OS2022 - Domaći 4

Konkurentno brojanje frekvencije reči

Cilj domaćeg zadatka je napisati C program za Linux sistem koji korisniku obezbeđuje brojanje frekvencije reči u datim fajlovima. Korisnik unosi proizvoljan broj fajlova, i na ekranu dobija najčešće reči sa njihovom frekvancijom. Potencijalne reči se čitaju iz tekstualnih datoteka. Reči u memoriji treba da se skladište u heš tabeli, koja treba da podržava konkurentno ubacivanje reči, kao i konkurentnu pretragu. Takođe treba obezbediti funkcionalnost filtriranje stop reči.

Ovaj zadatak ne zahteva rad u xv6 sistemu, niti mu je postavka grana xv6 repozitorijuma.

Uz zadatak su priloženi podaci koji mogu da se koriste za testiranje, u okviru arhive data.zip, koja se nalazi na materijalima. **Pri predaji se obavezno pobrinuti da se ovi podaci uklone iz projekta pre predaje.**

Zadatak podrazumeva samostalno čitanje man page dokumenata i drugih materijala radi pronalaženja odgovarajućih funkcija i sistemskih poziva za manipulaciju direktorijuma i vremena modifikacije.

Na materijalima postoji snimak koji demonstrira kako izgleda urađen zadatak.

Interakcija sa korisnikom

Korisnik sistemu može da zadaje komande preko komandne linije. Treba podržati sledeće komande:

1. `_count_ <fajl1>`
2. `_stop_`

Komanda `_count_` dodaje novi fajl sa tekstom u sistem. Pri izvršavanju ove komande treba krairati novu scanner nit koja će da skenira tekst unutar fajla.
Komanda `_stop_` gasi aplikaciju.

Ako korisnik na konzoli otkuca bilo šta drugo i pritisne enter, podrazumeva se da je poslednja reč uneta na toj liniji reč za koju želimo da vidimo ukupnu frekvenciju. Ukoliko korisnik unese više reči u istoj liniji, bilo koje pre poslednje se ignorišu. Na ekranu treba odmah ispisati frekvenciju tražene reči. Ako pretraga još uvek traje, treba ispisati trenutnu frekvenciju reči. Korisnik ne može da unosi komande dok se ispisuje frekvencija. Korisnik završava trenutnu pretragu kombinacijom tastera CTRL+D. Nakon unosa CTRL+D se zaustavlja ispis i korisnik može da unese novu komandu ili prefiks za pretragu.

Scanner nit

Svaka scanner nit obilazi svoj fajl i čita reči iz njega. Nit treba da ignoriše bilo kakve poddirektorijume i da čita samo fajl, pretpostavljajući da one sadrže običan ASCII kodiran tekst. Ova nit treba da prati kada je fajl poslednji put menjan pomoću deljenog niza koji sadrži spisak svih do sada skeniranih fajlova, kao i njihovih poslednjih vremena izmena.

Kada nit dobije neki fajl koju nije već pročitala, ili fajl koja je menjana od prethodnog čitanja, ona započinje čitanje. Nit čita reč po reč iz fajla. Reči su razdvojene znakovima SPACE (‘ ‘), TAB (‘\t’) ili ENTER (‘\n’). Ako reč sadrži bilo kakve karaktere osim malih ili velikih slova engleske abecede, treba je celu ignorisati. Reči će imati maksimalnu dužinu od 63 slova. Za reči duže od toga zadržati samo prvih 63 karaktera. Kada je reč pročitana, sva velika slova treba pretvoriti u mala. Tako obrađena reč se ubacuje u hash map strukturu koja je deljena između svih scanner niti.

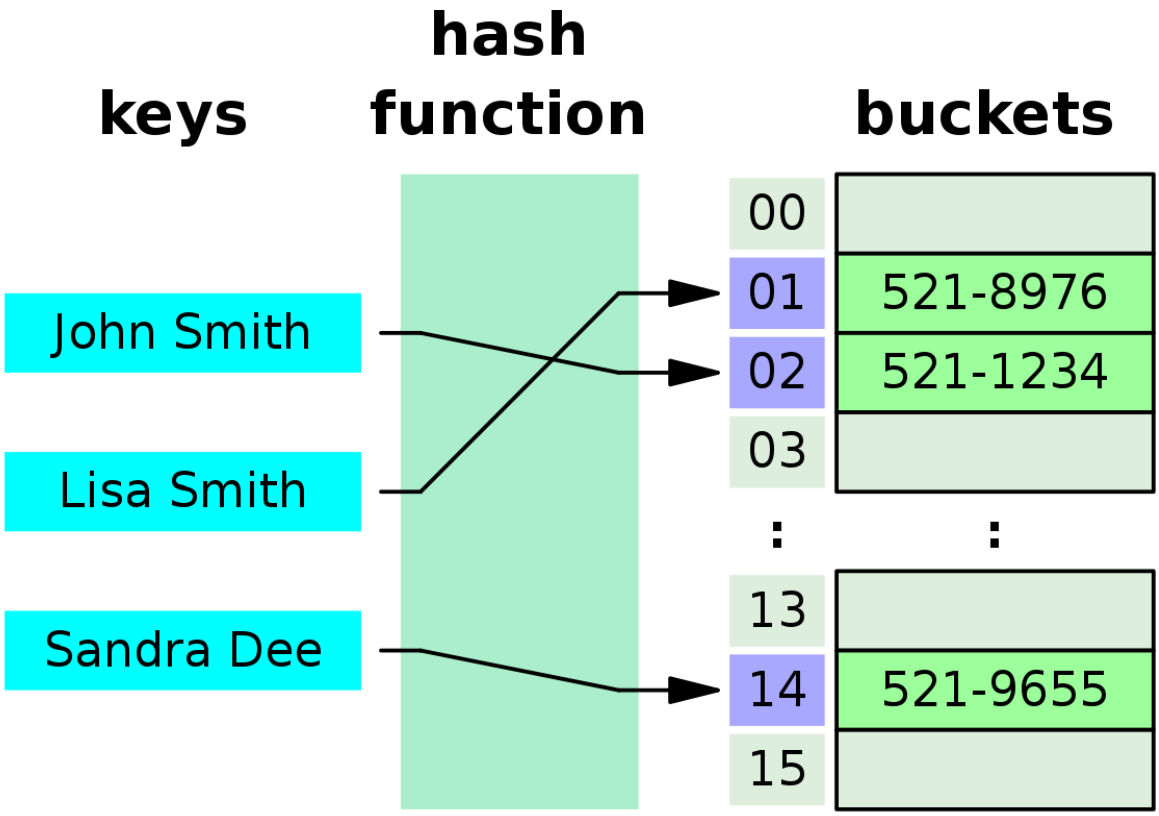
Kada nit obiđe svoj zadati fajl, ona pravi pauzu od pet sekundi, nakon čega proverava da li je fajl menjan. Ako je fajl menjan ona ponovo obilazi fajl. Nit treba da obavlja ovaj posao u beskonačnoj petlji. Ovo znači da sistem treba da detektuje i reaguje na dodavanje nove datoteke, kao i izmenu postojećih datoteka. Izmena fajla u ovom slučaju može da podrazumeva samo dodavanje novih reči u fajl.

Hash map struktura

Hash map je struktura koja sadrži:

1. Heš ključa
2. Vrednost

Pored toga hash mapa čuva svoju maksimalnu i trenutnu veličinu.
Da bi dobili heš vrednost za dati string, treba da implementirate heš funkciju koja slika stringove u brojeve (uglavnom **long**), i korišćenjem te vrednosti ubacijete u strukturu koja podseća na niz, gde vam je indeks izračunata heš vrednost, a vrednost je par string i njegova frekvencija.



Ovde treba pripaziti o pojavi kolizije, odnosno situacija gde dva stringa imaju istu heš vrednost. Ako se to desi, to treba da rešite, a da ne gubite podatke.

Konkurentnost

U aplikaciji će postojati N + 1 nit, gde N predstavlja broj izvršenih `_count_` komandi, a jedna dodatna nit je main nit, iz čijeg konteksta se izvršava pretraga deljene hash map strukture. Sve ove niti treba da rade sa deljenom hash map strukturom uz sledeća ograničenja:

- U nizu skeniranih datoteka treba da se nalaze sve datoteke koje su do sada skenirane, sa njihovim tačnim poslednjim vremenima modifikacije.
- Nije dozvoljeno zaključavati čitavu hash map strukturu.
- Ubacivanje vrednosti, kao i pretraga hash map strukture treba da mogu da se obavljaju konkurentno. Pritom je dozvoljeno da operacije dodavanja ili pretrage koje se rade nad sličnim vrednostima bilo koje dužine to rade uz međusobno isključivanje.

Poenta ove vrste međusobnog isključivanja je da se obezbedi što je više moguće konkurentnog rada, dok i dalje imamo garanciju da pretraga neće da bude prekinuta ubacivanjem, te da proizvede neočekivane rezultate, kao i da dva ubacivanja u istom delu stabla neće da se dešavaju istovremeno, i da se na taj način napravi loša struktura stabla.

Dodavanje stop reči

Pri pozivu korisničkog programa, može da se prosledi i putanja ka fajlu, koji sadrži stop reči koje treba da se filtrira pri pretrazi. Ako se pri pretrazi fajlova, pojavi neka od stop reči ona se ne dodaje u deljenu hash mapu. Struktura fajla je niz stringova u jednom redu razdvojeni *space karakterom.*

Funkcije i strukture

Sledi spisak predloga za strukture i funkcije unutar domaćeg zadatka. Dozvoljeno je i podstiče se izmena svega navedenog onako kako mislite da je prigodno da biste ispunili zahteve u zadatku.

```
#define MAX_WORD_LEN 64 //najveca dozvoljena duzina reci, uz \0
#define LETTERS 26 //broj slova u abecedi

extern void scanner_init(); //poziva se jednom na pocetku rada sistema
extern void *scanner_work(void *_args); //funkcija scanner niti

extern void map_init(); //poziva se jednom na pocetku rada sistema
```

```
extern void map_add_word_count(char *word, int value); //operacija za dodavanje reci i njihove frekvencije
extern search_result *map_get_frequency(char *word); //operacija za pretragu
extern void add_stopword(char* word); //dodaje rec u niz stop reci


typedef struct hm_object //element unutar hash_map strukture
{
	char* key; // reč
	int value; // trenutna frekvencija
} hm_object;

typedef struct search_result //rezultat pretrage
{
	char* key; // reč
	int value; // trenutna frekvencija
} search_result;

typedef struct scanned_file //datoteka koju je scanner vec skenirao
{
	char file_name[256]; //naziv datoteke
	time_t mod_time; //vreme poslednje modifikacije datoteke
} scanned_file;
```

Bodovanje

Zadatak se boduje na sledeći način:

- Konkurentni skeneri - 8 bodova
- Hash map - 6 bodova
  - Konkurentno dodavanje - 3 bodova
  - Konkurentna pretraga - 3 bodova
- Stop reči - 1 bodova

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. Direktorijum koji sadrži kod imenovati na sledeći način: “os_2023_d4_ime_prezime_ind”.
Npr. “os_2023_d4_student_studentic_rn0101”.

Arhivirati ovaj direktorijum (.zip). Naziv arhive mora da bude u obliku:
“os_2023_d4_ime_prezime_ind.zip”
Npr. “os_2023_d4_student_studentic_rn0101.zip”

Rok za predaju je:

- Utorak, 6. jun 23:59:59 za grupu 203.
- Sreda, 7. jun 23:59:59 za grupu 202, 204.
- Petak, 9. jun 23:59:59 za grupe 201, 205.

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.