

JEGYZŐKÖNYV

Operációs rendszerek BSc

2021.04.30.

Készítette:

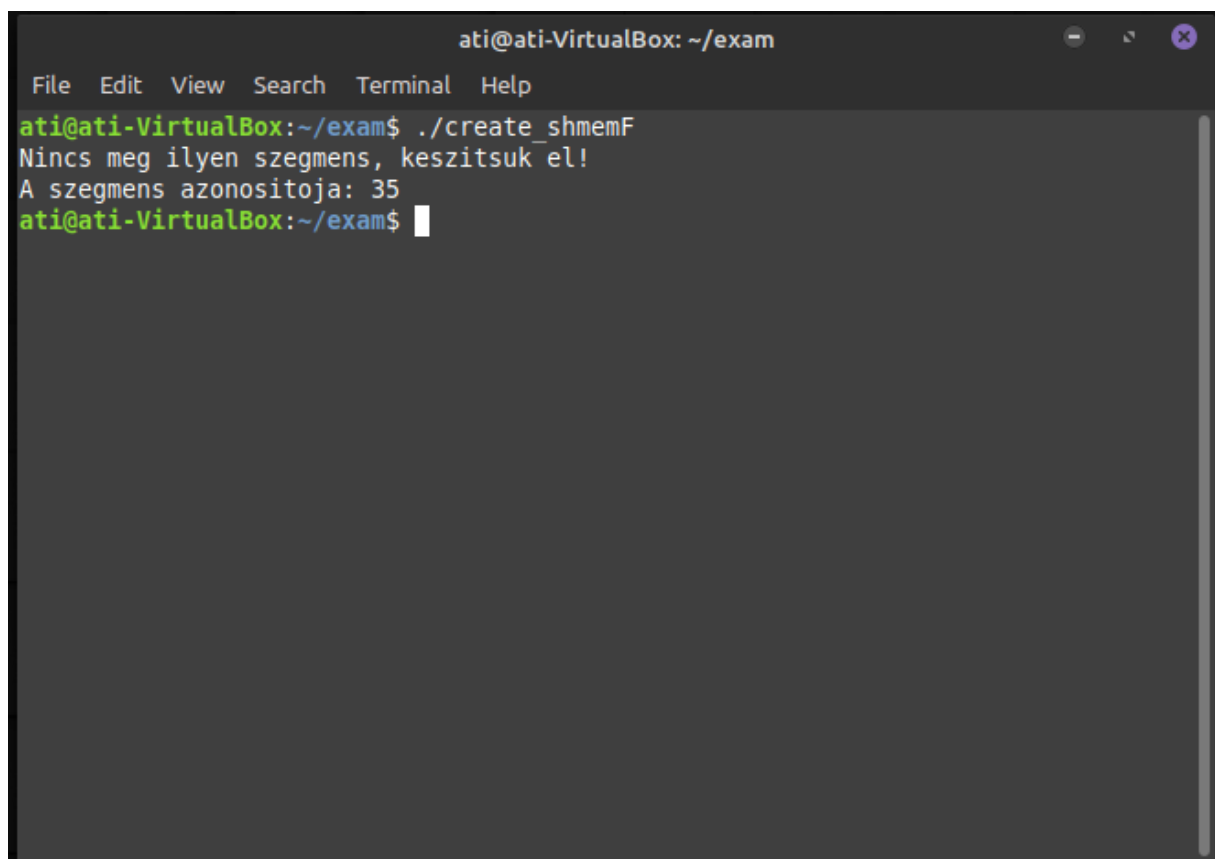
Hegedűs Attila László BSc
Mérnök-informatikus
levelező
D2OVJ9

Miskolc, 2021

A feladat leírása: Írjon C nyelvű programokat, ami létrehoz egy osztott memória szegmenst, az egyik program ír bele és vár pár másodpercet bináris szemafor segítségével védi az írást, a másik program pedig kiolvas belőle.

A feladat elkészítésének lépései: A feladatom elkészítése során a termelő és fogyasztó probléma megoldására koncentráltam. A termelőm egy writerF.c nevű program, a fogyasztóm egy readerF.c nevű program a raktárat, osztott memória szegmenst pedig egy create_shmF.c program készíti el. Az osztott memória szegmens készítése során használok többek között az shmctl(), shmget() rendszerhívásokat. Konstans változóként használok kulcsként az 123456L sztringet, illetve a memória méretét, melyet 512byte-ban határozok meg. A futtatás után kiírja a szegmens azonosítóját is. A reader.c program létrehoz két POSIX bináris szemafor az írás és olvasás koordinálására, a sem_open() rendszerhívással, csatlakozik az osztott memória szegmenshez shmctl() rendszerhívással, várakozik, hogy a writer program „billentsen” a termelő szemaforján, majd az osztott memória szegmensből olvas és kiírja az olvasás eredményét. A writer.c program csatlakozik az osztott memória szegmenshez és a reader által létrehozott szemaforokhoz, majd 5-ször ír bele az osztott memória szegmensbe a strncpy() paranccsal, miközben a szemaforokat dekrementálja és inkrementálja. Az osztott memóriába a terminálban megadott szöveget írja bele. A programok a sem_close hívással törlik a szemaforokat. A delete_shmemF.c törli az osztott memória szegmenst az shmctl() rendszerhívással, ami az IPC_RMID flaget tartalmazza.

A futtatás eredménye: Először elkészítjük az osztott memória szegmenst.



```
ati@ati-VirtualBox: ~/exam
File Edit View Search Terminal Help
ati@ati-VirtualBox:~/exam$ ./create_shmemF
Nincs meg ilyen szegmens, készítsuk el!
A szegmens azonosítója: 35
ati@ati-VirtualBox:~/exam$
```

Elindítjuk a readerF-et, ami a writerre várakozik.

```
ati@ati-VirtualBox: ~/exam
File Edit View Search Terminal Help
ati@ati-VirtualBox:~/exam$ ./create_shmemF
Nincs meg ilyen szegmens, készítsuk el!
A szegmens azonosítója: 35
ati@ati-VirtualBox:~/exam$ ./readerF
█
```

A writer beleír ötször az osztott memóriába, a reader pedig képes mindet kiolvasni.

```
ati@ati-VirtualBox: ~/exam
File Edit View Search Terminal Help
ati@ati-VirtualBox:~/exam$ ./create_shmemF
Nincs meg ilyen szegmens, készítsuk el!
A szegmens azonosítója: 35
ati@ati-VirtualBox:~/exam$ ./readerF
Olvasom: D2ovj9
Olvasom: D2ovj9
Olvasom: D2ovj9
Olvasom: D2ovj9
Olvasom: D2ovj9
^C
ati@ati-VirtualBox:~/exam$

ati@ati-VirtualBox: ~/exam
File Edit View Search Terminal Help
ati@ati-VirtualBox:~/exam$ cd exam/
ati@ati-VirtualBox:~/exam$ ./writerF D2ovj9
Beleírok
Beleírtam: D2ovj9
ati@ati-VirtualBox:~/exam$ █
```

A delete_shmemF.c programmal pedig töröljük az osztott memória szegmenst.

```
ati@ati-VirtualBox: ~/exam
File Edit View Search Terminal Help
ati@ati-VirtualBox:~/exam$ ipcs -m

----- Shared Memory Segments -----
key          shmid      owner      perms      bytes      nattch     status
0x00000000    5          ati        600        16384      1          dest
0x00000000    8          ati        600        67108864   2          dest
0x00000000    13         ati        600        7864320    2          dest
0x00000000    16         ati        600        7864320    2          dest
0x00000000    19         ati        600        4194304    2          dest
0x00000000    20         ati        600        7372800    2          dest
0x00000000    23         ati        600        524288     2          dest
0x00000000    24         ati        600        524288     2          dest
0x00000000    30         ati        600        524288     2          dest
0x00000000    34         ati        600        524288     2          dest
0x00000000    39         ati        600        524288     2          dest
0x00000000    40         ati        600        16777216   2          dest
0x00000000    41         ati        600        7372800    2          dest
0x00000000    60         ati        600        4194304    2          dest
0x00000000    61         ati        600        1441792    2          dest
0x0001e240    62         ati        666        512        0          dest

ati@ati-VirtualBox:~/exam$
```

```
ati@ati-VirtualBox: ~/exam
File Edit View Search Terminal Help
ati@ati-VirtualBox:~/exam$ ./delete_shmemF
ati@ati-VirtualBox:~/exam$ ipcs -m

----- Shared Memory Segments -----
key          shmid      owner      perms      bytes      nattch     status
0x00000000    5          ati        600        16384      1          dest
0x00000000    8          ati        600        67108864   2          dest
0x00000000    13         ati        600        7864320    2          dest
0x00000000    16         ati        600        7864320    2          dest
0x00000000    19         ati        600        4194304    2          dest
0x00000000    20         ati        600        7372800    2          dest
0x00000000    23         ati        600        524288     2          dest
0x00000000    24         ati        600        524288     2          dest
0x00000000    30         ati        600        524288     2          dest
0x00000000    34         ati        600        524288     2          dest
0x00000000    39         ati        600        524288     2          dest
0x00000000    40         ati        600        16777216   2          dest
0x00000000    41         ati        600        7372800    2          dest
0x00000000    60         ati        600        4194304    2          dest
0x00000000    61         ati        600        1441792    2          dest

ati@ati-VirtualBox:~/exam$
```

A forráskód:

```
< > create_shmemF.c x writerF.c x readerF.c x delete_shmemF.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <sys/ipc.h>
5  #include <sys/shm.h>
6
7  #define SHMKEY 123456L
8  #define SIZE 512
9
10 int main(int argc, char const *argv[])
11 {
12
13     int shmid;
14     key_t key;
15     int shmflg;
16
17     key = SHMKEY;
18     shmflg = 0;
19
20     if((shmid = shmget(key, SIZE, shmflg)) < 0){
21         printf("Nincs meg ilyen szegmens, keszitsuk el!\n");
22         shmflg = 0666 | IPC_CREAT;
23         if((shmid = shmget(key, SIZE, shmflg)) < 0){
24             perror("Az shmget system-call sikertelen!\n");
25             exit(-1);
26         }
27     } else {
28         printf("A szegmens mar letezik!\n");
29     }
30
31     printf("A szegmens azonositoja: %d\n", shmid);
32
33     return 0;
34 }
```

```
< > create_shmemF.c x writerF.c x readerF.c x delete_shmemF.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/types.h>
5  #include <sys/ipc.h>
6  #include <sys/shm.h>
7  #include <fcntl.h>
8  #include <semaphore.h>
9
10 #define SHMKEY 123456L
11 #define SIZE 512
12 #define SEM_PRODUCER "/producer"
13 #define SEM_CONSUMER "/consumer"
14
15 int main(int argc, char const *argv[])
16 {
17     int shmid, shmflg;
18     key_t key = SHMKEY;
19     shmflg = 0;
20
21     char *data;
22
23     sem_t *sem_producer = sem_open(SEM_PRODUCER, O_EXCL);
24     sem_t *sem_consumer = sem_open(SEM_CONSUMER, O_EXCL);
25
26
27     if((shmid = shmget(key, SIZE, shmflg)) < 0){
28         printf("Nincs meg a szegmens!\n");
29     }
30
31     data = shmat(shmid, NULL, 0);
32     printf("Beleirok\n");
33     for(int i = 0; i < 5; i++){
34         sem_wait(sem_consumer);
35         strncpy(data, argv[1], 10);
36         sem_post(sem_producer);
37     }
38
39     printf("Beleirtam: %s\n", argv[1]);
40
41     sem_close(sem_producer);
42     sem_close(sem_consumer);
43
44     return 0;
45 }
46
```

```

< > create_shmemF.c x writerF.c x readerF.c x delete_shmemF
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <stdbool.h>
5  #include <sys/stat.h>
6  #include <sys/ipc.h>
7  #include <sys/shm.h>
8  #include <fcntl.h>
9  #include <semaphore.h>
10
11  #define SHMKEY 123456L
12  #define SIZE 512
13  #define SEM_PRODUCER "/producer"
14  #define SEM_CONSUMER "/consumer"
15
16
17  int main(int argc, char const *argv[])
18  {
19      int shmid;
20      key_t key;
21      int shmflg;
22      char *data;
23
24
25      key = SHMKEY;
26      shmflg = 0;
27
28      sem_unlink(SEM_PRODUCER);
29      sem_unlink(SEM_CONSUMER);
30
31      sem_t *sem_producer = sem_open(SEM_PRODUCER, O_CREAT, 0660, 0);
32      if(sem_producer == SEM_FAILED){
33          perror("semopen/producer");
34      }
35      sem_t *sem_consumer = sem_open(SEM_CONSUMER, O_CREAT, 0660, 1);
36      if(sem_producer == SEM_FAILED){
37          perror("semopen/consumer");
38      }
39      if((shmid = shmget(key, SIZE, shmflg)) < 0){
40          printf("Nincs meg a szegmens!\n");
41      }
42
43      data = shmat(shmid, NULL, 0);
44
45      while (1){
46          sem_wait(sem_producer);
47          if(strlen(data) > 0){
48              printf("Olvasom: %s\n", data);
49          }
50          sem_post(sem_consumer);
51
52      }
53
54      sem_close(sem_consumer);
55      sem_close(sem_producer);
56
57
58      return 0;
59  }
60

```

```
< > create_shmemF.c x writerF.c x readerF.c x delete_shmemF.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <sys/ipc.h>
5  #include <sys/shm.h>
6
7  #define SHMKEY 123456L
8  #define SIZE 512
9
10 int main(int argc, char const *argv[])
11 {
12
13     int shmid;
14     key_t key;
15     int shmflg;
16
17     key = SHMKEY;
18     shmflg = 0;
19
20     shmid = shmget(key, SIZE, shmflg);
21     shmctl(shmid, IPC_RMID, NULL);
22
23     return 0;
24 }
```