Rad sa memorijom

XV6 sistem je izmenjen tako da podržava nove sistemske pozive `mmap()`, `munmap()`, `msync()`, `shm_open()`, `shm_unlink()`, `ftruncate()` i `shm_stat()`. Svi pozivi su budu implementirani tako da nema curenja memorije. Pod curenjem se podrazumeva da čak i nakon što korisnički program oslobodi memoriju, ona u celosti ili neki njen deo, ostane zauzet ili nedostupan.

Implementirani sledeći sistemski pozivi:

mmap()

```
void *mmap(void *addr, int length, int perm, int flags, int fd, int
offset);
```

Ovaj sistemski poziv se koristi za mapiranje nekog targeta u memoriju trenutnog procesa. Argumenti imaju sledeće značenje:

- `addr` - hint za kernel gde bismo želeli da target bude mapiran. Kernel ima puno pravo da ignoriše ovaj hint, u slučaju da nije validan.
- `length` - dužina mapiranog prostora u bajtovima; mora biti umnožak od `PGSIZE`.
- `perm` - bitfield or-ovanih permisija koje opisuju da li se u novomapiran region može pisati (`PROT_WRITE`) i da li se iz njega može čitati (`PROT_READ`).
- `flags` - opcije kojima se naznačava koji je tip targeta. Jedina vrednost različita od nule koja je ovde validna je `MAP_ANONYMOUS`.
    - Ako je `MAP_ANONYMOUS` postavljen, memorijski prostor je inicijalizovan nulama. Ovo predstavlja alternativno mapiran memorijski prostor, nezavisno od `sbrk()`.
    - Ako `MAP_ANONYMOUS` nije postavljeno, mapira se sadržaj fajla opisanog sa `fd`, počevši od pomeraja `offset`, u dužini `length`. Taj sadržaj se smešta u zadatu memorijsku lokaciju. Svaka izmena ovog sadržaja treba da se upiše u datoteku (pri pozivu `msync()`). Ako se dođe do kraja fajla pre dužine `length`, ostatak prostora popuniti nulama. Ne treba proširivati fajl u ovom slučaju.
- `fd` - fajl deskriptor koji opisuje target koji mapiramo. Deskriptor treba da može da bude dobijen pomoću `shm_open()`, ili pomoću `open()`. Ignoriše se kod `MAP_ANONYMOUS`.
- `offset` - pomeraj unutar targeta. Ignoriše se kod `MAP_ANONYMOUS`.

Povratna vrednost ovog sistemskog poziva je adresa na kojoj je izvršeno mapiranje. Ako je došlo do greške, vraća se simbolička konstanta `MAP_FAILED`.

Preporučeno mesto za definisanje novih simboličkih konstanta za sistemske pozive je `kernel/fcntl.h`. Ovaj fajl je već include-ovan u `user/mmaptests.c`.

munmap()

```
int munmap(void *addr, int length);
```

Ovaj sistemski poziv se koristi da se odmapira prethodno mapiran prostor. Početak prostora koji se odmapira se daje sa `addr`, i dužina prostora koji se odmapira se daje sa `length`. Parametar `addr` ne mora da se nalazi na početku nekog mapiranog regiona, već može da bude i u sred regiona. Parametar `length` može da bude takav da zbir `addr` i `length` bude pre kraja mapiranog regiona, ali ne posle njega. Parametar `length` mora da bude umnožak od `PGSIZE`.

Ako je sistemski poziv uspešno izvršen, povratna vrednost je 0, u suprotnom -1.

Regioni memorije mapirani pomoću `mmap()`, tj. strukture u kernel-u koje ih opisuju se moraju pravilno počistiti na kraju rada procesa u slučaju da nisu demapirani tokom rada procesa pomoću `munmap()`.

msync()

```
int msync(void *addr, int length);
```

Ovaj sistemski poziv se koristi za zapisivanje izmenjenog mapiranog fajla na disk. Datoteka koja je mapirana pomoću `mmap()` i menjana u memoriji će ostati u memoriji sve dok korisnički program ne napravi ovaj poziv. Pomoću argumenta `addr` se navodi adresa u mapiranoj memoriji od koje počinju podaci koje treba zapisati na disk. Zapisuje se `length` bajtova počevši od te adrese.

Sistemski poziv vraća -1 ako `addr` pokazuje van stranice mapirane sa `mmap()` koja mapira neku datoteku. U suprotnom sistemski poziv vraća 0.

shm_open() i shm_unlink()

```
int shm_open(const char *name, int flags);
int shm_unlink(const char *name);
```

Uvedena nova vrsta fajl deskriptora koji se odnosi na objekat deljene memorije.

`shm_open()` sistemski poziv otvara ili opciono kreira objekat deljene memorije nazvan `name`. Po konvenciji, naziv treba da počne sa znakom ‘/’, i da ne sadrži druge ‘/’ znakove unutar sebe.

`flags` argument odgovara istoimenom argumentu za sistemski poziv `open()` koji radi sa normalnim fajlovima i prihvata iste flag-ove (`O_RDONLY`, `O_RDWR`, `O_CREATE`). `O_WRONLY` nije validan.

Ova deljena memorija se kreira sa veličinom 0. Veličinu ove deljene memorije možemo izmeniti sistemskim pozivom `ftruncate()`.

`shm_unlink()` briše postojeći objekat deljene memorije sa nazivom `name`.

`shm_open()` vraća novi fajl deskriptor ako se uspešno otvorio `shm` objekat, a -1 ako je došlo do greške. `shm_unlink()` vraća 0 ako je uspešno izbrisao `shm` objekat i -1 ako je došlo do greške.

ftruncate()

```
int ftruncate(int fd, int length);
```

ftruncate() je sistemski poziv koji skraćuje fajl na zadatu dužinu ili ga produžava do nje, popunjavajući novi prostor nulama. Na Unix sistemima se tipično koristi za fajlove, ali je `shm` objektima od naročite vrednosti pošto je to jedini način da se `shm` objektu zada veličina (pošto su pri nastanku dužine 0, a `read()` i `write()` sistemski pozivi nisu validni za `shm` objekte).
Nije implementiran `ftruncate()` za obične fajlove, već samo za `shm` objekte.

shm statistika

Za svaki shared memory objekat je neophodno voditi statistiku o tome koliko različitih procesa je vršilo pisanje u njega. Ovo je realizovano tako što se stranice za ove objekte isprva podešavaju tako da nije dozvoljeno pisanje u njih. Kada proces pokuša da piše u stranicu, desiće se page fault čiji handler treba da poveća odgovarajući brojač i promeni podešavanje za stranicu tako da sada može u nju da se piše. Page fault handler se u ostalim situacijama ponaša na standardan način.

shm_stat()

```
int shm_stat(int fd);
```

Ovaj poziv dohvata informaciju o tome koliko različitih procesa je do sada vršilo pisanje u neki shared memory objekat. Argument `fd` identifikuje objekat, a povratna vrednost je vrednost brojača.

Sistemski poziv vraća -1 ako `fd` nije validan deskriptor.