JEGYZŐKÖNYV

Operációs rendszerek BSc

2021. tavasz féléves feladat

Készítette: **Kocsis Katalin**
Neptunkód: **WGOWUG**

A feladat leírása:

12. Irjon C nyelvu programokat, ami
hozzon letre egy osztott memoria szegmenst
ebbe olvassa be egy nagy file tartalmat
egy masik program pedig olvass ki az osztott mem szegmenbol
es irja bele egy masik fileba
vegul szuntesse meg az shm szegmenst

A feladat elkészítésének lépései:

I. A feladatom az osztott memóriákról szól. Az osztott memória lefoglal egy memóriablokkot, és arra különböző processzek csatlakozhatnak. Ebben az osztott memóriaszegmensbe lehet adatokat tárolni, onnan adatot kiolvasni.
A feladat teljesítéséhez két részre osztottam a feladatot, ezáltal két darab C nyelvű programot írtam meg, amik shma.c és shmb.c néven vannak elmentve. Létre kell még hozni egy .txt nagyméretű fájlt is, amit az shma.c fog beolvasni, és az shmb.c pedig kiolvassa azt. Jelen esetben én nagyFajl.txt nevet adtam neki, amiben "Operációs Rendszerek BSc 2017/2018 II. félidő Memóriamenedzselés Dr. Vincze Dávid" pdf-ből kivett részletet (1-10. oldalig) tartalmazza.

a. Az shma.c program feladata:

1. file-t beolvas

2. méretének megfelelő memória szegmenst gyárt

3. beírja a szegmensbe a file tartalmát

b. Az shmb.c program feladata:

1. megnézi van-e valami a memória szegmensben

2. ha van, akkor kiolvassa onnan és beírja egy file-ba

3. megszünteti a memória szegmenst

A memória szegmensnek van egy id-ja, azt írja ki az shma.c, majd ezzel az id-vel meg tudjuk hívni az shmb.c programot.

II. Nézzük meg először az shma.c programot lépésről lépésre.

1. Dekralálok:
   - osztott memóriának id azonosítót
   - flaget a jellemzőkhöz (osztott memóriának, tulajdonságot tud tárolni, ki tud hozzáférni, memóriának a hozzáférési engedélyét szerkeszti)
   - size (méretét 512-re inicializálom, annyi karaktert tud beolvasni a lezáró 0val együtt)

- char típusú tömböt (eltároljuk a fájlban beolvasott szövegek, le kell vonni az integer méretét, az osztottmemóriába így kell tárolni a szöveget, és tároljuk h mi a mérete, hossza a szövegnek (integer ehhez kell), 4 bájtba nem tudok szöveget rakni, 508 bájtnyi szöveget tudok tárolni)

nevű változókat.

2. Definiálok egy kulcsot az shm-emhez. (#define SHMKEY 123456L, key_t key;). Majd kell a fájlnak egy pointer, amibe beolvasom a fájlomat. (FILE * filePointer;)

3. A struktúrát leképezzük szegmensre.

```
struct segmStruct {
        int  textLength;  \\ kell egy szöveghossz, milyen hosszú lesz a szövegünk ezt jelzi
        char text[512-sizeof(int)]; \\ szövegnek kell egy tömb, betároljuk
    } *segm;        \\ ezzel meghatározok egy struktúra változót, memória címet jelen, a szöveg és hosszára mutat
```

4. Beolvasás a fájlból és tartalmának kiíratása. Szükségesek hozzá: filepointer, fopen, fclose.

```
filePointer = fopen("nagyFajl.txt", "r"); \\ megnyitja a fájlt, a „r” = read, olvasásra nyitja meg
    printf("Content of the read file:\n");
    while(fgets(readData, 512-sizeof(int), filePointer) != NULL) {
        printf("%s", readData);
    }
    fclose(filePointer);
    printf("Data successfully read from file nagyFajl.txt\n");
     printf("The file is now closed.");
```

Fájl karakterein végig megyünk (while ciklussal), fgets fájból karaktereket olvas be, addig megy amíg nem éri el az utolsó karaktert , majd kiírjuk a file tartalmát.
Fclose-val bezárjuk a fájlt.

5. Megnézzük, hogy van SHMKEY-es, size méretű szegmens.
    - Define-vel globális változót készítettünk. (#define SHMKEY 123456L). A key változóba beleteszem az előre definiált kulcs változót, kulcsot vizsgálunk h létezik e.

```
 key = SHMKEY;
shmflg = 0;
```

    - shmid -> ebbe a változóba rakja bele az shmget
    - shmget-> Az osztottmemória adatait lekérdezi, adatait megkapjuk. Kulcsra, méretére, flagre szüksége van.
    - shmget megadott paraméter alapján megvizsgálja h az osztott memóriába van e ilyen paraméterű osztott memória, ha van akkor visszatér adott id-vel, ha nem talál, nincs osztott memória akkor -1-el tér vissza, és akkor megy a printf-re.

```
if ((shmid = shmget(key, size, shmflg)) < 0) {
        printf("\n Such a segment doesn't exist yet! Let's create it!");
```

```
shmflg = 00666 | IPC_CREAT;     /* a 00666 a szegmens hozzáférési
```
engedélyeit állítja be, az IPC_CREAT létrehozza a szegmenst */
megint lekérdezi h van e olyan szegmens (amit előbb létrehoztam azt ellenőrzi), ha van system call hibával tér vissza, ha nincs akkor a segment létezik. Kiírjuk az osztott memória idjét, ezzel tudjuk meghívni az shmb.c-t.

```
                if ((shmid=shmget(key, size, shmflg)) < 0) {
        perror("\nThe shmget system-call failed!");
        exit(-1);
            }
    } else printf("The segment already exists!\n!");

    printf("ID of the shared memory: %d \n", shmid);
```

6. Attach: hogy tudjak a szegmensbe írni, rácsatlakozok ezzel a processzel, és detach: lecsatlakozunk az osztott memóriáról.

- Osztottmemóriához csatlakozunk. Shma függvényt hívunk ezzel képezzük le struktúrára, így hívjuk meg a struktúrát, és így teszi bele a struktúrába a szöveget, és hogy milyen hosszú a szöveg.
- Itt a NULL azt jelenti, hogy az OS-re bízom, hogy milyen címtartományt használjon.

```
shmflg = 00666 | SHM_RND;
    segm = (struct segmStruct *)shmat(shmid, NULL, shmflg);
```

- Megnézzük hogy mit rakott bele a struktúrába, ha -1et (nem rakott bele adatot) akkor hiba történik, akkor sikertelen csatolást kiírjuk , és kilép.

```
if (segm == (void *)-1) {
            perror("Attach failed!\n");
            exit (-1);
    }
```

```
    strcpy(segm->szoveg, readData);  \\ meghívjuk a string másolót, megadom a
```
struktúra változót, és text adatágát használom fel. A readData-ba másolja a szöveget

```
/* Detach, megadjuk a struktúrát hogy arról menjen le*/
    shmdt(segm);
```

7. Kiíratom az osztott memóriaszegmens id-ját, és kilépünk.
   printf("ID of the shared memory segment, call the other process with this: %d\n", shmid);

```
    exit(0);
```

III. Most nézzük meg az shmb.c programot lépésről lépésre.

1. shma.c lefutott beleírtuk az adatot, és másik ez a processzel rácsatlakozunk, tudnia kell hogy melyikre kell rácsatlakozni, ezért a main függvénynek 2 paramétert adunk. Az argc paraméterek számát tárolja, hány db paraméterrel hívtam meg az shmb.c –t (jelen esteben ez 1db, ami az shmid), ez fog eltárolódni. Az argv azokat paramétert tárolja, amivel meghívom a programot, első elem a neve a programnak amit hívok (shmb), a második elem az shmid. Azaz az argv 2 paramétert fog átadni.

- Eltároljuk az argv 2. elemét (shmid), és atoivel konvertáljuk a betűt int-té

```
int shmid = atoi(argv[1]);
```
2 paramétert kapott az argv? Leellenőrizzük, ha nem akkor else ágra ugrik, és kiírja, hogy túl sok vagy túl kevés a paraméter.
```
if (argc == 2) {
        int shmid = atoi(argv[1]);
        int shmflg; \\amivel az osztott memóriához csatlakozok
        FILE * filePointer; \\ ez a pointer szövöget tárol
```

2. A struktúrát leképezzük szegmensre.

```
struct segmStruct {
        int  textLength;  \\ kell egy szöveghossz, milyen hosszú lesz a szövegünk ezt jelzi
        char text[512-sizeof(int)]; \\ szövegnek kell egy tömb, betároljuk
    } *segm;        \\ ezzel meghatározok egy struktúra változót, memória címet jelen, a szöveg és hosszára mutat
```

3. Attach: hogy tudjak a szegmensbe írni, rácsatlakozok ezzel a processzel, és detach: lecsatlakozunk az osztott memóriáról.

- Osztottmemóriához csatlakozunk. Shma függvényt hívunk ezzel képezzük le struktúrára, így hívjuk meg a struktúrát, és így teszi bele a struktúrába a szöveget, és hogy milyen hosszú a szöveg.
- Itt a NULL azt jelenti, hogy az OS-re bízom, hogy milyen címtartományt használjon.

```
shmflg = 00666 | SHM_RND;
    segm = (struct segmStruct *)shmat(shmid, NULL, shmflg);
```

- Megnézzük hogy mit rakott bele a struktúrába, ha -1et (nem rakott bele adatot) akkor hiba történik, akkor sikertelen csatolást kiírjuk , és kilép.

```
if (segm == (void *)-1) {
        perror("Attach failed!\n");
        exit (-1);
        }
```

- Itt bemásoljuk a struktúrába a segmens a változóba, strlen-nél az adott szöveg hosszát adja vissza

```
        segm->textLength=strlen(segm->text);
```

- megvizsgáljuk, hogy a szöveghossza nagyobb mint 0 az osztott memóriába, ha nagyobb kiírja a szöveget, ha nem akkor kiírja, hogy nincs az adott memóriába semmi, és hibával lép ki.

```
    if (segm->textLength > 0) {
                printf("\nThe text in the shared memory: %s (on %d length\n)",segm->text,segm->textLength);
            } else {
                printf("There is no text in the shared memory!");
                exit(-1);
            }
```

- belerakjuk a fájl tartalmát, meghívjuk a fájl tartalmát , utána megpróbálunk írni a fájlba

```
        filePointer = fopen("masikFajl.txt", "w");
```

- ha memóriába nincs semmi, akkor nem sikeres az írás, ha van valami, akkor sikeres az írás.

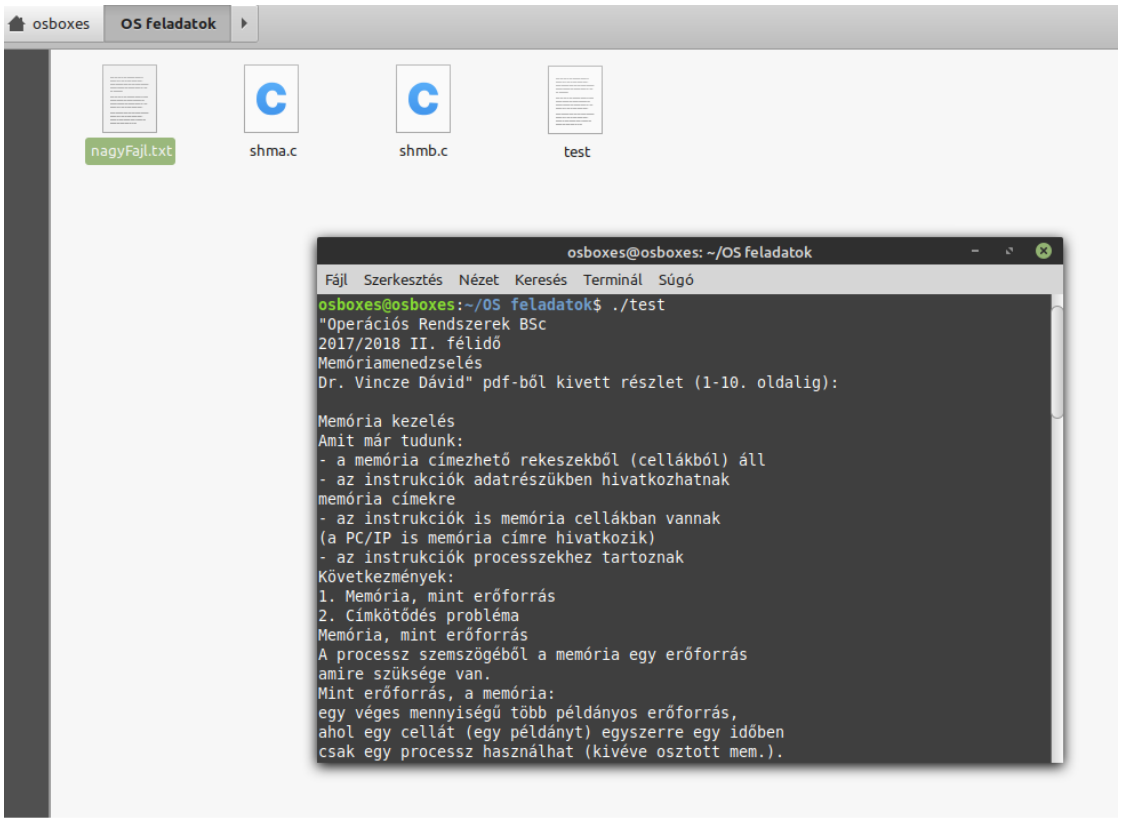
```
                    if (fputs(segm->text, filePointer) == EOF) {
                            printf("Unsuccesful writing to file!\n");
                    } else {
                            printf("Successful writing to file!\n");
                    }
            fclose(filePointer);  \\ Fclose-val bezárjuk a fájlt.

        /* Detach, megadjuk a struktúrát hogy arról menjen le*/
        shmdt(segm);
```

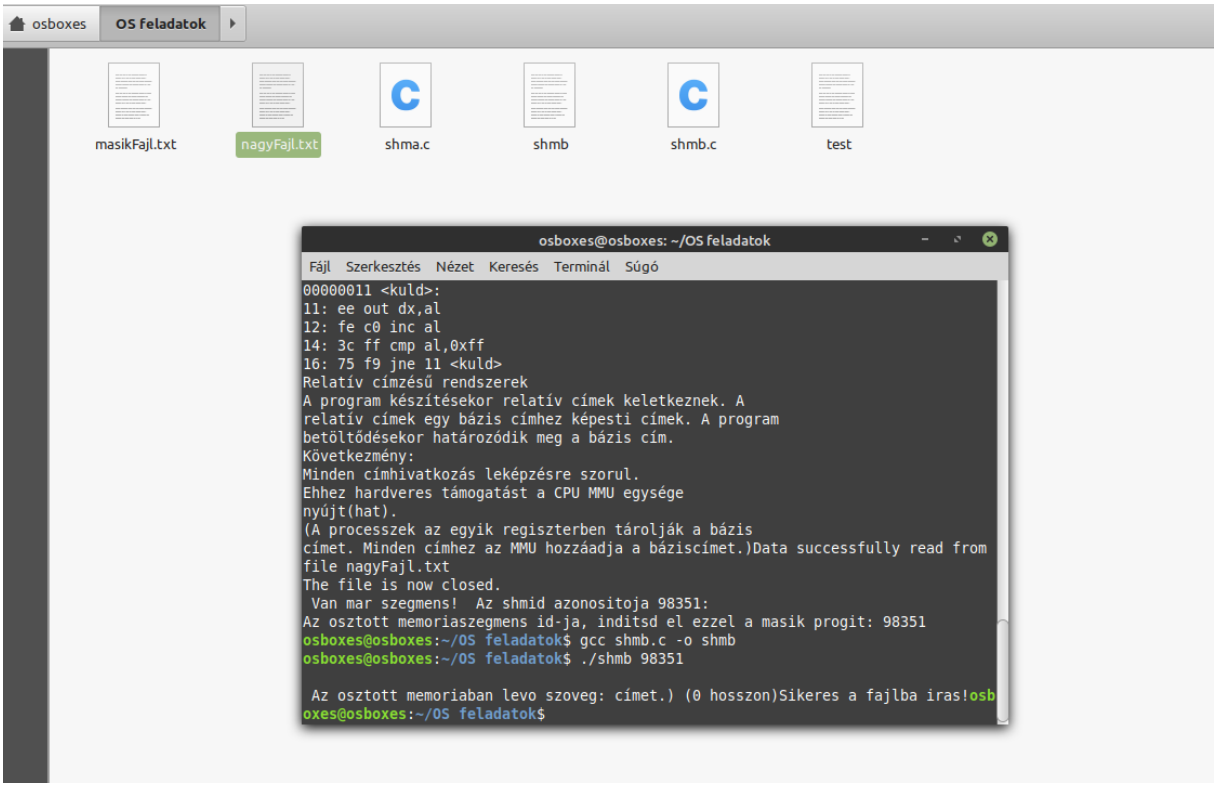
4. Memória „elpusztítása”
   Kitöröljük az shm id-t, töröljük az osztott memóriát.

```
   shmctl(shmid,IPC_RMID,NULL);

   exit(0); \\kilépünk

   } else {printf("Túl sok vagy túl kevés paramétert adott meg!");
            exit (-1);
   }
   }
```

Az 1. pontban említett else ág. (2 paramétert kapott az argv? Leellenőrizzük, ha nem akkor else ágra ugrik, és kiírja, hogy túl sok vagy túl kevés a paraméter.)

A futtatás eredménye:





Az shma.c –t lefuttatva beolvasta a nagyFajl.txt tartalmát, és kiírta. Majd ellátta az osztott memória szegmenst id-val, és ezt ki is írta, hogy az shmb.c –t le tudjuk futtatni vele. Az shmb.c – t az id azonosítóval együtt lefuttatjuk. Az shmb.c az osztott memória szegmensből kiolvasta az id segítségével a nagyFajl.txt tartalmát, majd beleírta egy másik fájlba. Végül megszűnt az shm szegmens.