Operativni sistemi
2010

Osnove Linux-a

System programming

Katedra za računarstvo
Elektronski fakultet u Nišu

Prof. dr Dragan Stojanović
mr Aleksandar Stanimirović
mr Bratislav Predić

Sadržaj

- Programski jezik C
- Sistemsko programiranje
- Sistemski pozivi
- Razvoj C programa

Sadržaj

- Programski jezik C
- Sistemsko programiranje
- Sistemski pozivi
- Razvoj C programa

Programski jezik C

Značaj

- C je jedan od najznačajnijih programskih jezika koji su razvijeni tokom istorije računarske nauke.
- C je jezik koji je danas široko prihvaćen kao jedan od glavnih jezika za sistemsko programiranje ali i za razvoj aplikacija.
- Dominantan jezik za sistemsko programiranje na UNIX/Linux platformi. Najveći deo jezgra Linux-a je razvijen korišćenjem C-a.

Karakteristike

- C je tradicionalni proceduralni jezik.
- Osnovne karakteristike programskog jezika C su: jednostavnost, efikasnost, fleksibilnost i mali memorijski zahtevi.

Programski jezik C

Karakterisitke

- Podrška za C programski jezik postoji na velikom broju različitih platformi što obezbeđuje portabilnost koda.
- Programski jezik C spada u grupu jezika niskog nivoa (low level).

Istorija

- Razvijen 70-tih godina 20 veka u Bell laboratorijama. Tvorac: Dennis Ritchie.
- Jezgro UNIX-a je 1973 godine kompletno prepisano korišćenjem C-a čime je UNIX postao prvi OS koji nije razvijen isključivo u asembleru.
- ANSI C – standardna specifikacija programskog jezika C:
  - ANSI X3.159-1989 Programming Language C – ANSI C
  - ISO 9899:1999 – C99

Sadržaj

- Programski jezik C
- Sistemsko programiranje
- Sistemski pozivi
- Razvoj C programa

Sistemsko programiranje

Sistemski softver

- Sistemsko programiranje predstavlja aktivnost razvoja sistemskog softvera.
- Za razliku od aplikativnog programiranja sistemsko programiranje zahteva dobro poznavanje okruženja (hardvera i operativnog sistema) na kome će se softver izvršavati.
- Veoma je teško napraviti jasnu podelu između sistemskog i aplikativnog softvera.
- Tradicionalno sistemski softver podrazumeva jezgro operativnog sistema i drajvere uređaja.
- Pored toga sistemski softver može da obuhvata i veliki broj alata koji se razvijaju za potrebe korisnika i ne moraju direktno da komuniciraju sa hardverskom komponentnom sistema.

Sistemsko programiranje

Karakterisitke sistemskog programiranja

- Dobro poznavanje sistema kako bi se što bolje iskoristile njegove karakteristike.
- Korišćenje programskog jezika niskog nivoa:
  - na raspolaganju su ograničeni resursi
  - zahteva se velika efikasnost
  - runtime biblioteka je veoma ograničena ili uopšte ne postoji
  - direktan pristup memoriji
  - kombinovanje sa delovima koda koji su razvijeni korišćenjem asemblera
- Otklanjanje grešaka (debugging) može biti jako komplikovano. Program koji se razvija često mora da se izvršava u simuliranom okruženju.
- Programski jezik ispunjava sve postavljene zahteve.

Sadržaj

- Programski jezik C
- Sistemsko programiranje
- Sistemski pozivi
- Razvoj C programa

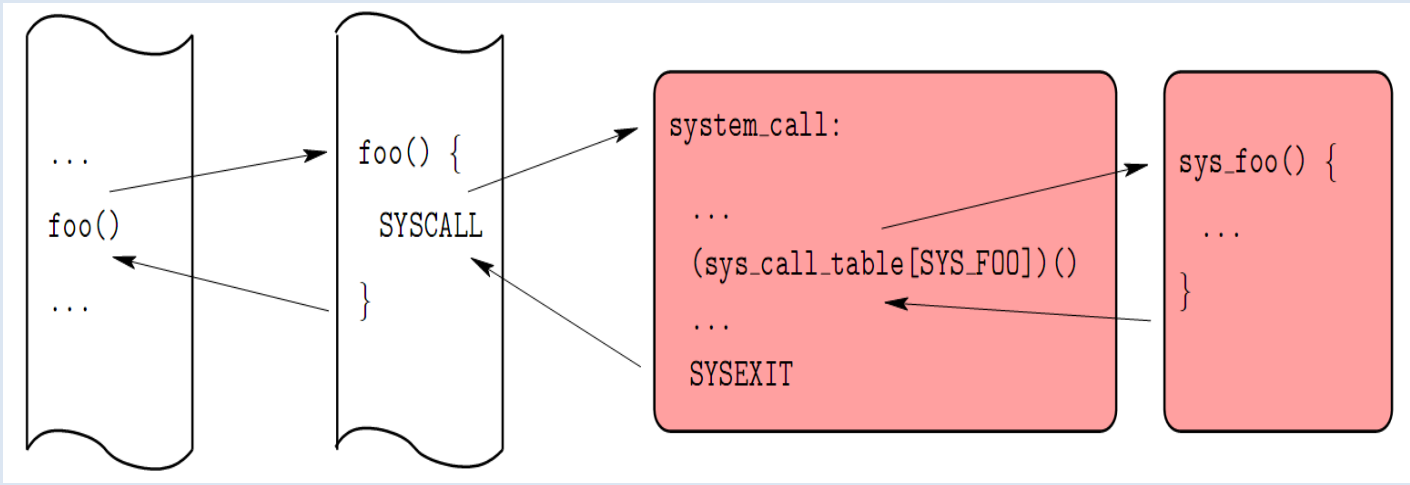
Sistemski pozivi

Pojam

- Sistemsko programiranje započinje sa sistemskim pozivima.
- Korisnički programi i jezgro operativnog sistema se izvršavaju u dva različita režima:
  - režim jezgra (kernel mode) – softver se izvršava bez ograničenja
  - korisnički režim (user mode) – postoje ogrničenja koja se tiču kako dozvoljenih instrukcija tako i dozvoljenih memorijskih regiona
- Korisnička aplikacija prelazi u režim jezgra kada zahteva servise operativnog sistema: sistemski poziv.

Sistemski pozivi

Sistemski poziv



Korisnički režim

Režim jezgra

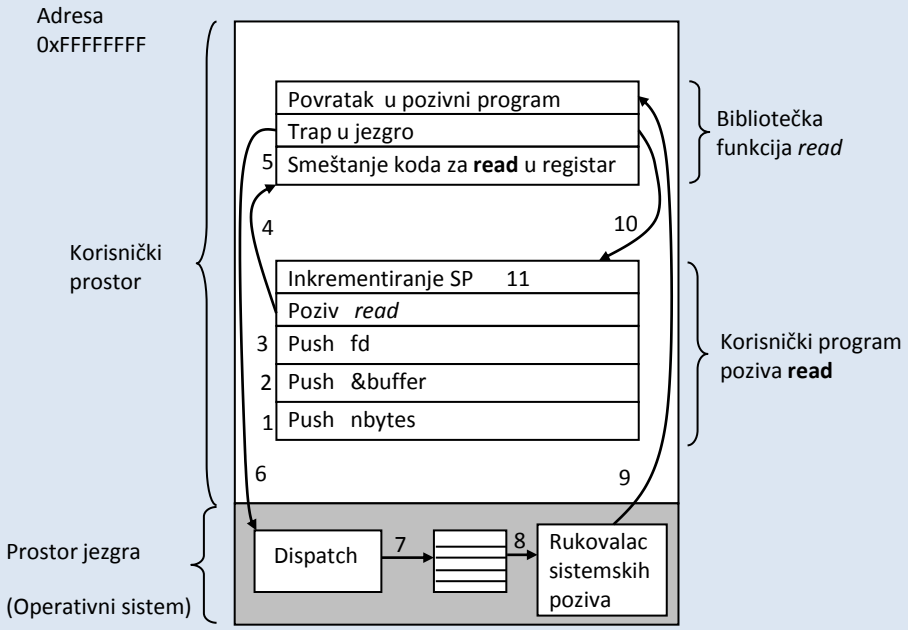
Sistemski pozivi

Promena režima

- Nemoguće je direktno povezati korisničku aplikaciju sa jezgrom operativnog sistema.
- Iz razloga pouzdanosti i bezbednosti nije dozvoljeno da korisnička aplikacija direktno izvršava kod jezgra ili da manipuliše objektima u jezgru sistema.
- Umesto toga jezgro operativnog sistema obezbeđuje mehanizam kojim korisnička aplikacija signalizira da želi da pozove sistemski poziv.
- Nakon signalizacije aplikacija prelazi u režim jezgra i izvršavaju se samo oni servisi za koje jezgro dozvoli aplikaciji da ih izvrši.
- Promena režima izvršavanja aplikacije moguća je samo kroz ovaj dobro definisani mehanizam.
- Implementacija ovog mehanizma zavisi od arhitekture sistema.

Sistemski pozivi

Primer: Sistemski poziv read



Sistemski pozivi

Povratna vrednost i greške

- Sistemski poziv uvek vraća vrednost tipa int ili long (često se zbog portabilnosti koriste POSIX wrapper tipovi).
- Povratna vrednost sistemskog poziva može biti:
  - = -1 – ukoliko je došlo do greške
  - >= 0 – izvršenje sistemskog poziva je proteklo normalno.
- Ukoliko se desi greška numerički kod greške se smešta u promenljivu errno.
- Svaki proces poseduje promenljivu errno koja je na početku postavljena na vrednost 0.
- Deklarisana je u zaglavlju <errno.h>
- Karakteristike:
  - globalna celobrojna promenljiva
  - jedini način za dobijanje detaljnijih informacija o grešci
  - resetuje se vrednost tek pozivanje drugog sistemskog poziva

Sistemski pozivi

perror

- Bibliotečka funkcija koja konvertuje tekuću vrednost promenljive errno u opis greške na engleskom jeziku.

```
#include <stdio.h>
void perror(const char *str)
```

Primeri

errno	Značenje
EPERM	Operation not permitted
ENOENT	No such file or directory
ESRCH	No such process
EINTR	Interrupted system call
EIO	I/O error
ECHILD	No child process
EACCESS	Access permission denied
EAGAIN/EWOULDBLOCK	Resource temporarily unavailable

Sistemski pozivi

POSIX

- POSIX – Portable Operating System Interface
- Kolekcija povezanih IEEE standarda koji definišu zajednički programski interfejs za različite varijante UNIX operativnog sistema.
- POSIX definiše standardne interfejse ka funkcijama operativnog sistema.
- POSIX 1003.1 specificira jedinstveni bibliotečki interfejs za sve verzije UNIX operativnog sistema.
- Pored operativnog sistema i biblioteke sistemskih poziva, UNIX operativni sistemi uključuju i veliki broj standardnih programa (POSIX 1003.2)
  - komandni interfejs (shell)
  - programski prevodioci
  - editori
  - programi za manipulaciju datotekama.

Sadržaj

- Programski jezik C
- Sistemsko programiranje
- Sistemski pozivi
- Razvoj C programa

Razvoj C programa

Šta je neophodno?

- Tekstualni editor – koristi se razvoj programa odnosno source koda. Tekst editor omogućava editovanje čisto tekstualnih dokumenata. Postoji veliki broj različitih editora koji mogu raditi u tekstualnom režimu (vi, joe, pico, emcs, ...) ili grafičkom režimu (gedit, emacs, Kwrite, Kate, ..).
- Prevodilac – specijalizovani program koji kod programa prevodi u mašinske instrukcije koje CPU direktno izvršava. Za prevođenje C programa može se koristiti gcc (GNU Compiler Collection). .
- C standardna biblioteka – kolekcija standardnih C funkcija bez kojih je razvoj programa nemoguć. glibc predstavlja predstavlja implementaciju C standardne biblioteke i sastavni je deo GNU projekta.

Razvoj C programa

Koraci u razvoju aplikacije

1. Kreiranje nove tekstualne datoteke korišćenjem nekog od tekstualnih editora.
2. Pisanje C programa (Primer: jednostavna **Hello World aplikacija**).

```
#include <stdio.h>

main(int argc, char * argv[])
{
	printf("Hello, world!\n");
	return 0;
}
```

Razvoj C programa

Koraci u razvoju aplikacije

3. Kreirani program se snima pod proizvoljnim imenom (treba izbegavati imena koja odgovaraju komandama operativnog sistema). Ime programa treba obavezno da ima ekstenziju .c (Primer: **hello.c**). U suprotnom prevodilac neće biti u stanju da program prevede.
4.Program se prevodi korišćenjem gcc prevodioca.

gcc -o hello hello.c

5.Uklanjanje grešaka ukoliko postoje i eventualno ponovno prevođenje programa.
6.Izvršavanje programa.

./hello

Razvoj C programa

Alati

- tekstualni editor – može se koristiti bilo koji editor
- gcc – C prevodilac koji je sastavni deo svih Linux distribucija (nekad nije uključen u instalaciju po default'u pa ga je potrebo naknadno dodati)
- make – alat koji se koristi za razvoj kompleksnijih projekata korišćenjem programskog jezika C
- gdb – GNU debugger koji olakšava lociranje i otklanjanje grešaka.