

Լաբորատոր աշխատանք 6

Հիշողության արտապատկերում (Memory Mapping)

`mmap()` համակարգային կանչը պրոցեսի վիրտուալ հիշողության տարածքում ստեղծում է նոր արտապատկերում: Ֆունկցիայի պրոտոտիպը.

```
#include <sys/mman.h>
void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
```

- **addr** - այս արգումենտը սահմանում է վիրտուալ հասցեն, որտեղ պետք է տեղակայվի արտապատկերում: Եթե այս արգումենտը փոխանցվում է որպես `NULL`, հասցեն ընտրվում է ՕՐ միջուկի կողմից: Սա արտապատկերում ստեղծելու նախընտրելի մեթոդն է: Եթե այս արգումենտը `NULL` չէ, ապա ՕՐ միջուկը կլորացնում է փոխանցված արժեքը միևնույն մոտակա էջի սահմանը: 2 դեպքում էլ ՕՐ միջուկն ընտրում է այնպիսի հասցե, որը չի հատվում գոյություն ունեցող արտապատկերումների հետ: Եթե `flags` արգումենտով սահմանված է `MAP_FIXED` դրոշակը, ապա `addr` արգումենտով փոխանցված հասցեն պետք է լինի համակարգային էջի բազմապատիկ: Զամակարգային էջի չափը կարելի է ստանալ հետևյալ հրամանի միջոցով. `getconf PAGE_SIZE`
- **length** արգումենտը սահմանում է արտապատկերման չափը՝ բայթերով: Չնայած, որ պարտադիր չէ, որ այս արգումենտը լինի համակարգային էջի բազմապատիկ, ՕՐ միջուկը արտապատկերումները ստեղծում է այդ չափի միավորներով այնպես, որ երկարությունը կլորացվում է դեպի վեր՝ միևնույն մոտակա էջի բազմապատիկը:
- **prot** արգումենտը սահմանում է արտապատկերման պաշտպանության բիթերը: Այն կարող է ընդունել `PROT_NONE` արժեքը կամ թվարկված այլ դրոշակների համադրություն.
- `PROT_NONE` – հատվածը հասանելի չէ,
- `PROT_READ` – հատվածի պարունակությունը կարելի է կարդալ
- `PROT_WRITE` – հատվածի պարունակությունը կարող է փոփոխվել
- `PROT_EXEC` – հատվածի պարունակությունը կարող է կատարվել:
- **flags** արգումենտը դեկավարում է արտապատկերման գործողությունը: Այն պետք է պարունակի հետևյալ դրոշակներից ճիշտ մեկը.
- `MAP_PRIVATE` – ստեղծում է մասնավոր արտապատկերում: Այդ հատվածում կատարված փոփոխությունները տեսանելի չեն նույն արտապատկերումը

օգտագործող այլ պրոցեսների համար: Ֆայլային արտապատկերման դեպքում փոփոխությունները չեն գրանցվում ֆայլում:

- MAP_SHARED – ստեղծում է ընդհանուր արտապատկերում: Այդ հատվածում կատարված փոփոխությունները տեսանելի են նույն արտապատկերումը MAP_SHARED ատրիբուտով օգտագործող այլ պրոցեսների համար: Ֆայլային արտապատկերման դեպքում փոփոխությունները գրանցվում են ֆայլում:

Մյուս 2 արգումենտները՝ **fd** և **offset**, կիրառվում են ֆայլային արտապատկերումների դեպքում և անտեսվում են անանուն արտապատկերումների դեպքում:

- **fd** արգումենտը ֆայլային դեսկրիպտոր է, որը նույնականացնում է արտապատկերվող ֆայլը:
- **offset** արգումենտը սահմանում է ֆայլում արտապատկերման սկիզբը և պետք է լինի համակարգային էջի բազմապատիկ: Ամբողջ ֆայլն արտապատկերելու համար պետք է offset-ը փոխանցել որպես 0, իսկ length-ը՝ ֆայլի չափով:

Դաջող կանչի դեպքում mmap() ֆունկցիան վերադարձնում է նոր արտապատկերման սկզբնական հասցեն: Սխալի դեպքում վերադարձնում է **MAP_FAILED** արժեքը:

Արտապատկերման ստեղծումը

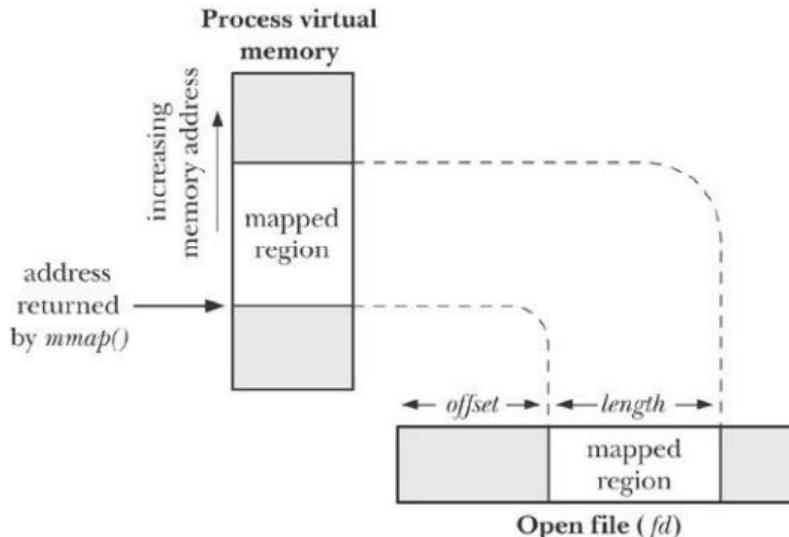
mcat ծրագրում ներկայացված է cat հրամանի պարզ տարրերակի իրագործումը: Այն արտապատկերում է ֆայլը, որի անունը փոխանցվել է ծրագրին որպես հրամանային տողի արգումենտ, և գրում է դրա պարունակությունը stdout հոսքի մեջ: Ծրագիրը կատարելու համար անհրաժեշտ քայլերն են.

- Ստեղծել ծրագրի կատարվող ֆայլը.
gcc mcat.c -o mcat
- Ստեղծել որոշակի պարունակությամբ ֆայլ, որը պետք է արտապատկերվի ծրագրի կողմից.
echo "Hello world" > file.txt
- Կատարել ծրագիրը՝ որպես հրամանային տողի արգումենտ փոխանցելով ստեղծված ֆայլի անունը.
./mcat file.txt

ֆայլային արտապատկերում

Ֆայլային արտապատկերում ստեղծելու համար անհրաժեշտ է իրականացնել հետևյալ քայլերը.

1. Ստեղծել ֆայլային դեսկրիպտոր (օրինակ՝ `open()` ֆունկցիայի կանչով),
2. Փոխանցել ստեղծված ֆայլային դեսկրիպտորը որպես `mmap()` ֆունկցիայի `fd` արգումենտ:



Նկ. 1 ֆայլային արտապատկերում (file mapping)

`mmap` ծրագիրը ստեղծում է ընդհանուր ֆայլային արտապատկերում: Այն արտապատկերում է ֆայլը, որի անունը փոխանցվել է ծրագրին որպես հրամանային տողի առաջին արգումենտ, և արտածում է արտապատկերված հատվածում գտնվող պարունակությունը: Եթե ծրագրին հրամանային տողով փոխանցվում է ևս մեկ արգումենտ, ապա այդ արգումենտի արժեքը պատճենվում է ընդհանուր հիշողության հատվածում: Ծրագիրը կատարելու համար անհրաժեշտ քայլերն են.

- Ստեղծել ծրագրի կատարվող ֆայլ.
gcc mmap.c -o mmap
- Ստեղծել ֆայլ, որի պարունակությունը պետք է արտապատկերվի ծրագրի կողմից.
touch mfile.txt
- Քանի որ `touch` հրամանի արդյունքում ստեղծված ֆայլի չափը 0 է, ապա հարկավոր է ֆայլի համար սահմանել որոշակի չափ և սկզբնավորել պարունակությունը 0 արժեքով.
truncate -s 100 mfile.txt

- Կատարել ծրագիրը՝ որպես հրամանային տողի արգումենտներ փոխանցելով ստեղծված ֆայլի անունը և կամայական արժեք, որը պետք է գրանցվի ֆայլում.
`./mmap mfile.txt "Hello world"`

Քանի որ արտապատկերման պարունակությունը սկզբնավորվում է ֆայլից, և արտապատկերման մեջ կատարված փոփոխությունները գրանցվում են ֆայլում, ապա կարելի է ասել, որ այս ծրագիրն իրականացնում է հիշողության արտապատկերմամբ Ն/Ա (memory-mapped I/O) տեխնիկան:

Անանոնս արտապատկերում

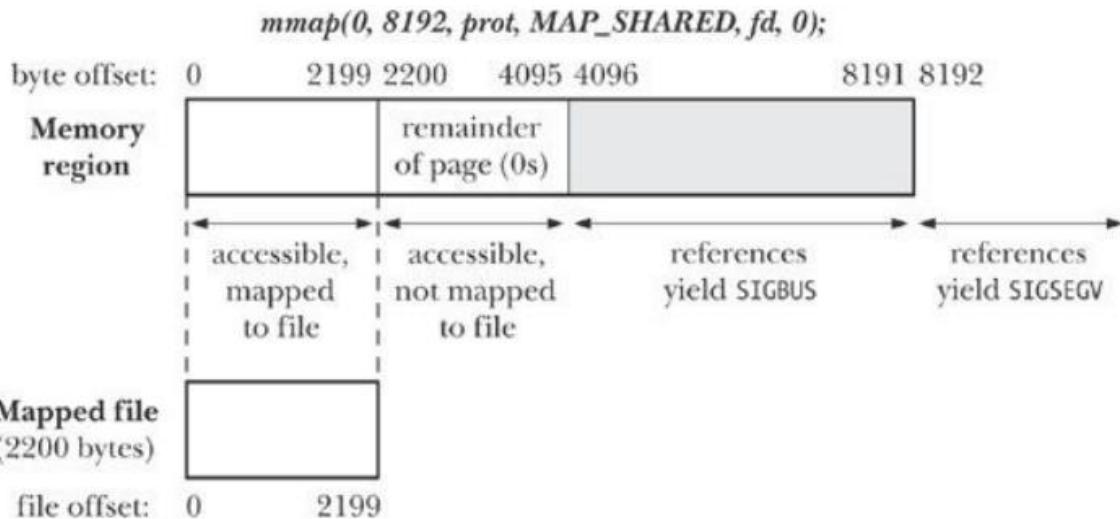
`anon_mmap` ծրագիրը ներկայացնում է անանոնս արտապատկերման կիրառումը ծնող և դուստր պրոցեսների կողմից: Կախված `USE_MAP_ANON` մակրոյի առկայությունից՝ այն արտապատկերումը ստեղծում է `MAP_ANONYMOUS` դրոշակի կամ `/dev/zero` սարքավորման ֆայլի միջոցով:

Միևնույն ժամանակ կանչը ծնող պրոցեսը սկզբնավորում է արտապատկերումը 1 արժեքով: Դրանից հետո դուստր պրոցեսը մեծացնում է այդ արժեքը 1-ով և ավարտվում: Ծնող պրոցեսը սպասում է դուստր պրոցեսի ավարտին և տպում փոփոխված արժեքը: Ծրագիրը կատարելու համար անհրաժեշտ քայլերն են.

- Ստեղծել ծրագրի կատարվող ֆայլը.
`gcc anon_mmap.c -o anon_mmap`
կամ
`gcc anon_mmap.c -D USE_MAP_ANON -o anon_mmap`
- Կատարել ծրագիրը.
`./anon_mmap`

Սահմանային դեպքեր

Նկ. 4-ում պատկերված է իրավիճակ, երբ արտապատկերման չափը գերազանցում է ֆայլի չափը: Քանի որ ֆայլի չափը համակարգային էջի բազմապատիկ չէ, ապա այն կլորացվում է դեպի վեր՝ միևնույն համակարգային էջի մոտակա բազմապատիկը: Բայց այս դեպքում դեպի վեր կլորացված հատվածի բայթերը չեն արտապատկերվում ֆայլից, քանի որ դրանք ֆայլում հասանելի չեն: Փոխարենը, դրանք սկզբնավորվում են 0 արժեքով: Այսուամենայնիվ, այս բայթերը հասանելի են մյուս պրոցեսների համար, որոնք օգտագործում են արտապատկերման տվյալ հատվածը: Այս հատվածում կատարված փոփոխությունները չեն գրանցվում ֆայլում:



Նկ. 2 Հիշողության արտապատկերում, որը գերազանցում է ֆայլի չափը

Եթե արտապատկերման մեջ կան դեպի վեր կլորացված հատվածից հետո եկող բայթեր, ապա դրանց դիմելու դեպքում գեներացվում է **SIGBUS** ազդանշան: Արտապատկերման վերջից դուքս բայթերին դիմելու դեպքում գեներացվում է **SIGSEGV** ազդանշան:

Այս իրավիճակը կարելի է ստեղծել տուր ծրագրում՝ փոխանցելով 0 երկարությամբ ֆայլ:

Արտապատկերման հեռացումը

munmap() համակարգային կանչն իրականացնում է **mmap()** կանչի հակառակ գործողությունը՝ հեռացնելով արտապատկերումը պրոցեսի վիրտուալ հիշողության տարածքից: Ֆունկցիայի կիրառման օրինակը ներկայացված է `anon_mmap` ծրագրում:

Ֆունկցիայի պրոտոտիպը.

```
#include <sys/mman.h>
int munmap(void *addr, size_t length);
```

- **addr** արգումենտը հեռացվող հասցեների տարածության սկզբնական հասցեն է:
- **length** արգումենտը ոչ բացասական ամբողջ թիվ է, որը սահմանում է հեռացվող հատվածի չափը՝ բայթերով:

Արտապատկերման հեռացումից հետո `addr` հասցեի դիմումը գեներացնում է **SIGSEGV** ազդանշան:

Արտապատկերման սինխրոնիզացիա

ՕՅ միջուկը ավտոմատ կերպով գրանցում է ընդհանուր ֆայլային արտապատկերման փոփոխությունները համապատասխան ֆայլում, բայց չի սահմանվում, թե երբ այդ գործողությունը տեղի կունենա:

msync() համակարգային կանչը ծրագրին բացահայտ վերահսկողություն է տրամադրում այն գործողության նկատմամբ, որը սինխրոնացնում է արտապատկերման պարունակությունը ֆայլի հետ (գրանցում է դրանք ֆայլում): Ֆունկցիայի կիրառման օրինակը ներկայացված է տարբերակում:

```
#include <sys/mman.h>
int msync(void *addr, size_t length, int flags);
```

addr և **length** արգումենտները սահմանում են սինխրոնացվող հատվածի սկզբի հասցեն և չափը: **addr** արգումենտի արժեքը պետք է լինի համակարգային եջի բազմապատիկ, իսկ **length** արգումենտը կլորացվում է դեպի վեր մինչև համակարգային եջի մոտակա բազմապատիկը:

flags արգումենտի հնարավոր արժեքներն են.

- **MS_SYNC** – Իրականացնում է սինխրոն գրանցում: *msync()* կանչն արգելափակվում է, մինչև հիշողության հատվածի բոլոր փոփոխված եջերը գրանցվեն ֆայլում:
- **MS_ASYNC** – Իրականացնում է ասինխրոն գրանցում: Փոփոխված եջերը գրանցվում են ֆայլում ավելի ուշ, բայց անմիջապես տեսանելի են դառնում այլ պրոցեսների համար, որոնք կարդում են ֆայլի տվյալ հատվածից: *MS_SYNC* գործողությունից հետո հիշողության հատվածը սինխրոնացված է ֆայլի հետ, իսկ *MS_ASYNC* գործողությունից հետո այն պարզապես սինխրոնացված է ՕՅ միջուկի բոլքերի քեշի հետ:
- **MS_INVALIDATE** – Անվավեր է դարձնում արտապատկերված տվյալների քեշավերված պատճենները: Եթե հիշողության հատվածի բոլոր փոփոխված եջերը գրանցվում են ֆայլում, հիշողության բոլոր եջերը, որոնք չեն համապատասխանում ֆայլի եջերին, նշվում են որպես անվավեր: Հիշողության այդ եջերին հաջորդ դիմումի դեպքում դրանց պարունակությունը պատճենվում է ֆայլի համապատասխան եջերից:

Առաջադրանքներ

1. Ստեղծել որոշակի պարունակությամբ ֆայլ, կատարել `mcat` ծրագիրը՝ որպես հրամանային տողի արգումենտ փոխանցելով ստեղծված ֆայլի անունը:
2. Ստեղծել 1KB ծավալով նոր ֆայլ և սկզբնավորել այն 0 արժեքով: Կատարել մար ծրագիրը՝ որպես հրամանային տողի արգումենտներ փոխանցելով ստեղծված ֆայլի անունը և կամայական արժեք: Բացատրել ցուցադրված հաղորդագրությունները:
3. Ծրագրում `mmap()` կանչից հետո ավելացնել `sleep(5)` հրամանը, կրկին կատարել ծրագիրը, և մինչ պրոցեսը գտնվում է սպասման վիճակում, ընդհատել այն (`Ctrl + Z`): Բացել `/proc/{pid}/maps` ֆայլը և ցուցադրել ստեղծված արտապատկերման հասցեների միջակայքը:
4. Կատարել մար ծրագիրն այնպես, որ գեներացվի `SIGBUS` ազդանշան: Ազդանշանի ստացման մեջ համոզվելու համար ավելացնել ազդանշանը մշակող ֆունկցիա, որը հաղորդագրություն կցուցադրի ազդանշանի ստացման մասին:
5. մար ծրագրում ընդհանուր ֆայլային արտապատկերումը դարձնել մասնավոր՝ `mmmap()` ֆունկցիայի `MAP_SHARED` արժեքի փոխարեն տեղադրելով `MAP_PRIVATE`: Ծրագիրը նորից կատարելով համոզվել, որ արտապատկերման տարածքում կատարված փոփոխությունները չեն գրանցվում ֆայլում:
6. մար ծրագրում ավելացնել `munmap()` կանչը՝ հեռացնելով արտապատկերումը պրոցեսի վիրտուալ հիշողության տարածքից: Այնուհետև արտապատկերման պարունակությունը կարդալու ևս մեկ փորձ կատարել: Բացատրել ծրագրի աշխատանքի արդյունքը:
7. Կատարել `anon_mmap` ծրագիրը: Ընդհանուր անանուն արտապատկերումը դարձնել մասնավոր՝ `mmmap()` ֆունկցիայի `MAP_SHARED` արժեքի փոխարեն տեղադրելով `MAP_PRIVATE`: Կրկին կատարել ծրագիրը և բացատրել ծրագրի նախորդ աշխատանքի արդյունքի հետ տարբերությունը:
8. Փոփոխել 3-րդ լաբորատոր աշխատանքում ներկայացված `writer` և `reader` ծրագրերը՝ `System V` ընդհանուր հիշողության փոխարեն կիրառելով հիշողության արտապատկերում:
9. Գրել ծրագիր, որը կիրականացնի `cp` հրամանի պարզ տարբերակը: Ծրագիրը հրամանային տողից պետք է ընդունի 2 արգումենտ՝ `src` և `dest`, և պատճենի առաջին ֆայլի պարունակությունը 2-րդի մեջ: Եթե առաջին ֆայլը գոյություն չունի, ապա պետք է հաղորդագրություն ցուցադրվի այդ մասին: Եթե երկրորդ ֆայլը գոյություն չունի, ապա այն պետք է ստեղծվի: