Device fajlovi

Cilj je izmeniti xv6 sistem tako da podržava razne nove device fajlove, korisnički alat za manipulisanje njima i sistemski poziv koji omogućava njegovu implementaciju.

Ovaj zadatak podrazumeva samostalno čitanje izvornog koda xv6 sistema i namenjen je da podstakne snalaženje u većim postojećim projektima.

Device fajlovi

Potrebno je implementirati sledeće specijalne fajlove:

- /dev/null
- /dev/zero
- /dev/kmesg
- /dev/random
- /dev/disk

Za nove device fajlove alocirati nove major brojeve. Dozvoljeno je da /dev/null i /dev/zero koriste isti major broj i da se razlikuju minor brojem.

Neophodno je implementaciju read() i write() operacija za sve device fajlove staviti u novi izvorni fajl u kernel-u. Komanda grep je korisna za određivanje koje je header fajlove neophodno include-ovati.

Modifikovati user/init.c tako da se novi device fajlovi kreiraju sistemskim pozivom mknod().

/dev/null

Najjednostavniji posebni fajl. Upis u ovaj fajl prijavljuje uspeh (vraća isti broj upisanih bajtova kao prosleđena veličina bafera); sadržaj bafera se ignoriše. Čitanje iz ovog fajla se tretira kao čitanje iz praznog fajla (vraća nulu).

Ovaj fajl se najčešće koristi u shell skriptama i generalno za redirekciju stream-ova. Kada ne želimo da se izlaz nekog programa printuje na ekran ili kada ne želimo da prosledimo fajl programu koji ga zahteva, redirect-ovaćemo output, tj. input tog programa u /dev/null.

/dev/zero

Upis u ovaj fajl se ponaša identično kao za /dev/null. Čitanje iz ovog fajla puni čitav prosleđen bafer nulama i vraća broj popunjenih bajtova.

Ovaj fajl se često koristi za kreiranje praznih fajlova fiksne veličine i za bristanje sadržaja diskova.

Za /dev/null i /dev/zero je neophodno napisati testove u korisničkom prostoru koji dokazuju ispravnost read() i write() operacija nad ovim fajlovima.

/dev/kmesg

Modifikovati kernel-ovu cprintf() funkciju tako da se sve što kernel ispisuje čuva u internom baferu. Iz ovog fajla se može iščitati sadržaj tog bafera. Upis u ovaj fajl nije validna operacija. Dodati par poziva funkcije cprintf() npr. u funkciju main(), tako da kernel ispisuje diagnostičke poruke dok sistem boot-uje.

Na Unix-olikim sistemima se za ovu svrhu uglavnom koristi komanda dmesg. Na Linux-u ona komunicira specijalnim protokolom sa /dev/kmsg.

Važno je da operacije čitanja ovog fajla poštuju vrednost offset-a; drugim rečima fajl mora da se ponaša ispravno u slučaju višestrukih iščitavanja sistemskim pozivom read().

Važno je i da upis više teksta od veličine bafera ne dovede do prežvrljanja ostalog sadržaja memorije. Upis preko granice bafera se mora smisleno podneti.

/dev/random

Unix sistemi pružaju korisničkim programima generator nasumičnih brojeva. Računari su fundamentalno determinističke mašine tako ne mogu lako generisati stvarno nasumične brojeve. Jednostavni generatori pseudonasumičnih brojeva kao rand() iz standardne biblioteke samo koriste trenutno vreme kao inicijalnu vrednost, pa tu vrednost transfomišu fiksnom, tj. determinističnom operacijom. Operativni sistemi rešavaju taj problem tako što koriste spoljašnje uticaje radi jačanja “kvaliteta” nasumičnih brojeva. Na primer, svaki put kada korisnik pomeri miš ili pritisne taster na tastaturi, ili kada se desi disk interrupt ili uključi ventilator kao reakciju na toplotu procesora, tip i vreme događaja trebaju na neki način da utiču na entropiju (nasumičnost) u sistemu.

/dev/random se često koristi za generisanje šifara i nasumičnih keyfile-ova za upotrebu u kriptografiji. Koristi se takođe i za pripremu diskova za Full Disk Encryption - enkriptovani

podaci se teško razlikuju od nasumičnih brojeva, tako da punjenje diska nasumičnim brojevima pre inicijalizacije fajl sistema krije koliko je prostora na disku zapravo popunjeno.

Vaš xv6 sistem treba da obezbedi generisanje nasumičnih brojeva. Nasumičnost treba obezbediti pomoću reakcije na spoljašnje prekide - minimum tastaturu. Svaki put kada korisnik pritisne neki taster, sken kod tog tastera i vreme kada je pritisnut trebaju da nekako utiču na naredni generisani nasumični broj. Globalna promenljiva ticks predstavlja broj otkucaja sistemskog časovnika od startovanja sistema. Realne implementacije /dev/random-a koriste pažljivo izabrane kriptografske algoritme da garantuju kvalitet nasumičnih brojeva. U vašoj implementaciji je dovoljno da svaka promena prostom aritmetikom utiče na iščitane bajtove tako da na oko deluju dovoljno nasumični.

Čitanje iz ovog fajla puni bafer nasumičnim bajtovima i mutira entropiju tako da sledeće iščitavanje vrati drugu vrednost u baferu. Upis u ovaj fajl mutira bafer vrednošću upisanih bajtova i vremenom kada se upis desio.

/dev/disk

Apstrakcija nad hard diskom. Operacije čitanja i pisanja nad ovom fajlu operišu direktno na blokovima diska, zaobilažeći fajl sistem.

I /dev/disk, kao /dev/kmesg, treba poštovati offset fajla. Treba biti moguće npr. premotati do bloka 34 i iščitati ga.

Sistemski pozivi

int lseek(int fd, int offset, int whence);

read() i write() operacije nad otvorenim fajlovima dodaju internom offset-u file strukture za taj fajl u tom procesu broj bajtova koji su uspešno iščitani/ispisani.

Sistemski poziv lseek() omogućava korisničkim programima da ručno modifikuju offset otvorenog fajla fd. Ponašanje sistemskog poziva zavisi od argumenta whence. Argumentu whence (“odakle”) se prosleđuje jedna od sledećih simboličih konstanta (koje se trebaju definisati u nekom header fajlu koji će se include-ovati i u kernel-u i u korisničkom prostoru):

- SEEK_SET - offset fajla se setuje na apsolutnu vrednost offset argumenta
- SEEK_CUR - offset fajla se sabira sa argumentom offset (negativna vrednost dovodi do premotavanja unazad)
- SEEK_END - offset fajla se setuje na zbir veličine fajla i offset argumenta

Potrebno je izmeniti sistemski poziv write() tako da u slučaju da je fajlov offset postavljen na vrednost veću od prave dužine fajla, prostor između starog sadržaja fajl i mesta gde se upisuju novi podaci se treba popuniti nulama.

Napisati test program u korisničkom prostoru koji demonstrira ispravnost sistemskog poziva lseek() u svakom modu operacije.

void clrscr(void);

Sistemski poziv clrscr() treba da obriše ceo ekran. Kursor treba da se nalazi u gornjem levom ćošku ekrana nakon brisanja.

Napisati test program u korisničkom prostoru koji demonstrira ispravnost sistemskog poziva clrscr().

int getcp(int *x, int *y);

Sistemski poziv getcp() treba da obezbedi čitanje trenutne pozicije kursora. Pozicija se upisuje u prosleđene int-ove po referenci. Prvi int treba da sadrži kolonu (0-79) u kojoj se nalazi kursor, a drugi int treba da sadrži red (0-24) u kojem se nalazi kursor.

int setcp(int x, int y);

Sistemski poziv setcp() treba da obezbedi podešavanje trenutne pozicije kursora. Pozicija se zadaje kao par int-ova, gde je prvi int kolona (0-79) u kojoj treba da se nalazi kursor, i drugi int je red (0-24) u kojem treba da se nalazi krusor.

Sistemski poziv vraća 0 ako je pomeranje uspešno, i -1 ako nije. Neuspeh se dešava ako su zadate vrednosti van opsega ekrana.

Napisati test program u korisničkom programu koji demonstrira ispravnost sistemskih poziva getcp() i setcp().

Korisnički program

dd

dd je Unix program koji se uglavnom koristi za manipulaciju diskova. On kopira sadržaj input fajla u output fajl u baferima od po bs bajtova. Na kraju ispisuje diagnostičku poruku na standardni error sa brojem prekopiranih blokova i koliko je to u bajtovima.

Za razliku od većine Unix programa, dd ima neobičnu sintaksu argumenata na komandnoj linije: key=value parovi. Potrebno je implementirati sledeće parametre:

- if - input file (default je standardni input)
- of - output file (default je standardni output)
- bs - block size (default je 512)
- count - broj blokova (default je do kraja input fajla)
- skip - koliko blokova treba preskočiti u input fajlu (default 0)
- seek - koliko blokova treba preskočiti u output fajlu (default 0)

Napomena: korisnička biblioteka xv6 sistema sadrži standardnu implementaciju malloc-a.

Primeri

- dd if=/home/README of=/home/readme2

Kopira README u readme2.

- dd if=/home/README of=/home/snippet skip=2 count=1

Kopira 512B iz README fajla, nakon prvih 1KB.

- dd if=/dev/random of=/home/keyfile bs=64 count=2

Generiše nasumičan fajl velik 128 bajtova.

- dd if=/dev/random of=/test bs=8 count=1 seek=7

Generiše fajl sa 56 nula, pa 8 nasumičnih bajtova.

- dd if=/dev/random count=1

Prikazuje 512 nasumičnih bajtova na ekranu.

Napomena: Unix program koji se ne gasi se normalno može prekinuti sa Ctrl-c, ali u xv6 to nije implementirano, tako da će morati da se ugasi qemu ako npr. neograničeno kopiramo /dev/random na ekran.

- dd if=/dev/zero of=/dev/disk

Briše sadržaj diska tako da će xv6 crash-ovati pri sledećem boot-u.
(Posle uraditi rm fs.img tako da sledeće pokretanje make qemu regeneriše fajl sistem.)