OS2023 - Domaći 3

Nasleđena deljena memorija

Cilj domaćeg zadatka je izmeniti xv6 sistem tako da podržava deljenje memorije između procesa, ograničeno na procese koji su u roditelj-dete vezi. Roditeljski proces treba da prijavi operativnom sistemu memoriju koja je deljena, i sva neposredna deca tog procesa mogu od operativnog sistema da dobiju u svom adresnom prostoru pristup baš toj memoriji u roditeljskom procesu. Pored ovih izmena, treba implementirati i nekoliko korisničkih programa koji koriste ove izmene sistema.

Od velike je važnosti da ova nova funkcionalnost ne naruši trenutni rad sistema. Pazite da procesi mogu normalno da se startuju i završe, kao i da se sva zauzeta memorija uredno oslobađa. Obratiti pažnju na to ko treba da oslobodi memoriju.

Kôd koji treba koristiti kao polaznu tačku za domaći zadatak može da se dohvati pomoću komande:

```
git clone https://github.com/RAFOperativniSistemi/xv6-raf --branch
vezbe6 domaci3
```

Na materijalima postoji snimak koji demonstrira kako izgleda rad sistema sa urađenim zadatkom.

Sistemski pozivi

Neophodno je implementirati dva nova sistemska poziva: **`share_data`** i **`get_data`**.
Oba sistemska poziva rade sa novom **`shared`** strukturom koja ima sledeće atribute:

- Stringovni naziv, maksimalno 10 karaktera.
- Pokazivač na početak deljenog prostora.
- Veličina deljenog prostora.

`int share_data(char *name, void *addr, int size);`
Ovaj sistemski poziv se poziva iz roditeljskog procesa kako bi se prijavila deljena struktura. Pri prijavljivanju deljene strukture se navodi jedinstveni stringovni naziv za strukturu (ako je dati string duži od 10 karaktera, uzeti prvih 10), adresa na kojoj se struktura nalazi, i veličina strukture. Neophodno je izmeniti **`proc`** strukturu unutar xv6 tako da skladišti do 10 **`shared`** struktura. Neiskorišćene **`shared`** strukture treba da budu označene veličinom 0 u

novokreiranom procesu.
U slučaju uspešnog upisivanja nove **shared** strukture, ovaj sistemski poziv vraća njen redni broj unutar procesa (0~9). Moguće greške za ovaj sistemski poziv su:

- -1 - loše prosleđen parametar.
- -2 - već postoji **shared** struktura sa istim imenom.
- -3 - već postoji 10 **shared** struktura u trenutnom procesu.

int get_data(char *name, void **addr)

Ovaj sistemski poziv se poziva iz deteta procesa kako bi se pristupilo **shared** strukturi koju je roditelj prethodno prijavio pomoću **share_data** sistemskog poziva. Prvi parametar je stringovni naziv strukture kojoj želimo da pristupimo, a drugi parametar je pokazivač (prosleđen po referenci) koji treba da pokazuje na deljeni prostor nakon izvršavanja sistemskog poziva.
Dete proces bi trebalo da već ima pripremljenu **shared** strukturu iz koje samo čita vrednost za addr pokazivač. Više o izmenama dete-procesa videti u odeljku izmene sistema.

U slučaju uspešnog izvršavanja ovog sistemskog poziva, povratna vrednost je 0. Moguće greške za ovaj sistemski poziv su:

- -1 - loše prosleđen parametar.
- -2 - ne postoji **shared** struktura sa navedenim nazivom.

Izmene sistema

Neophodno je napraviti tri primarne izmene u sistemu za potrebe zadatka: **fork()**, **exec()** i gašenje procesa.

fork() izmena

Sistemski poziv **fork()** sada treba da se pobrine da novokreirani proces nasledi sve **shared** strukture od svog roditelja. Pošto je memorijska slika identična kod deteta, nema potrebe menjati bilo šta unutar samih struktura ili straničnih struktura deteta. Dok ne uradi **exec()**, dete proces ne može da pristupi roditeljevim deljenim objektima, već ima pristup samo svojoj kopiji. Pored ovoga, u **proc** strukturi deteta treba sačuvati pokazivač ka roditeljevom direktorijumu stranica pomoću novog atributa koji je posebno namenjen za to.

exec() izmena

Oganičićemo regularnu veličinu procesa u virtuelnoj memoriji na 1GB, a počevši od 1GB ćemo da mapiramo deljene objekte. Sistemski poziv **exec()** treba izmeniti tako da novostartovani proces u svom virtuelnom prostoru, počevši od 1GB, ima mapirane sve stranice koje su navedene u **shared** strukturama nasleđenim od roditelja. Ova mapiranja

treba da pokazuju na iste okvire kao u roditeljskom direktorijumu, kojem imamo pristup zato što smo ga sačuvali pri rađenju **`fork()`** sistemskog poziva. Nakon što se obavi mapiranje, neophodno je izmeniti **`shared`** strukture, tako da su virtuelne adrese sada validne uzimajući u obzir novo mapiranje. Obratiti pažnju na činjenicu da adrese sasvim verovatno neće biti okrugle po modulu 4K. Dozvoljeno je koristiti višestruka mapiranja u slučaju da različiti **`shared`** objekti pokazuju na isti okvir.

Izmena gašenja procesa

Neophodno je da i roditeljski i svi dečiji procesi mogu uspešno da se završe, kao i da se pri njihovom gašenju oslobodi sva zauzeta memorija. Obratiti pažnju na to ko sme šta da obriše pri gašenju procesa.

Korisnički programi

Neophodno je implementirati tri korisnička programa - jedan roditeljski (**`dalle`**), koji u okviru svog rada startuje druga dva (**`coMMa`** i **`liSa`**).

`dalle`

Korisnički program **`dalle`** pokreće interaktivni sistem za analiziranje tekstualnih fajlova. On kao argument prima putanju do fajla (obratiti pažnju da se nama program nalazi na `"/bin/"` putanji, gde su smešteni svi korisnički programi), ili ako je nema, podrazumevana vrednost je `"../home/README"`. Program priprema sledeće strukture koje će deliti sa svojom decom:

- Putanja do fajla koji obrađujemo - *char **
- Brojač do koje rečenice smo stigli u okviru fajla - *int*
- Najduža reč u trenutnoj rečenici - *char **
- Veličina najduže reči u čitavom tekstu do trenutne rečenice (sa njom) - *int*
- Najkraća reč u trenutnoj rečenici - *char **
- Veličina najkraće reči u čitavom tekstu do trenutne rečenice (sa njom) - *int*
- Najduža reč u čitavom tekstu do trenutne rečenice (sa njom) - *char **
- Indikator komande - *int*
- Najkraća reč u čitavom tekstu do trenutne rečenice (sa njom) - *char **

Nakon kreiranja devet deljenih struktura za ove podatke, startuju se dva nova procesa kao deca: **`coMMa`** i **`liSa`**. Nakon startovanja dece, **`dalle`** samo čeka završavanje oba procesa i potom se i sam završava.

coMMa

Korisnički program **coMMa** ima zadatak da prihvata komande od korisnika u petlji. Moguće komande koje korisnik može da zada su:

- **latest** - ispisuje koju rečenicu po redu obrađujemo, kao i najdužu i najkraću reč u toj rečenici
- **global extrema** - ispisuje najveću i najmanju reč do sada pronađenuo i njihovu veličinu
- **pause** - pauzira obradu teksta.
- **resume** - nastavlja obradu teksta.
- **end** - javlja **liSa** procesu da treba da se ugasi, i sam se završava.

Komande **global extrema** i **latest** direktno čitaju iz **shared** struktura kako bi ispisale navedene podatke. Komande **pause**, **resume** i **end** šalju signale **liSa** procesu pomoću deljenog indikatora komande.

liSa

Korisnički program **liSa** ima zadatak da obrađuje tekstualni fajl. Otvorivši fajl, pronalazi najkraću i najdužu reč u trenutnoj rečinici (kao i globalno) i smešta ih u odgovarajuću deljenu strukturu, zatim uvećava deljeni brojač trenutne rečenice i poziva *sleep(150)* kako bismo simulirali analizu podataka. Nakon obrade svake rečenice, ovaj proces treba da proveri da li je **coMMa** naveo neku komandu. U slučaju da je navedena **pause** komanda, treba odraditi *sleep(1)*, i onda ponovo proveriti komandu. Tek kada se navede **resume** ili **end** komanda se nastavlja sa radom, odnosno završava rad programa. Pri završetku rada programa ispisati poruku da je zadatak gotov.

Bodovanje

Zadatak se boduje na sledeći način:

- **fork()** izmena = 2 boda
- **exec()** izmena = 8 bodova
- Pravilno gašenje procesa = 2 boda
- **share_data()** sistemski poziv = 3 boda
- **get_data()** sistemski poziv = 3 boda
- Korisnički programi (boduju se samo ako zaista rade) = 7 boda

U slučaju da je neka od stavki implementirana parcijalno, biće dodeljeni parcijalni poeni. U slučaju parcijalne implementacije, obavezno se pobrinuti da je moguće pokazati da napravljena izmena zaista ima efekta.

Predaja i rokovi

Zadatak se predaje putem **google forms**-a, a link će biti naknadno objavljen. Obavezno izvršiti komandu "**make clean**" pre predaje. Direktorijum koji sadrži xv6 kod (zvaće se "domaci3", ako je skinut pomoću git komande navedene na vrhu ovog dokumenta) preimenovati na sledeći način: "os_2022_d3_ime_prezime_ind".
Npr. "os_2022_d3_student_studentic_rn0101".

Arhivirati ovaj direktorijum (.zip). Naziv arhive mora da bude u obliku:
"os_2022_d3_ime_prezime_ind.zip"
Npr. "os_2022_d3_student_studentic_rn0101.zip"

Rok za predaju je:

- Sreda, 24. maj 23:59:59 za grupu 205.
- Petak, 26. maj 23:59:59 za grupe 201, 202.
- Utorak, 30. maj 23:59:59 za grupe 203, 204.

Rok je definisan po grupi kojoj student zvanično pripada. Studenti koju slušaju vežbe van svog termina i dalje moraju da poštuju termine na osnovu zvaničnog spiska. Za sve ponovce se primenjuje najkasniji rok.

Neće se pregledati zadaci (tj. biće dodeljeno 0 poena) ako se desi bilo koje od sledećeg:

- Sadržaj google forms-a nije po navedenom obliku.
- Naziv arhive nije po navedenom obliku.
- Predaja se desi nakon navedenog roka.
- Kod se ne kompajluje.
- Kod nije uredno uvučen.

Odbrana domaćih zadataka je obavezna. Ako ste iz bilo kog razloga sprečeni da prisustvujete odbrani, obavezno to najavite što pre, kako bismo mogli da zakažemo vanredni termin za odbranu.

Svrha odbrane je da se pokaže autentičnost zadatka. Ovo podrazumeva odgovaranje na pitanja u vezi načina izrade zadatka, ili izvršavanje neke izmene nad zadatkom na licu mesta. U slučaju da odbrana nije uspešna, zadatak se boduje sa -5 bodova umesto

namenjenog broja bodova.