

```

1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/shm.h>
4 #include <stdio.h>
5 #define SHMSZ 27
6
7 void server() {
8     char c;
9     int shmid;
10    key_t key = 5678;
11    char *shm, *s;
12
13    // Creer le segment
14    shmid = shmget(key, SHMSZ, IPC_CREAT | 0666);
15    if(shmid < 0) { perror("shmget"); exit(1); }
16
17    // Attacher le segment
18    shm = shmat(shmid, NULL, 0);
19    if (shm == (char *) -1) { perror("shmat"); exit(1); }
20
21    // Mettre quelques choses dans la memoire pour l'autre processus
22    s = shm;
23    for (c = 'a'; c <= 'z'; c++) *s++ = c;
24    *s = NULL;
25
26    // On attend que le client lise en mettant en premier caractere '*'
27    while (*shm != '*') sleep(1);
28
29    shmdt((void*) shm);
30    shmctl(shmid, IPC_RMID, NULL);
31    exit(0);
32 }
33
34 void client() {
35     int shmid;
36     char *shm, *s;
37
38     // Obtenir le segment "5678" cree par le serveur
39     key_t key = 5678;
40     shmid = shmget(key, SHMSZ, 0666);
41     if(shmid < 0) { perror("shmget"); exit(1); }
42
43     // Attacher le segment a notre espace de donnees
44     shm = shmat(shmid, NULL, 0);
45     if (shm == (char*) -1) { perror("shmat"); exit(1); }
46
47     // Lire ce que le serveur a mis dans la memoire
48     for (s = shm; *s != NULL; s++) putchar(*s);
49     putchar('\n');
50
51     // Changez le premier caractere du segment en '*' pour indiquer la lecture du segment
52     *shm = '*';
53     exit(0);
54 }

```

```

1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/shm.h>
4 #include <stdio.h>
5 #define SHMSZ 27
6
7 int main(int, char**) {
8     int shmid;
9     char *shm, *s;
10
11    // Obtenir le segment "5678" cree par le serveur
12    key_t key = 5678;
13    shmid = shmget(key, SHMSZ, 0666);
14    if(shmid < 0) { perror("shmget"); exit(1); }
15
16    // Attacher le segment a notre espace de donnees
17    shm = shmat(shmid, NULL, 0);
18    if (shm == (char*) -1) { perror("shmat"); exit(1); }
19
20    // Lire ce que le serveur a mis dans la memoire
21    for (s = shm; *s != NULL; s++) putchar(*s);
22    putchar('\n');
23
24    // Changez le premier caractere du segment en '*' pour indiquer la lecture du segment
25    *shm = '*';
26    exit(0);
27 }
28

```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6 #include <unistd.h>
7 #include <sys/mman.h>
8
9 int main(int argc, char* argv[]) {
10     int i= 0, fd = open(filename, O_RDWR, 0666);
11     char* filename = "file.txt";
12     struct stat st;
13
14     stat(filename, &st);
15     long fileSize = st.st_size;
16     char* fileMap = (char*) mmap(NULL, fileSize, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
17
18     if(fileMap == (char*) MAP_FAILED) { perror("mmap"); exit(1); }
19     close(fd);
20
21     while(i < fileSize/2) {
22         char c = fileMap[i];
23         fileMap[i] = fileMap[fileSize - i-1];
24         fileMap[fileSize - i-1] = c;
25         i++;
26     }
27
28     printf("%s\n", fileMap);
29     munmap((void*) fileMap, fileSize);
30     return 0;
31 }
32

```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 #include <sys/types.h>
5 #include <sys/stat.h>
6 #include <unistd.h>
7 #include <sys/mman.h>
8 #define FILESIZE 10
9
10 int main(int argc, char* argv[]) {
11     int i= 0, fd = open("titi.dat", O_RDWR, 0666);
12
13     int* fileMap = (int*) mmap(NULL, FILESIZE*sizeof(int), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
14
15     if(fileMap == (int*) MAP_FAILED) { perror("mmap"); exit(1); }
16     close(fd);
17
18     while(1) {
19         scanf("%d", &i);
20         if(i == 99) break;
21         if(i < 10 && i >= 0) fileMap[i] = fileMap[i]+1;
22     }
23
24     munmap((void*) fileMap, FILESIZE*sizeof(int));
25     return 0;
26 }
27

```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <fcntl.h>
4  #include <sys/types.h>
5  #include <sys/stat.h>
6  #include <unistd.h>
7  #include <sys/mman.h>
8  #define FILESIZE 10
9
10 int main(int argc, char* argv[]) {
11     int i= 0, fd = open("titi.dat", O_RDONLY, 0666);
12
13     int* fileMap = (int*) mmap(NULL, FILESIZE*sizeof(int), PROT_READ, MAP_PRIVATE, fd, 0);
14
15     if(fileMap == (int*) MAP_FAILED) { perror("mmap"); exit(1); }
16     close(fd);
17
18     while(1) {
19         scanf("%d", &i);
20         if(i == 99) break;
21         for(int j= 0; j< 10; j++) printf("\t%d\n", fileMap[j]);
22     }
23
24     munmap((void*) fileMap, FILESIZE*sizeof(int));
25     return 0;
26 }
27

```

```

1  #include <stdio.h>
2  #include <fcntl.h>
3  #include <sys/types.h>
4  #include <sys/shm.h>
5  #include <sys/wait.h>
6  #include <unistd.h>
7
8  #define READSIZE 1024
9
10 int copyFile(int f1, int f2) {
11     int totalByteRead= 0, lastByteRead= READSIZE;
12     char buffer[READSIZE];
13
14     while(lastByteRead == READSIZE) {
15         lastByteRead = read(f1, buffer, READSIZE);
16         write(f2, buffer, lastByteRead);
17         totalByteRead += lastByteRead;
18     }
19
20     return totalByteRead;
21 }
22
23 int main(int argc, char* argv[]) {
24     int file;
25     id_t id = shmget(IPC_PRIVATE, sizeof(int), 0666);
26     int* shmArea = (int*) shmat(id, NULL, 0);
27
28     pid_t chld = fork();
29
30     if(chld == 0) file = open((argc > 2) ? argv[2] : "file2", O_RDONLY);
31     else file = open((argc > 1) ? argv[1] : "file1", O_RDONLY);
32
33     int byteCopied = copyFile(file, fileno(stdout));
34     close(file);
35
36     if(chld == 0) *shmArea = byteCopied;
37     else
38     {
39         wait(NULL);
40         printf("Bytes copied: %d & %d\n", byteCopied, *shmArea);
41
42         shmdt((void*) shmArea);
43         shmctl(id, IPC_RMID, NULL);
44     }
45
46     return 0;
47 }

```