Osnovni koncepti operacijskih sustava

Upravljanje dretvama: komunikacija

Principi međudretvene komunikacije

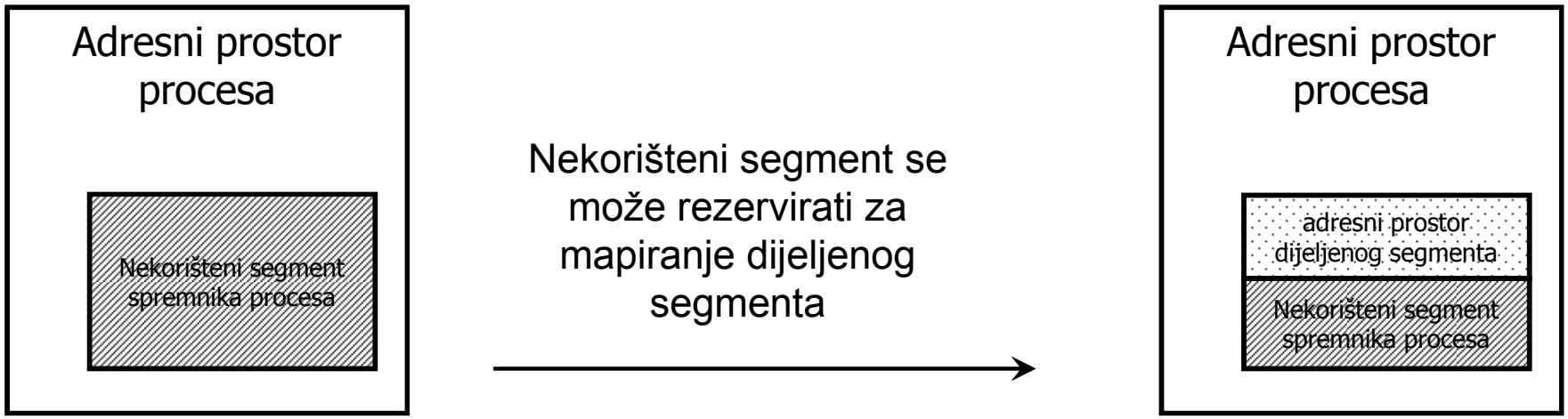
- Procesi ili dretve koji surađuju na zajedničkom zadatku:
  - provode operacije nad skupom podataka
  - dijele podatke, i/ili
  - izmjenjuju podatke/poruke/događaje/signale, i/ili
  - međusobno se sinkroniziraju

- Osnovni principi komunikacije uključuju:
  - zajednički (dijeljeni) spremnik
  - poruke – slanje i primanje poruka
  - cjevovode – slanje podataka u cjevovod, čitanje iz njega
  - signali – detektirani ili generirani događaji od strane jedne dretve ili jezgre traže posebno postupanje i od druge

  - datoteke – pasivna (“offline”) komunikacija

Zajednički spremnik

- Sve dretve istog procesa dijele adresni prostor procesa
  - to je zajednički spremnik za te dretve
- Dretve različitih procesa mogu stvoriti dijeljene segmente spremnika koristeći jezgrine pozive
  - dio adresnog prostora procesa može se iskoristiti za mapiranje adresnog prostora dijeljenog segmenta



Sučelje za dobavljanje zajedničkog spremnika

- Uobičajeno sučelje se može definirati sa:

`dobavi_zajednicki_segment(ime,vel,adresa,zast)`

  - **`ime`** – identifikacija novog ili postojećeg segmenta
    - može biti broj ili znakovna konstanta (ili ime datoteke)
    - mora biti jedinstveno u sustavu
  - **`vel`** – tražena veličina zajedničkog spremnika
  - **`adresa`** – u koji dio adresnog prostora procesa mapirati dohvaćeni segment
  - **`zast`** – prava korištenja, “stvori ako ne postoji” zastavica…
  - Funkcija vraća ili status ili početnu adresu ili opisnik

- UNIX: **`shmget, shmat`**
- POSIX: **`shm_open, ftruncate, mmap`**
- Win32: **`CreateFileMapping, MapViewOfFile`**

Zaštita zajedničkog spremnika

- Korištenje zajedničkog spremnika treba zaštititi od istovremenog korištenja (izmjene), ili može doći do grešaka
  - koristiti neki od sinkronizacijskih mehanizama

- Zajednički spremnik može biti najbrži način međudretvene komunikacije (ako se minimizira sinkronizacija)

- Zajednički spremnik može biti izvor grešaka koje je vrlo teško otkriti! Uglavnom zbog nekontroliranog korištenja

Poruke

- Poruka je kratki blok informacija koji se šalje među dretvama (jedna šalje drugoj)

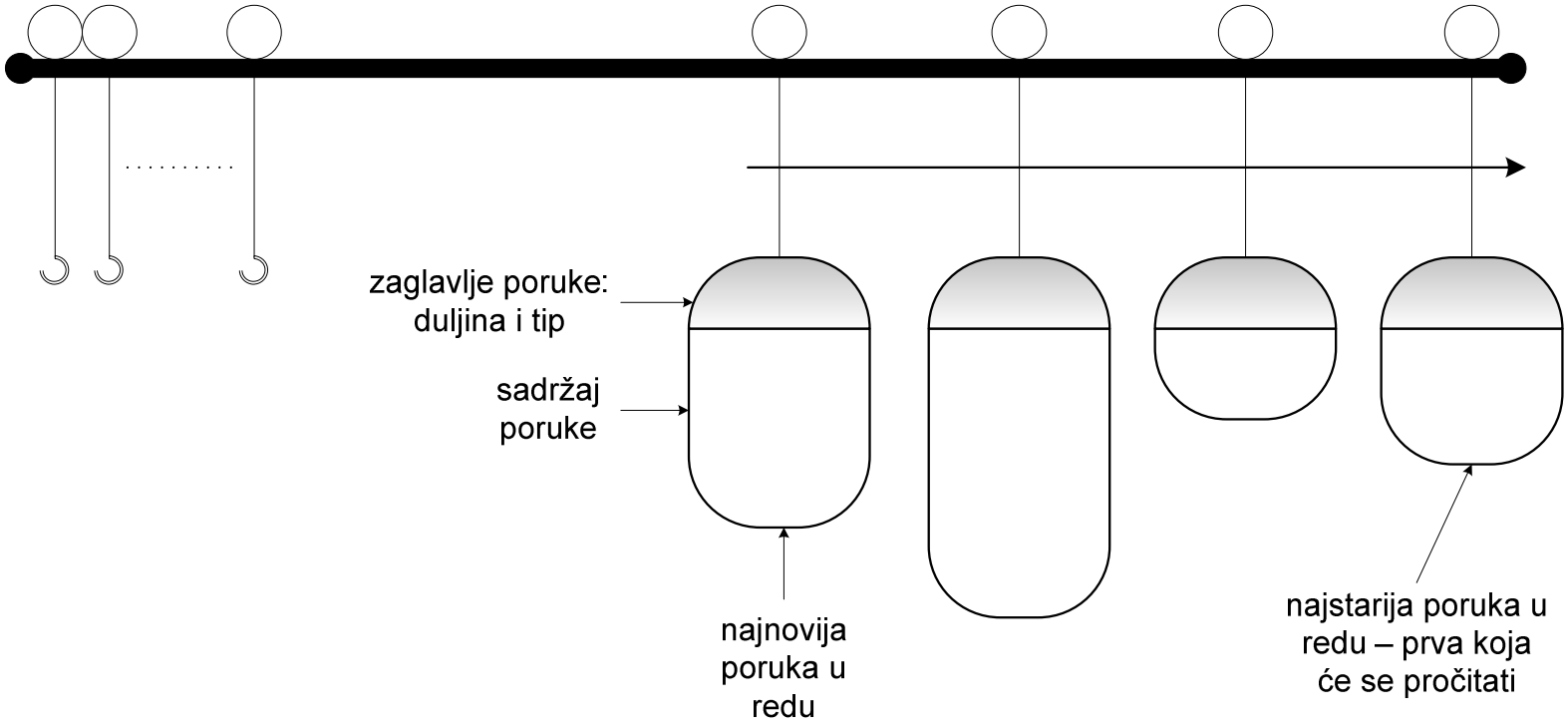
- Poruke se ne šalju izravno (između dretvi) već se koristi operacijski sustav i njegovi mehanizmi
- Kada se *šalje poruka*, poruka se stavlja *u red poruka*
- Kada se *dohvaća poruka*, ona se uzima *iz reda poruka*

- *Redovi poruka* se koriste kroz sučelje operacijskog sustava: stvaranje reda, slanje, primanje, brisanje reda

- U nekim sustavima (npr. RT) svakoj je dretvi već inicijalno (pri njenom stvaranju) pridijeljen vlastiti red
  - poruke su osnovno komunikacijsko sredstvo

Redovi poruka

- poruke mogu biti različitih veličina i tipova
- redovi poruka su FIFO strukture
  - ponekad (u nekim sučeljima) je moguće čitati poruke samo određenog tipa (čak i ako nisu prve u redu poruka)



Sučelja za korištenje reda poruka

- Stvaranje reda poruka:
  `dohvati_red_poruka (ime, zast)`
  - **`ime`** – identifikacija novog ili postojećeg reda poruka
    - može biti broj ili znakovna konstanta (ili ime datoteke)
    - mora biti jedinstveno u sustavu
  - **`zast`** – prava korištenja, "stvori ako ne postoji" zastavica…
  - funkcija vraća opisnik (ID) stvorenog/dohvaćenog reda

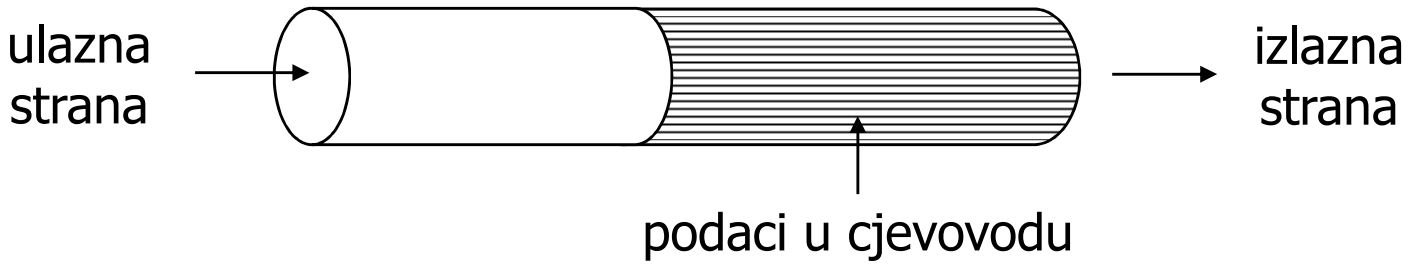
- Slanje i primanje poruka:
  `pošalji_poruku(id_reda, kpodaci, vel, zast)`
  `dohvati_poruku(id_reda, kpodaci, vel, zast)`
  - **`kpodaci`** – kazaljka na poruku koja se šalje ili lokacija kamo poruku treba staviti (pri dohvatu)
  - **`vel`** – duljina/najveća duljina poruke
  - **`zast`** – razne zastavice, npr. treba li dretvu blokirati ako nema poruke u redu ili je red pun a želi se slati poruka

Sučelja za poruke u raznim sustavima

- UNIX
  - `msgget, msgsnd, msgrcv`
- POSIX
  - `mq_open, mq_send, mq_receive`
- Win32
  - `MQCreateQueue, MQSendMessage, MQReceiveMessage`

Cjevovodi

- Cjevovod je FIFO struktura (slično redu poruka, ali …)
  - FIFO je jedini način korištenja cjevovoda (nema preskakanja podataka)
- Cjevovod ima dvije strane: *ulaznu* i *izlaznu*



- Podaci se u cjevovod šalju na ulaznu stranu, a čitaju iz izlazne
  - cjevovod ima dva opisnika, za svaku stranu po jedan
- Nema granulacije podataka i njihovo razlikovanje (kao kod reda poruka)
- Čitanje iz cjevovoda je destruktivno (podaci se miču)

Sučelje za rad s cjevovodima

- Korištenje cjevovoda je slično korištenju datoteka:
  - otvori, čitaj/piši, zatvori

- Stvaranje/otvaranje cjevovoda:

  `dohvati_cjev (ime, opisnik(ici), zast)`
  - **`ime`** – identifikacija novog ili postojećeg reda poruka
    - može biti broj ili znakovna konstanta (ili ime datoteke)
    - mora biti jedinstveno u sustavu
    - **ne mora biti podržano u svim implementacijama**
  - **`opisnik(ici)`** – opisnik tražene strane cjevovoda (ili oba opisnika)
  - **`zast`** – "koja strana: ulaz ili izlaz ili obje", "blokiraj dok se druga strana također ne otvori?", prava pristupa, …

Sučelje za rad s cjevovodima

- Čitanje i pisanje

```
piši_u_cijev (op_ulaza, podaci, vel, zast)
citaj_iz_cijevi (op_izlaza, podaci,vel,zast)
```

  - **`op_ulaza/op_izlaza`** – opisnici ulaza/izlaza cjevovoda
  - **`podaci`** – adresa podataka koji se šalju u cijev ili adresa kamo će se pročitani podaci staviti
  - **`vel`** – veličina za podatke (“**`podaci`**”)
  - **`zast`** – “blokiraj ako je cijev prazna/puna”, …

  - veličina upisanih/pročitanih podataka je obično povratna vrijednost funkcije

- Implicitno korištenje cjevovoda iz ljuske
  - npr: **`cat`** ***`dat1`*** **`|`** **`grep`** ***`ime`*** **`|`** **`sort`** **`>`** ***`dat2`***

Sučelja za cjevovode u raznim sustavima

- UNIX
  - **`pipe`** – neimenovani cjevovodi
    - procesi koji koriste ove cjevovode moraju biti u “srodstvu” (dretve unutar istog procesa ili procesi roditelj-dijete, djeca istog roditelja)
  - **`mknod, open, close`** – imenovani cjevovodi
    - ime cjevovoda postoji u datotečnom sustavu, procesi koji preko njih komuniciraju ne moraju biti u srodstvu
  - **`read, write`** – čitanje i pisanje
- Win32
  - **`CreatePipe`** – neimenovani cjevovodi
  - **`CreateNamedPipe`** – imenovani cjevovodi
  - **`CreateFile, CloseHandle`**
  - **`ReadFile, WriteFile`**

Signali

- Signali objavljuju asinkrone (“neočekivane”) događaje
- Signali su slični prekidima:
  - *prekidi* se koriste na sklopovskoj razini, gdje uređaji signaliziraju procesoru svoj zahtjev za posluživanje
  - *signali* se koriste na razini *operacijskog sustava*
    - OS ili dretve ih generiraju – odredišne dretve ih obrađuju
- OS šalje signal procesu kada traži posebne obrade od njegovih dretvi – kao reakciju na određeni događaj
  - npr. kada se tipka pritisne na tipkovnici generira se prekid; u obradi tog prekida generira se signal koji se šalje dretvi
- Dretve mogu slati signale jedna drugoj (preko sučelja OSa), npr. tražiti prekid rada druge dretve

Signali

- Dretva može reagirati na signal tako da:
  - ignorira signal
  - obradi signal s prethodno definiranom funkcijom
  - obradi signal s pretpostavljenom funkcijom (najčešće ta funkcija prekida rad dretve)
  - zadržati signal (odgoditi njegovo prihvaćanje do nekog trenutka u budućnosti kada se ponašanje za taj signal bude promijenilo)

- Signali ne nose dodatne informacije – samo oznaku
  - U nekim proširenim sučeljima (npr. RT), signali mogu nositi i dodatnu vrijednost

Sučelje za rad sa signalima

- Definicija “signalne maske” – ponašanje za signale

```
podesi_prihvat_signala (id, obrada, param)
```

  - **`id`** – identifikacija signala
  - **`obrada`** – funkcija koja će se pozvati pri primitku signala, ili oznaka koja definira:
    - ignoriraj signal
    - obradi signal pretpostavljenom funkcijom
    - zadrži signal
  - **`param`** – opcionalni parametar pri pozivu funkcije

Sučelje za rad sa signalima

- Slanje signala dretvi

```
pošalji_signal (id_dretve, id_signala, param)
```

  - **`id_dretve`** – identifikacija dretve kojoj se signal šalje
  - **`id_signala`** – id signala koji se šalje
  - **`param`** – opcionalni parametar uz signal

Sučelja za signale u raznim sustavima

- Na nekim (UNIX) sustavima signali su mehanizam na razini procesa – maske se definiraju na razini procesa

- Upravljanje signalima na razini dretve je noviji princip
  - različito se implementira na različitim sustavima
    - treba pažljivo čitati upute (za funkcije)

- UNIX/POSIX
  - **`signal, sigset, sigaction`** – definicija ponašanja za pojedini signal

  - **`kill, sigqueue, raise, pthread_kill`** – slanje signala dretvi/procesu

Datoteke kao komunikacijski mehanizam

- Komunikacija “preko” datoteka:
  - *dretva pošiljatelja* stvara datoteku, puni ju podacima
  - *dretva primatelja* (naknadno, kad je pošiljatelj već gotov) otvara datoteku i čita njen sadržaj
- Komunikacija je “statična”
  - *pošiljatelj* mora proizvesti sve podatke i tek ih onda u kompletu “poslati” primatelju (korištenjem datotečnog sustava)
  - dodatna sinkronizacija može biti neophodna

- Prednosti komunikacije preko datoteka
  - podaci u datoteci mogu ostati znatno duže nego korištenjem ostalih komunikacijskih mehanizama
    - podaci će “preživjeti” i gašenja računala
  - veličina podataka za razmjenu može znatno nadmašiti mogućnosti ostalih komunikacijskih mehanizama