Programiranje 2

Organizacija memorije

Milena Vujošević Janičić
Jelena Graovac

www.matf.bg.ac.rs/~milena
www.matf.bg.ac.rs/~jgraovac

Beograd, 20. februar, 2020.

Pregled

Pregled

Izvršavanje programa

- Nakon uspešnog prevođenja, program se može izvršiti
- Zahtev za izvršavanjem programa se izdaje operativnom sistemu
- Program koji treba da bude izvršen najpre se učitava iz spoljne memorije u radnu memoriju računara.
- U fazi izvršavanja moguće je da dođe do grešaka koje nije bilo moguće detektovati u fazi prevođenja i povezivanja.

Izvršavanje programa

- Pre konačne distribucije korisnicima, program se obično intenzivno testira
- Testiranjem se pronalaze greške koje je potrebno ispraviti u izvornom kodu
- U pronalaženju greške mogu pomoći programi koji se nazivaju debageri
- Da bi program mogao da bude analiziran primenom debagera, on mora biti preveden na poseban način, tako da izvršna verzija sadrži i informacije o izvornom kôdu (za gcc to je opcija -g).
- Takva verzija izvršnog programa zove se debag verzija, a verzija koja se isporučuje krajnjem korisniku zove se riliz (engl. release).

Organizacija memorije izvršnog programa

- Način organizovanja i korišćenja memorije u fazi izvršavanja programa može se razlikovati od jednog do drugog operativnog sistema.
- Kada se izvršni program učita u radnu memoriju računara, biva mu dodeljena određena memorija i započinje njegovo izvršavanje.
- Dodeljena memorija organizovana je u nekoliko delova koje zovemo segmenti ili zone:
  - segment kôda (engl. code segment ili text segment);
  - segment podataka (engl. data segment);
  - stek segment (engl. stack segment);
  - hip segment (engl. heap segment).

Segmenti u memoriji

Fon Nojmanova arhitektura računara predviđa da se u memoriji čuvaju podaci i programi.

- *Segment kôda* — u ovom segmentu nalazi se sâm izvršni kôd programa, ukoliko se pokrene više instanci jednog programa, ovaj segment može da bude zajednički za sve instance (zavisi od operativnog sistema)
- *Segment podataka* — u ovom segmentu čuvaju se određene vrste promenljivih koje su zajedničke za ceo program: globalne promenljive, promenljive koje imaju statički životni vek, kao i konstantni podaci (najčešće konstantne niske); ukoliko se pokrene više instanci jednog programa, svaka instanca ima svoj zaseban segment podataka
- *Hip segment* — u ovom segmentu nalaze se dinamički alocirani podaci

Stek segment

- *Stek segment* ili programski stek poziva (engl. call stack) čuva sve podatke koji karakterišu izvršavanje funkcija.
- Podaci koji odgovaraju jednoj instanci funkcije organizovani su u takozvani stek okvir (engl. stack frame).
- Stek okvir jedne instance funkcije, između ostalog, sadrži:
  - argumente funkcije;
  - lokalne promenljive (promenljive deklarisane unutar funkcije);
  - međurezultate izračunavanja;
  - adresu povratka (koja ukazuje na to odakle treba nastaviti izvršavanje programa nakon povratka iz funkcije);
  - adresu stek okvira funkcije pozivaoca.

Stek segment

- Veličina stek segmenta obično je ograničena.
- Zbog toga je poželjno izbegavati smeštanje jako velikih podataka na segment steka.

Primer

```
int main() {
  int a[1000000];
  ...
}
```

```
int a[1000000];
  int main() {
  ...
}
```

Stek segment

- Stek poziva je struktura tipa LIFO ("last in - first out") — stek okvir se može dodati samo na vrh steka i sa steka se može ukloniti samo okvir koji je na vrhu
- Stek okvir za instancu funkcije se kreira onda kada funkcija treba da se izvrši i taj stek okvir se oslobađa (preciznije, smatra se nepostojećim) onda kada se završi izvršavanje funkcije.

Stek segment

- Ako izvršavanje programa počinje izvršavanjem funkcije main, prvi stek okvir se kreira za ovu funkciju.
- Ako funkcija main poziva neku funkciju f, na vrhu steka, iznad stek okvira funkcije main, kreira se novi stek okvir za ovu funkciju.
- Ukoliko funkcija f poziva neku treću funkciju, onda će za nju biti kreiran stek okvir na novom vrhu steka.
- Kada se završi izvršavanje funkcije f, onda se vrh steka vraća na prethodno stanje i prostor koji je zauzimao stek okvir za f se smatra slobodnim (iako on neće biti zaista obrisan).

Stek segment

Nacrtati stek okvire za izvršavanje sledećeg koda:

Primer

```
int kub(int a) {
  return a*a*a;
}
int suma_kubova_i_uvecanje(int a[], int n) {
  int rezultat = 0, i;
  for(i=0; i<n; i++)
    rezultat += kub(a[i]++);
  return rezultat;
}
int main() {
  int a[3]={0, 1, 2}, s;
  s = suma_kubova_i_uvecanje(a, sizeof(a)/sizeof(int));
  printf("Rezultat je %d\n", s);
  return 0;
}
```

Pregled

Literatura

- Slajdovi su pripremljeni na osnovu knjige
  Predrag Janičić, Filip Marić: Programiranje 2
- Za pripremu ispita, slajdovi nisu dovoljni, neophodno je učiti iz knjige!