Sloj za Rukovanje Procesima

POGLAVLJE 9

Sloj OS za rukovanje procesima

- Osnovni zadaci sloja za rukovanje procesima su:
  - **stvaranje** procesa
  - **uništenje** procesa.

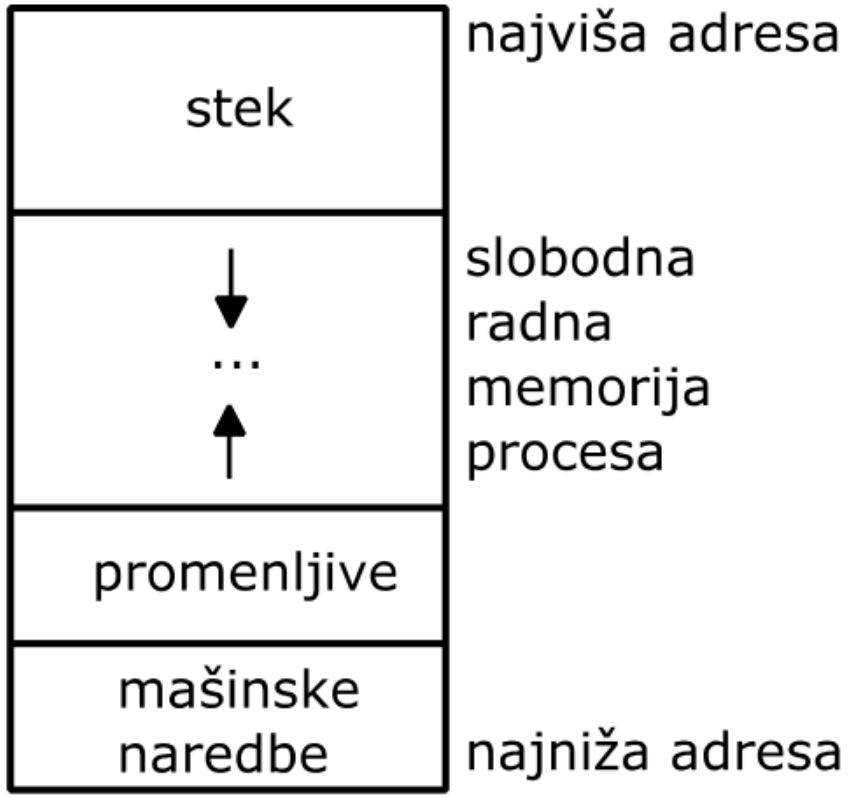
- **Stvaranje** procesa obuhvata:
  - stvaranje **slike** procesa
  - stvaranje **deskriptora** procesa
  - **pokretanje aktivnosti** procesa

- **Uništenje** procesa obuhvata:
  - **zaustavljanje aktivnosti** procesa
  - uništenje **slike** procesa
  - uništenje **deskriptora** procesa

Osnovni zadaci sloja za rukovanje procesima

- Pored sistemskih operacija stvaranja i uništenja procesa, potrebne su i sistemske operacije za **izmenu atributa procesa**, na primer, za **izmenu radnog imenika (direktorijuma)** procesa.
- Slika procesa obuhvata niz lokacija radne memorije sa uzastopnim (logičkim) adresama. Ona sadrži:
  - **izvršavane mašinske naredbe**
  - **promenljive**
  - **stek**

Osnovni zadaci sloja za rukovanje procesima



Osnovni zadaci sloja za rukovanje procesima

- Slika procesa počinje od lokacije radne memorije sa **najnižom** adresom → **mašinske naredbe**, a završava na lokaciji radne memorije sa **najvišom** adresom → **stek**.
- **stek** se puni u **smeru nižih adresa**
- Iza mašinskih naredbi dolaze **statičke promenljive** (**inicijalizovane**, pa **neinicijalizovane**).
- Između ovih promenljivih i steka se nalazi **slobodna radna memorija** procesa.

Osnovni zadaci sloja za rukovanje procesima

- Radna memorija procesa je na raspolaganju procesu za:
  - **širenje** (**punjenje**) **steka**
  - za stvaranje **dinamičkih promenljivih** (**heap**)
- Svi dinamički zahtevi za zauzimanjem radne memorije, postavljeni u toku aktivnosti procesa, se zadovoljavaju samo na **račun slobodne radne memorije procesa**.
- Ovakva organizacija slike procesa uslovljava da proces prvo **ugrozi svoju aktivnost**, kada njegovi zahtevi za radnom memorijom **nadmaše** njegovu **raspoloživu slobodnu** radnu memoriju (**preklapanja steka i promenljvih** dovodi do **fatalnog** ishoda po aktivnost procesa)

Osnovni zadaci sloja za rukovanje procesima

- Pored slike, za aktivnost procesa je važan i **deskriptor** procesa, koji sadrži **atribute** procesa.
- Ovi atributi karakterišu **aktivnost** procesa. Oni obuhvataju:
  - **stanje** procesa ("spreman", "aktivan", "čeka")
  - sadržaje **procesorskih registara** (zatečene u njima pre poslednjeg preključivanja procesora sa procesa)
  - **numeričku oznaku** vlasnika procesa
  - **oznaku** procesa stvaraoca
  - **trenutak pokretanja** aktivnosti procesa

Osnovni zadaci sloja za rukovanje procesima

- **ukupno trajanje aktivnosti** procesa (odnosno, ukupno vreme angažovanja procesora)
- podatke o **slici procesa** (njenoj **veličini** i njenom **položaju** u radnoj i masovnoj memoriji)
- podatke o **datotekama** koje proces koristi
- podatak o **radnom imeniku** procesa
- razne podatke neophodne za upravljanje aktivnošću procesa (poput **prioriteta procesa** ili **položaja sistemskog steka** procesa, koga koristi operativni sistem u toku obavljanja sistemskih operacija).

Sistemske operacije za stvaranje i uništenje procesa

- Za stvaranje procesa potrebno je pristupiti odgovarajućoj **izvršnoj datoteci** sa **inicijalnom slikom** procesa. Ona sadrži:
  - **mašinske naredbe**
  - **početne vrednosti statičkih promenljivih**
  - podatak o **veličini** delova **slike procesa**.
- Takođe, potrebno je zauzeti **deskriptor procesa**, kao i **dovoljno veliku zonu radne memorije** za **sliku procesa**.
- Sve ovo spada u nadležnost **sistemske operacije stvaranja procesa** (**fork()** i **exec()**).
- Ovu operaciju poziva proces **stvaralac** i ona se obavlja u toku njegove aktivnosti.

Sistemske operacije za stvaranje i uništenje procesa

- U okviru poziva sistemske operacije **stvaranja procesa** kao argument se navodi putanja odgovarajuće **izvršne datoteke**.
- **Svi atributi** stvaranog procesa **ne moraju biti navedeni** u okviru poziva sistemske operacije stvaranja procesa, jer se jedan njihov deo **nasleđuje iz deskriptora** procesa **stvaraoca:**
  - **numerička oznaka vlasnika** procesa
  - podatak o **radnom imeniku** procesa
  - njegov **prioritet**

Sistemske operacije za stvaranje i uništenje procesa

- Kada se, u okviru **stvaranja** procesa, stigne do **pokretanja** njegove aktivnosti, moguće je **preključivanje** procesora **sa procesa stvaraoca na stvarani proces**.
- To se desi, ako je prioritet stvaranog procesa **viši** od prioriteta procesa stvaraoca.
- U tom slučaju, proces stvaralac dospeva među **spremne procese**. Inače tamo dospeva **stvarani proces**.

Sistemske operacije za stvaranje i uništenje procesa

- Za **uništenje** procesa potrebno je **osloboditi njegov deskriptor** i **zonu radne memorije** sa njegovom slikom što spada u nadležnost sistemske operacije uništenja procesa (**exit()**).
- Nju automatski poziva proces na kraju svoje aktivnosti, čime izaziva svoje **samouništenje**.
- Uništenje procesa se završava **preključivanjem procesora** sa **uništavanog** na neki od **spremnih** procesa.

Sistemske operacije za stvaranje i uništenje procesa

- U okviru operacije uništenja procesa uputno je predvideti **argument** kojim proces saopštava svoje **završno stanje**, odnosno informaciju da li je aktivnost uništavanog procesa bila **uspešna ili ne**.
- Da bi proces **stvaralac** mogao iskoristiti ovakvu **povratnu informaciju** od **stvorenog** procesa, on mora pozivom posebne sistemske operacije (**wait()**) da zatraži zaustavljanje svoje aktivnosti i tako omogući **preključivanje** procesora na **stvarani** proces.

Sistemske operacije za stvaranje i uništenje procesa

- U ovom slučaju, proces **stvaralac** ne dospeva među **spremne** procese, nego među procese u stanju **čekanja**, jer čeka kraj aktivnosti stvorenog procesa.
- Takođe, u ovom slučaju sistemska operacija uništenja stvorenog procesa ima i zadatak da prevede proces stvaralac među **spremne procese** i tako omogući **nastavak njegove aktivnosti**.
- Sistemska operacija uništenja procesa se **automatski poziva**, kada se desi **nepopravljiv prekid** (**izuzetak**) u toku aktivnosti procesa.

Zamena slika procesa

- Najveći mogući broj slika procesa, koje mogu da istovremeno postoje u radnoj memoriji, se naziva **stepen multiprogramiranja** (**degree of multiprogramming**).
- Što je stepen multiprogramiranja **viši**, to je i veća verovatnoća da je procesor **zaposlen**, jer je veća verovatnoća da postoji **spreman proces**.
- Stepen multiprogramiranja zavisi od **veličine radne memorije**.
- Kada broj istovremeno postojećih procesa dostigne **stepen multiprogramiranja**, stvaranje novih procesa **postaje problematično**.
- U ovoj situaciji moguće rešenje je da se slika nekog od (**manje prioritetnih**) **postojećih procesa** privremeno izbaci u **masovnu memoriju** i tako oslobodi prostor u radnoj memoriji za sliku novog procesa.

Zamena slika procesa

- Predloženi pristup podrazumeva da je broj **deskriptora procesa veći od stepena multiprogramiranja** i da su svi deskriptori **stalno prisutni** u radnoj memoriji.
- Prethodno opisani način oslobađanja prostora za sliku procesa u radnoj memoriji uzrokuje da, uz sliku procesa u radnoj memoriji, obavezno postoji i njena **kopija u masovnoj memoriji**.
- Pošto se, u toku aktivnosti procesa, menja samo **deo** njegove **slike** u radnoj memoriji, jer se menjaju samo vrednosti njegovih **promenljivih** i njegov **stek**, prilikom izbacivanja slike procesa potrebno je samo njen **izmenjeni deo** prebacivati u kopiju slike u masovnoj memoriji.

Zamena slika procesa

- Ali, pri **vraćanju** slike u radnu memoriju, prebacuje se **cela njena kopija**, da bi se u radnoj memoriji obnovila **cela slika procesa**.
- Do vraćanja slike procesa u radnu memoriju dolazi, kada se u njoj **oslobodi** prostor, pa je **vraćanje** slike **jednog** procesa vezano za **uništavanje drugog** procesa, jer se tada oslobađa prostor u radnoj memoriji.
- Podaci o **kopiji slike procesa** (koja **nastaje** istovremeno sa **nastankom procesa**, odnosno sa nastankom njegove slike, ili prilikom njenog **prvog izbacivanja**, a **nestaje** istovremeno sa **nestankom procesa**, odnosno sa nestankom njegove slike) se čuvaju u **deskriptoru procesa**, zajedno sa podacima o slici procesa.

Zamena slika procesa

- Prilikom izbacivanja slike procesa, u deskriptoru procesa se naznačava da se njegova slika **ne nalazi** više u radnoj memoriji.
- Da procesi, čije su slike izbačene van radne memorije, ne bi bili dugo **zapostavljeni**, uputno je periodično vršiti **zamenu slika procesa** (**swapping**), znači **izbacivati** sliku **jednog** procesa, radi **ubacivanja** slike **drugog** procesa.

Zamena slika procesa

- U nadležnosti ove operacije **zamene** (**swap**) je **dugoročno raspoređivanje** (**long term scheduling**), u okviru koga se odabira proces, čija slika se **izbacuje**, kao i proces, čija slika se **ubacuje**.
- Važno je uočiti da se dugoročno raspoređivanje razlikuje od **običnog** ili **kratkoročnog** raspoređivanja (**short term scheduling**), koje među **spremnim** procesima odabira proces na koga se preključuje procesor.

Rukovanje nitima

- Rukovanje nitima **može**, ali i **ne mora**, biti u nadležnosti **sloja za rukovanje procesima**.
- Kada je rukovanje nitima povereno sloju za **rukovanje procesima**, tada operativni sistem nudi sistemske operacije za rukovanje nitima, koje omogućuju **stvaranje**, **uništavanje** i **sinhronizaciju niti**.
- U ovom slučaju, **deskriptori** i **sistemski stek** niti se nalaze u **sistemskom prostoru**, dok se sopstveni **stek** niti nalazi u **korisničkom prostoru** (**unutar slike procesa**).

Rukovanje nitima

- U slučaju kada rukovanje nitima **nije** u nadležnosti **operativnog sistema**, brigu o nitima potpuno preuzima **konkurentna biblioteka**.
- Pošto ona pripada slici procesa, rukovanje nitima u ovom slučaju se potpuno odvija u **korisničkom prostoru**, u kome se nalaze i **deskriptori niti**, kao i **stekovi** niti.
- Osnovna prednost rukovanja nitima **van operativnog sistema** je **efikasnost**, jer su pozivi potprograma konkurentne biblioteke **brži** (**kraći**) od poziva **sistemskih operacija**.

Rukovanje nitima

- Ali, kada operativni sistem ne rukuje nitima, tada poziv blokirajuće sistemske operacije iz jedne niti dovodi do **zaustavljanja aktivnosti procesa kome** ta **nit pripada**, jer operativni sistem pripisuje sve pozive sistemskih operacija samo **procesima**, pošto **ne registruje postojanje niti**.
- Na taj način se **sprečava konkurentnost unutar procesa**, jer zaustavljanje aktivnosti procesa **sprečava aktivnost njegovih spremnih niti**.
- Zato savremeni operativni sistemi **podržavaju rukovanje nitima**. Tako, u okviru **POSIX** (**Portable Operating System Interface**) standarda (IEEE 1003 ili ISO/IEC 9945) postoji deo **pthread** (**POSIX threads**) koji je posvećen nitima.

Osnova sloja za rukovanje procesima

- Sloj za **rukovanje procesima** se oslanja na slojeve:

–za **rukovanje datotekama** (radi pristupa sadržaju **izvršne datoteke** i radi rukovanja kopijama **slika procesa**)

–za **rukovanje radnom memorijom** (radi **zauzimanja** i **oslobađanja** zona radne memorije, potrebnih za smeštanje slika procesa)

–za **rukovanje procesorom** (jer do preključivanja dolazi prilikom **stvaranja** i **uništenja** procesa)

Osnove komunikacije između procesa

- Komunikacija između procesa može biti, kroz POSIX:
  - Signal
  - Baferi (Pipe, FIFO, Message Queue)
  - Deljena memorija
  - Semafori

Signali

- Signali su softverski prekidi koji označavaju da se u sistemu desio nekakav događaj.
- Ponekad ih generišu drugi procesi, ponekad sam operativni sistem, a ponekad korisnik direktno.
- Signali imaju različito značenje
- Može se dobiti kompletna lista iz operativnog sistema

Signali

```
veljko@HPC:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

Reakcija na signale

Svaki tip signala ima jednu podrazumevanu akciju iz sledećeg skupa:

- **Term**
  - Proces koji dobija signal se terminira.
- **Ign**
  - Proces koji dobija signal ga ignoriše.
- **Core**
  - Podrazumevana akcija jeste da se terminira proces i da se u fajl izbaci sva memorija procesa.
- **Stop**
  - Proces se pauzira.
- **Cont**
  - Proces se nastavi ako je pauziran.

Namena signala

Ime signala	Broj signala	Reakcija	Svrha
SIGTERM	15	Term	Signal za terminaciju
SIGUSR1	10	Term	Rezervisan za korisnika.
SIGUSR2	12	Term	Rezervisan za korisnika.
SIGCHLD	17	Ign	Dete-proces je prekinut.
SIGCONT	18	Cont	Nastavi izvršavanje.
SIGSTOP	19	Stop	Pauziraj izvršavanje.
SIGSTP	20	Stop	Pauziraj izvršavanje (pokrenut sa terminala)
SIGTTIN	21	Stop	Komunikacija sa terminalom za pozadinski proces
SIGTTOU	22	Stop	Komunikacija sa terminalom za pozadinski proces

Namena signala

Ime signala	Broj signala	Reakcija	Svrha
SIGBUS	7	Core	Greška magistrale
SIGPOLL	29	Term	Sinonim za SIGIO
SIGPROF	27	Term	Tajmer za profiliranje je istekao
SIGSYS	31	Core	Greška u sistemskom pozivu.
SIGTRAP	5	Core	Breakpoint dostignut.
SIGURG	23	Ign	Hitna reakcija na socket-u.
ISGVTALRM	26	Term	Virtualni alarm
SIGXCPU	24	Core	Potrošeno svo CPU vreme.
SIGXFSZ	25	Core	Potrošeno ograničenje veličine fajla.

Namena signala

Ime signala	Broj signala	Reakcija	Svrha
SIGIOT	6	Core	Isto što i SIGABRT.
SIGSTKFLT	16	Term	Stek greška na koprocesoru. (Nekorišćeno)
SIGIO	29	Term	I/O sada moguć.
SIGPWR	30	Term	Greška sa napajanjem.

Slanje signala

```
kill -<signal> <pid>
```

```
kill -SIGUSR1 10366
```

Slanje signala

```c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

int main() {
    for(int i = 1; i < 99999999; i++){
        printf("%d\n", i);
        if(i == 4817){
            kill(getpid(), SIGTERM);
        }
    }
    return 0;
}
```

Reagovanje na signal

```c
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

void signalCallback(int sig){
    printf("Your precious TERM signal won't save you now! I got: %d\n", sig);
}

int main() {
    signal(SIGTERM, signalCallback);
    for(int i = 1; i < 99999999; i++){
        printf("%d\n", i);
        if(i == 4817){
            kill(getpid(), SIGTERM);
        }
        if(i == 5993){
            kill(getpid(), SIGKILL);
        }
    }
    return 0;
}
```

Baferi

- Bafer je opšte ime za nekoliko metoda za komunikaciju.
- Ovde ćemo pogledati malo bolje 'pipe' i 'fifo' mehanizam koji funkcioniše kao fajl koji, kad se napravi, ima dva deskriptora: iz jednog se čita, a u drugi piše.
- Više procesa može da deli ove, sa tim da ako jedan piše, drugi ne može. To se rešava tako što svaki proces zatvori ono što nije neophodno.
- Pipe živi koliko i jedan proces, FIFO je permanentan i ima ime.

Pipe—Include

```
1  #include <iostream>
2  #include <unistd.h>
3  #include <sys/types.h>
4  #include <cstring>
5
6  using namespace std;
7
```

```
 8  int main(){
 9      int pipeD[2];
10      int pipeR[2];
11      pid_t pid;
12      char inbuff[256];
13      char outbuff[256];
14      int status = pipe(pipeD);
15      if(status == -1){
16          cout << "Ne mogu da napravim bafer." << endl;
17          return 1;
18      }
19      status = pipe(pipeR);
20      if(status == -1){
21          cout << "Ne mogu da napravim bafer." << endl;
22          return 1;
23      }
24      pid = fork();
25      if(pid == 0){…
35      }else{…
45      }
46      return 0;
47  }
```

Pipe Main

Pipe—Dete proces

```
if(pid == 0){
    //dete
    close(pipeR[1]);
    close(pipeD[0]);
    read(pipeR[0], inbuff, 256);
    cout << "[DETE] Ja sam dobio " << inbuff << endl;
    strcpy(outbuff, "Test 123");
    cout << "[DETE] Ja sad saljem " << outbuff << endl;
    write(pipeD[1], outbuff, strlen(outbuff) + 1);
    cout << "[DETE] Poslao!" << endl;
```

Pipe—Roditelj proces

```
}else{
    //roditelj
    close(pipeR[0]);
    close(pipeD[1]);
    strcpy(outbuff, "Test 321");
    cout << "[RODITELJ] Ja sad saljem " << outbuff << endl;
    write(pipeR[1], outbuff, strlen(outbuff) + 1);
    cout << "[RODITELJ] Poslao!" << endl;
    read(pipeD[0], inbuff, 256);
    cout << "[RODITELJ] Ja sam dobio " << inbuff << endl;
}
```

FIFO

FIFO je apsolutno identičan Pipe mehanizmu sa tom razlikom da ima ime i postojanje koje je nezavisno u odnosu na proces koji ga stvara.

FIFO uvek postoji negde na disku, tj. ima putanju.

Deljena memorija

Deljenu memoriju više proučavamo kroz kod, ali ideja je jednostavna: definišemo region memorije koji dele dva procesa, i komuniciramo kroz njega.

Jako je brzo, ali su problemi sinhronizacije poprilični.