T Q V S X U Z W Y
0 :  B D A F C H E J G L I N K P M R O T Q V S X U Z W Y
0 :  B A D C F E H G J I L K N M P O Q R S T U V W X Y Z
0 :  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ipcs -m | head -3

------ Segment de m moire partag --------
c l          shmid      propri taire perms      octets
nattch       tats
ipcs -m | grep 26
0x00000000 262144     langevin  600        393216
2          dest
0x00000000 622607     langevin  600        393216
2          dest
0x00000000 2326546    langevin  600        393216
2          dest
0x0000f59c 2588692    langevin  600        26          0
0x0000f556 2621462    langevin  600        26          0
0x0000f558 2654233    langevin  600        26          0
```

Figure 2: shm.out