OS2021 - Projekat

Unapređenje terminala i deljena memorija

Projekat se radi umesto trećeg i četvrtog domaćeg zadatka. Ako radite projekat, nije dozvoljeno da radite treći ili četvrti domaći zadatak.

Cilj projekta je izmeniti xv6 sistem tako da podržava sledeće funkcionalnosti:

- Unapređeni rad sa sistemom koji podržava:
  - Višestruke terminale
  - Istoriju komandi na konzoli
- Deljena memorija između procesa, pomoću deljenih memorijskih objekata.

Potrebno je implementirati sledeće sistemske pozive i korisničke programe:

- Sistemski pozivi:
  - `int shm_open(char *name);`
  - `int shm_trunc(int fd, int size);`
  - `int shm_map(int fd, void **va, int flags);`
  - `int shm_close(int fd);`
- Korisnički programi:
  - `colour`

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

```
git clone git://github.com/RAFOperativniSistemi/xv6-raf --branch
vezbe8 projekat
```

Na materijalima postoji snimak koji demonstrira kako izgleda rad sistema sa unapređenim terminalima, kao i korisnički programi koji mogu da se koriste za testiranje deljene memorije između procesa.

Predaja i rokovi

Zadatak se predaje putem mail-a na bmilojkovic@raf.rs ili mveniger@raf.rs.
Obavezno izvršiti komandu "make clean" pre predaje, pošto Google mail neće dozvoliti predaju rada koji u sebi sadrži izvršni kod. Direktorijum koji sadrži xv6 kod (zvaće se "domaci2", ako je skinut pomoću git komande navedene u ovom dokumentu) preimenovati na sledeći način: "os_2021_proj_ime_prezime_ind".
Npr. "os_2021_proj_student_studentic_rn0101".

Arhivirati ovaj direktorijum (.zip) i arhivu poslati kao attachment uz mail. Naziv arhive mora da bude u obliku: “os_2021_proj_ime_prezime_ind.zip”
Npr. “os_2021_proj_student_studentic_rn0101.zip”

U tekstu mail-a obavezno navesti:

- Ime i prezime
- Broj indeksa
- Grupa, po zvaničnom spisku, ili “ponovac” za ponovce

Subject mail-a mora da bude u obliku: “[OS 2021] Proj ime_prezime_ind”.
Npr. “[OS 2021] Proj student_studentic_rn0101”

Rok za predaju je:

- Subota, 5. jun 23:59:59 za sve studente.

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

- Sadržaj mail-a nije po navedenom obliku.
- Subject mail-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 projekta je obavezna. Termin za odbranu projekta će biti tokom druge kolokvijumske nedelje. Odbrane će se vršiti individualno. Raspored odbrana će biti formiran i objavljen ubrzo nakon predaje projekta. 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 -20 bodova umesto namenjenog broja bodova.

Unapređenje terminala

Unapređenje terminala se obavlja iz dve faze: podrška za rad sa višestrukim terminalima, i podrška za istoriju unetih komandi na konzoli.

Višestruki terminali

Sistem treba da ima 6 aktivnih terminala, umesto jednog. Svaki od ovih terminala ima svoj `sh` proces koji prihvata komande, i ovi `sh` procesi treba da svi budu aktivni istovremeno. Ovi procesi se svi startuju unutar `init` procesa, i neophodno je izmeniti `init` proces tako da se ovo dešava. Za svaki od terminala treba da postoji posebna datoteka koja se koristi za ispis na tom terminalu i za čitanje ulaza na tom terminalu, kao što je `/dev/console` bio do sada za čitav sistem. Ove datoteke treba da se zovu `/dev/ttyX`, gde je `X` broj terminala, sa numeracijom počevši od 1, dakle `/dev/tty1`, `/dev/tty2`, itd. Sve ove datoteke treba da imaju svoj `major` atribut postavljen na 1, pošto svaka od njih predstavlja neku konzolu. Sistem interno može da razlikuje ove datoteke na osnovu njihovog `minor` atributa, koji treba da odgovara rednom broju terminala, tj. imaće vrednost 1, 2, ... 6.

Promena trenutno aktivnog terminala se vrši pomoću kombinacije tastera `alt + X`, gde je `X` broj terminala na koji korisnik želi da se prebaci. Naziv trenutno aktivnog terminala treba da bude fiksirano ispisan u donjem desnom ćošku ekrana, kao što je prikazano na snimku.

Za svaki terminal treba da bude vezana boja slova i boja pozadine tog terminala. Ove boje treba da bude moguće menjati pomoću korisničkog programa `colour`. Takođe svaki terminal treba da ima različitu kombinaciju boja po default, i izbor boje se ostavlja studentu.

colour

Korisnički program `colour` treba da omogući podešavanje boje slova i boje pozadine na trenutno aktivnom terminalu. Pri navođenju boja mogu da se koriste hex zapisi za boje, koji uvek imaju prefiks 0x i uvek su dvocifreni brojevi, npr. “0x02”. Pored hex zapisa, takođe treba da bude moguće koristiti tekstualni opis za boje, koji je oblika “black”, “blue”, “green”, “aqua”, “red”, “purple”, “yellow”, ili “white”. Za svetlije varijante boja se koriste isti stringovi sa karakterom “L” kao prefiksom, dakle “Lblack”, “Lblue”, itd. Svi parametri su opcioni, i mogu da budu navedeni u proizvoljnom redosledu. Ovaj program treba da prima sledeće opcije:

- `--help (-h)` ispisuje help meni.
- `reset` vraća boje na xv6 podrazumevane boje.
- `--foreground (-fg)` podešava boju slova, i očekuje tekstualni opis boje.

- `--background (-bg)` podešava boju pozadine, i očekuje tekstualni opis boje.
- `[bez parametra]` podešava obe boje, i očekuje hex zapis.

Istorija komandi na konzoli

Svaka konzola za sebe treba da održava istoriju koja sadrži poslednjih 8 unetih komandi. Ovoj istoriji može da se pristupi pomoću strelica gore i dole na tastaturi. Strelica gore prikazuje prethodnu komandu u istoriji, a strelica dole narednu. Ako korisnik ukuca neki tekst kao početak komande, i onda počne da lista istoriju, treba da može da ponovo vidi svoj delimično ukucan tekst ako se vrati na početak istorije. Takođe treba da bude moguće izmeniti neku liniju iz istorije i uneti je kao novu komandu. Nije neophodno da se izmena neke linije u istoriji pamti na tom mestu u istoriji, već samo treba da se pojavi kao najnovija komanda.

Deljena memorija pomoću shm objekata

Neophodno je obezbediti kreiranje “shared memory” (shm) objekata, kao i njihovo mapiranje u virtuelni adresni prostor procesa. Ovi objekti su globalni na nivou sistema, i identifikuju se pomoću stringovnog naziva koji se navodi pri njihovom otvaranju. Objekti se kreiraju pomoću sistemskog poziva `shm_open()`. Pri prvom otvaranju nekog objekta, tj. pri njegovom kreiranju, veličina objekta u memoriji se postavlja na 0. Nakon toga se pomoću sistemskog poziva `shm_trunc()` postavlja veličina za objekat, koja se nakon toga ne može promeniti - svaki naknadni poziv `shm_trunc()` treba da se ignoriše, tj. ne treba da promeni veličinu objekta.

Ako je neki objekat već kreiran, i nakon toga neki proces koji nije do sada pristupao tom objektu pokuša da ga otvori, on će dobiti pristup već kreiranom objektu - to jest procesi koji su inače potpuno nezavisni treba da mogu da pristupe istom shm objektu, dok god koriste isti naziv objekta. Child procesi kreirani pomoću `fork()` sistemskog poziva takođe treba da imaju pristup shm objektu kao da su uradili `shm_open()` za njega, nezavisno od roditelja, i od njih se takođe očekuje da zatvore objekat kada završe sa radom.

Proces će kao rezultat `shm_open()` sistemskog poziva dobiti fajl deskriptor koji opisuje otvoreni objekat, i ovaj deskriptor se onda može iskoristiti da se izvrši inicijelno podešavanje veličine objekta, kao i mapiranje objekta u virtuelni adresni prostor procesa. Ovo mapiranje se vrši pomoću sistemskog poziva `shm_map()`. Proces će kao rezultat ovog poziva dobiti pokazivač na početak deljenog objekta, i taj pokazivač može da se koristi za pristup sadržaju deljene memorije.

Kada proces završi sa radom sa objektom, on treba da pozove sistemski poziv `shm_close()`, i da time naznači da ne želi više da radi sa objektom. Kada su svi procesi ovo učinili, sistem treba da ukloni objekat iz memorije. Ako neki proces izostavi da ovo učini, onda zatvaranje treba da se obavi unutar `exit()` sistemskog poziva, kao što se čini i za datoteke.

Sistem treba da bude ograničen na 64 shm objekta u jednom trenutku, i svaki shm objekat treba da može da bude dugačak najviše 32 stranice.

int shm_open(char *name)

Ovaj sistemski poziv kreira objekat sa navednim imenom ako on ne postoji, a u suprotnom ga samo otvara za trenutni proces. U svakom slučaju se u `proc` strukturi trenutnog procesa zapisuje da je ovaj objekat sada otvoren za trenutni proces, i vraća se fd koji opisuje taj objekat. Povratne vrednosti za ovaj sistemski poziv su:

- -1: objekat nije uspešno kreiran
- >0: uspešno otvoren objekat, i vraćena vrednost je fd koji opisuje objekat.

int shm_trunc(int fd, int size)

Ovaj sistemski poziv podešava veličinu novokreiranog shm objekta na `size`. Sistemski poziv treba da ima efekta samo ako je u pitanju prvi `shm_trunc()` poziv za navedeni objekat. Povratne vrednosti za ovaj sistemski poziv su:

- -1: neuspešna alokacija za objekat.
- >0: podešavanje veličine shm objekta je završeno uspešno, i vraćena vrednost je veličina objekta.

int shm_map(int fd, void **va, int flags)

Ovaj sistemski poziv obavlja mapiranje otvorenog shm objekta u virtuelni adresni prostor trenutnog procesa, i postavlja `*va` na početak tog prostora. Argument `flags` navodi da li se objekat mapira samo za čitanje, ili za čitanje i pisanje, i koristi iste konstante kao open - `O_RDWR` ili `O_RDONLY`. Nije dozvoljeno mapirati prostor samo za pisanje. Povratne vrednosti za ovaj sistemski poziv su:

- -1: neuspešno mapiranje objekta.
- 0: mapiranje shm objekta je završeno uspešno.

int shm_close(int fd)

Ovaj sistemski poziv zatvara shm objekat, tj. onemogućava dalje korišćenje ovog deskirptora unutar trenutnog procesa. Ovde se takođe obavlja odmapiranje objekta u slučaju da je bio mapiran. Ako je ovo poslednji proces koji je koristio objekat, onda ovaj sistemski poziv treba takođe da obriše objekat iz sistema. Povratne vrednosti za ovaj poziv su:

- -1: zatvaranje deskriptora nije bilo uspešno.
- 0: zatvaranje deskriptora je završeno uspešno.

Bodovanje

Projekat se boduje na sledeći način:

- Unapređenje terminala **= 20 bodova**
  - Višestruki terminali = *11 bodova*
  - `colour` = *2 bodova*
  - Istorija komandi na konzoli = *7 bodova*
- Deljena memorija pomoću shm objekata **= 20 bodova**
  - `shm_open()` = *4 boda*
  - `shm_trunc()` = *2 boda*
  - `shm_map()` = *8 bodova*
  - `shm_close()` = *2 boda*
  - `Izmene ostatka sistema` = *4 boda*

U slučaju da je neka od stavki implementirana parcijalno, biće dodeljeni parcijalni poeni. U slučaju da se jave greške koje nisu pokrivene priloženim testom, mogu da budu dodeljeni negativni poeni do -5 poena.