Operativni sistemi
2011

Sistemsko programiranje

Upravljanje procesima i nitima

Katedra za računarstvo
Elektronski fakultet u Nišu

Prof. dr Dragan Stojanović
mr Aleksandar Stanimirović
mr Bratislav Predić

Sadržaj

- Procesi
- Sistemski pozivi za upravljanje procesima
- Primer
- Niti
- Sistemski pozivi za upravljanje nitima
- Primer

Sadržaj

- Procesi
- Sistemski pozivi za upravljanje procesima
- Primer
- Niti
- Sistemski pozivi za upravljanje nitima
- Primer

Procesi

Pojam

- Proces je u osnovi program koji se izvršava.
- Svaki proces koji se izvršava na UNIX / Linux sistemu ima jedinstveni identifikator (PID).
- Svaki proces ima svoj nezavistan adresni prostor i (obično) jednu nit izvršenja.
- Na sistemu može postojati više procesa koji izvršavaju isti program, ali oni su potpuno nezavisni i imaju zasebne kopije programa i podatke u razdvojenim adresnim prostorima.
- Svaki proces sadrži: programski kod, podatke, magacin i pid.
- Po startovanju sistema postoji samo jedan proces, init, čiji je pid 1.

Procesi

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char * argv[])
{

   // KOD PROCESA

}
```

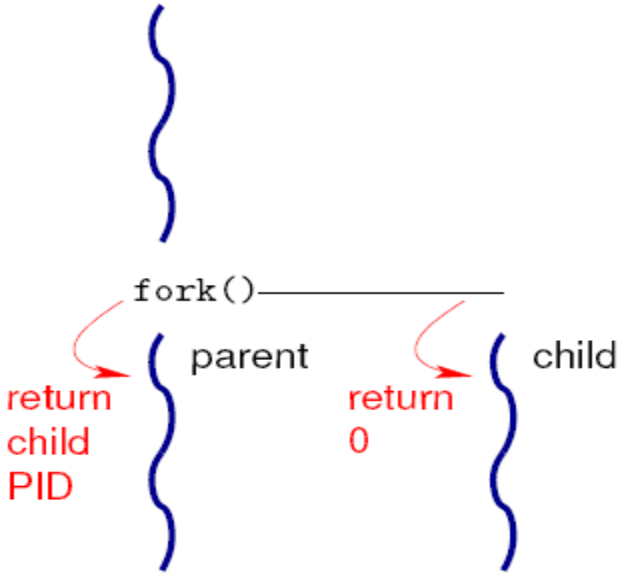
Procesi

Kreiranje novog procesa

- Kod Linux-a se za kreiranje novog procesa koristi tehnika dupliranja postojećeg procesa.
- Nakon dupliranja novi proces je potpuno identičan starom po svemu osim po PID-u i po adresnom prostoru.
- Procesi su u sistemu organizovani hijerarhijski. Svaki proces ima roditelja koji ga je kreirao izvršavanjem odgovarajućeg sistemskog poziva. Svi procesi koje je neki proces kreirao se nazivaju decom tog procesa.
- Pošto procesi deca nasleđuju sve atribute od roditeljskog procesa, osim PID-a i adresnog prostora, nasleđuju i programski kod pa proces dete izvršava isti kod, i to ne od početka, već od linije koja sledi iza sistemskog poziva za kreiranje novog procesa.
- Proces može u potpunosti zameniti svoj programski kod i započeti sa izvršavanjem nekog drugog programa korišćenjem odgovarajućeg sistemskog poziva.

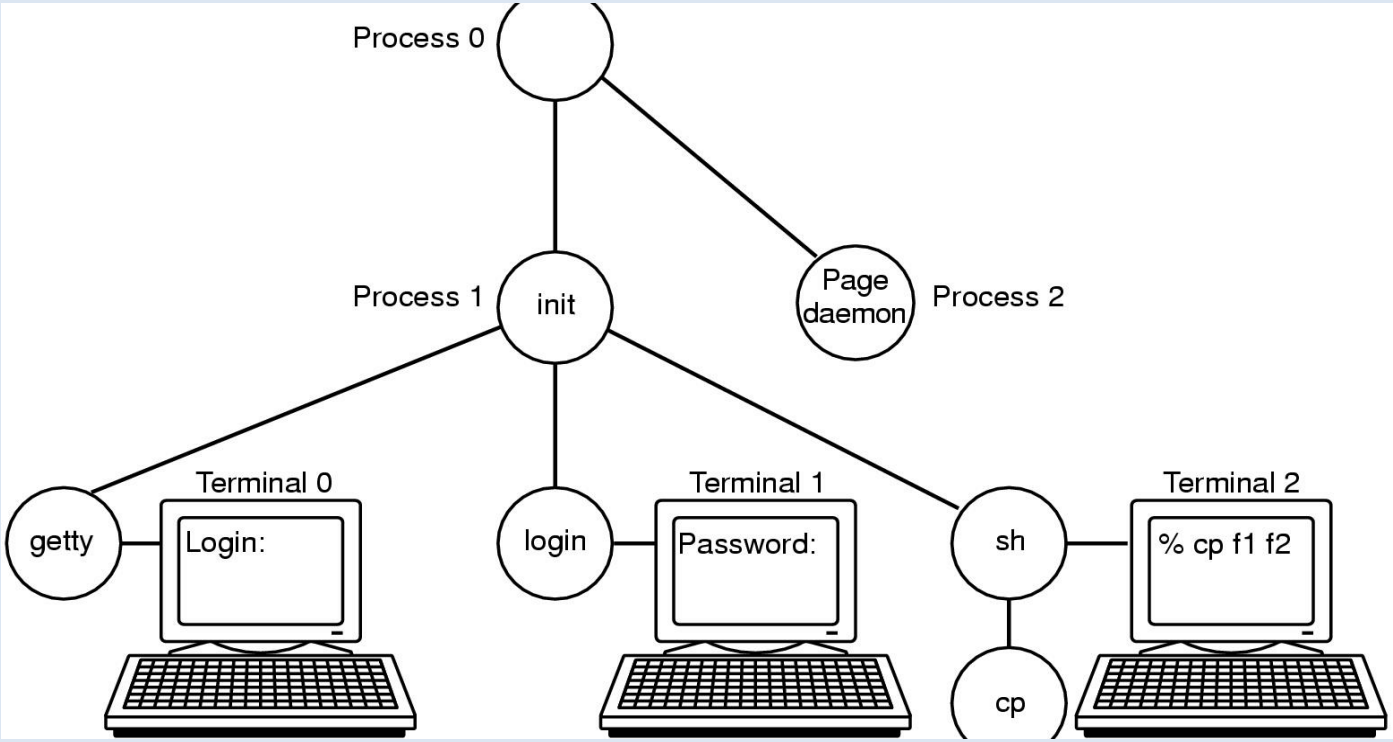
Procesi

Dupliranje procesa



Procesi

Hijerarhija procesa



Sadržaj

- Procesi
- Sistemski pozivi za upravljanje procesima
- Primer
- Niti
- Sistemski pozivi za upravljanje nitima
- Primer

Sistemski pozivi

Kreiranje novog procesa

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork();
```

Semantika

- Nakon kreiranja proces dete je identičan roditeljskom procesu osim u pogledu:
  - Različiti PID i PPID (PID roditeljskog procesa)
  - Različit adresni prostor
  - Proces dete poseduje kopije svih resursa koje je proces roditelj koristio (copy – on – write)
- U slučaju uspeha u roditeljskom procesu vraća PID deteta a u procesu detete vraća 0. Time se na jednostavan način može odrediti da li kod izvršavaju proces dete ili proces roditelj.
- U slučaju greške vraća -1.

Sistemski pozivi

Tipičan način korišćenja

```
switch (cpid = fork())
{
   case -1:                          // Greška
      perror("Funkcija fork nije uspela");
      exit(1);
   case 0:                           // Izvršava se proces dete
      continue_child();
      break;
   default:                          // Izvršava se proces roditelj
      continue_parent(cpid);
}
```

Sistemski pozivi

PID tekućeg procesa

```
#include <sys/types.h>
#include <unistd.h>
pid_t getpid();
```

PID roditeljskog procesa

```
#include <sys/types.h>
#include <unistd.h>
pid_t getppid();
```

Sistemski pozivi

Suspendovanje izvršavanja procesa

```
#include <sys/types.h>
#include <unistd.h>
unsigned int sleep(unsigned int seconds);
```

Semantika

- Izvršavanje procesa se suspenduje zadati broj sekundi.
- Nakon isteka specificiranog vremena proces normalno nastavlja svoje izvršenje od prve naredne linije koda.

Sistemski pozivi

Završetak procesa

```
#include <stdlib.h>
void exit(int status);
```

Semantika

- Završava se izvršavanje procesa koji je pozvao funkciju:
  - zatvaraju se sve otvorene datoteke
  - oslobađa se zauzeta memorija
  - obaveštava se roditeljski proces i prosleđuje mu se exit kod
  - procesi deca za roditeljski proces dobijaju proces init (PID = 1)
- Nikad ne dolazi do greške prilikom poziva funkcije exit i funkcija ne vraća vrednost.
- Argument status definiše exit kod procesa (status kojim se završio).
- Uobičajeno je da 0 označava da se proces normalno završio a negativna vrednost da je došlo do greške.
- Ne postoji standardna lista statusa koji se mogu primenjivati.
- Status procesa se može dobiti korišćenjem wait funkcije.

Sistemski pozivi

Status procesa

```
#include <sys/types.h>
#include <unistd.h>
pid_t wait ( int *status-ptr )
pid_t waitpid ( pid_t pid, int *status-ptr )
```

Semantika

- Proces koji je pozvao funkciju se zaustavlja dok se ne završi proces dete.
- Funkcija wait zaustavlja proces dok se ne završi bilo koji proces dete, a funkcija waitpid dok se ne završi proces dete čiji je PID specificiran.
- Promenljiva **status-ptr** prihvata exit kod sa koji se završio proces dete.

Sistemski pozivi

Izvršavanje programa

```
#include <sys/types.h>
#include <unistd.h>

int execl(const char *path, const char *arg, ...);
int execv(const char *path, char *const argv[]);
int execlp(const char *file, const char *arg, ...);
int execvp(const char *file, char *const argv[]);
int execle(const char *path, const char * arg, ..., char *const envp[]);
int execve(const char *filename, char *const argv[], char *const envp[]);
```

Sistemski pozivi

Semantika

- Argumenti: putanja do izvršne datoteke, argumenti poziva programa i vrednosti environment promenljivih (SHELL sistemske promenljive)
- U slučaju uspeha funkcija se ne vraća u proces koji ju je pozvao (započinje izvršavanje novog programa):
  - menja se kod procesa koji je pozvao funkciju kodom iz izvšne datoteke koja se učitava
  - zadržavaju se vrednosti PID i PPID
  - zadržavaju se sve otvorene datoteke
- U slučaju greške funkcija vraća -1.

Sistemski pozivi

Greške

- Tipične vrednosti promenljive errno koje se mogu javiti u slučaju greške:
  - EACCESS – korisnik nem aprivilegije da izvrši specificiranu datoteku
  - ENOEXEC – specificirana datoteka nije u izvršnom formatu

Argumenti

- Funkcija execl očekuje listu argumenata koja se završava sa NULL.
- Funkcija execv očekuje niz (vektor) argumenata.
- Prvi argument odgovara argv[0] i mora da bude ime programa koji se poziva.
- Funkcije execlp i execvp omogućavaju pretraživanje izvršnih datoteka u PATH putanji (ukoliko ime izvršne datoteke ne sadrži / a u suprotnom se ponašaju kao standardne execl i execv funkcije).
- Funkcije execle i execve omogućavaju specificiranje vrednosti environment promenljivih.

Sadržaj

- Procesi
- Sistemski pozivi za upravljanje procesima
- Primer
- Niti
- Sistemski pozivi za upravljanje nitima
- Primer

Primer

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

main ( )
{
   int   pid ;

   printf ("Ja sam originalni proces sa PID %d i PPID %d.\n", getpid ( ), getppid ( ) ) ;

   pid = fork ( ) ;

   if ( pid != 0 )
   {
      printf ("Ja sam roditeljski proces sa PID %d i PPID %d.\n", getpid ( ), getppid ( ) ) ;
      printf ("Pid procesa koji je moje dete je: %d\n", pid ) ;
   }
   else
   {
      sleep (4) ; // Da se osiguramo da roditeljski proces završi prvi
      printf ("Ja sam proces dete sa PID %d i PPID %d.\n", getpid ( ), getppid ( ) ) ;
    }
    printf ("PID %d izlazi.\n", getpid ( ) ) ;
}
```

Duplira se postojeći proces. Nakon ove linije roditelj i dete nastavljaju svoje izvršavanje.

PID je različit od nule. Ovaj deo koda izvršava roditelj.

PID je nula. Ovaj deo koda izvršava dete.

Primer

- Procesi
- Sistemski pozivi za upravljanje procesima
- Primer
- Niti
- Sistemski pozivi za upravljanje nitima
- Primer

Niti

Pojam

- Niti (threads) su u principu delovi izvršenja jednog procesa.
- Niti koje se izvršavaju u okviru jednog procesa dele memoriju i mogu pristupati zajedničkim globalnim promenjivima.
- Niti se izvršavaju paralelno, odnosno u različitim dodeljenim vremenskim intervalima na jednoprocesorskim sistemima ili stvarno istovremeno na višeprocesorskim sistemima.
- Potencijalne prednosti korišćenja niti su vremensko preklapanje izvršenja operacija koje su vremenski zahtevne. Na primer, ulazno/izlaznih operacija koje po principu zavise od sporih uređaja kod kojih bi procesor većinu vremena proveo čekajući i intenzivnih računskih operacija.

Niti

Razvoj višenitnih programa

- Za rad sa nitima neophodno je uključiti zaglavlje **<pthread.h>** u izvorni kod programa.
- Prilikom prevođenja programa neophodno je linkeru dodati opciju **–lpthread**.
- Komandna linija za prevođenje programa threads.c ima sledeći izgled:

```
gcc threads.c -o threads –lpthread
```

Sadržaj

- Procesi
- Sistemski pozivi za upravljanje procesima
- Primer
- Niti
- Sistemski pozivi za upravljanje nitima
- Primer

Sistemski pozivi

Kreiranje niti

```
#include <pthread.h>
int pthread_create ( pthread_t *threadhandle, pthread_attr_t *attribute,
                     void *(*start_routine)(void*), void *arg ) ;
```

Semantika

- Funkcija započinje izvršavanje nove niti.
- Argumenti:
  - *handle* novokreirane niti ukoliko se funkcija uspešno izvrši
  - argumenti za kreiranje niti (mogu imati NULL vrednost)
  - pokazivač na funkciju koja sadrži programski kod koji će niti zvršavati. Funkcija mora da vraća void * i da ima jedan ulazni argument koji je void *.
  - pokazivač na ulazni argument niti

Sistemski pozivi

Čekanje niti

```
#include <pthread.h>
int pthread_join ( pthread_t threadhandle, void **returnvalue )
```

Semantika

- Nit koja je pozvala funkciju pthread_join se zaustavlja i čeka da se završi nit koja je završena svojim handle-om.

Sadržaj

- Procesi
- Sistemski pozivi za upravljanje procesima
- Primer
- Niti
- Sistemski pozivi za upravljanje nitima
- Primer

Primer

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int glob_data = 5 ;

void * kidfunc(void  *p)
{
   printf ("Nova nit. Globalni podatak je: %d.\n", glob_data) ;
   printf ("Nova nit. ID procesa je: %d.\n", getpid()) ;

  glob_data = 15 ;

 printf ("Ponovo nova nit. Globalni podatak je: %d.\n", glob_data) ;
}
main ( )
{
   pthread_t kid ;
   pthread_create (&kid, NULL, kidfunc, NULL) ;

   printf ("Osnovna nit. Globalni podatak je: %d\n", glob_data) ;
   printf ("Osnovna nit. ID procesa je: %d.\n", getpid()) ;

   glob_data = 10 ;
   pthread_join (kid, NULL) ;

   printf ("Kraj programa. Globalni podatak je: %d\n", glob_data) ;
}
```

Funkcija koju će nit izvršavati. Povratna vrednost je void *, ima jedan ulazni argument tipa void *.

Kreiranje niti.

Čeka se kraj kreirane niti.