OPERACIJSKI SUSTAVI 2

Vježba 2

U ovoj vježbi bavimo se, kako ja volim reći, pipama (pipes, cjevovodi po hrvatski). Riječ je o mehanizmu kojim operacijski sustav omogućava primanje i slanje veće količine podataka pri čemu sudionici tog transfera cjevovod vide kao običnu datoteku. Tako se u cjevovod piše i čita istim funkcijama kojima se koristimo pri radu sa datotekama. Jedina rezlika je funkcija koja stvara cjevovod. U tu svrhu ne koristi se funkcija `open`, već funkcija `pipe`.

Par riječi o datotekama...

U C-u postoje dvije klase funkcija koje se bave datotečnom problematikom. Prvu klasu čine sve one funkcije koje datoteku pamte preko tzv. *file-pointera*, odnosno varijable tipa (`FILE*`). To su funkcije poput fopen, fclose, fread, fwrite, fscanf, fprintf, fgets, fputs i slične. U drugu klasu spadaju funkcije koje datoteku pamte preko njezinog rukovatelja (handle). Svakoj se datoteci prilikom otvaranja pridružuje njezin nenegativni i u tom procesu jedinstveni broj koji jednoznačno definira tu datoteku. Prvih nekoliko rukovatelja zauzima se već prilikom pokretanja našeg procesa. To su redom:

Rukovatelj broj:	Opis sredstva:	FILE * ekvivalent:
0	Rukovatelj standardnog ulaza; obično veza prema tipkovnici	stdin
1	Rukovatelj standardnog izlaza; obično veza prema zaslonu/terminalu	stdout
2	Rukovatelj standardne greške; obično veza prema zaslonu/terminalu	stderr

Ukoliko želimo raditi sa rukovateljima datoteka, tada ćemo loristiti funkcije poput open, close, read, write, dup i sl. Ovaj tip funkcija koristan je zbog toga što i funkcija za stvaranje cjevovoda radi sa rukovateljima a ne sa file-pointerima.

Primjer uporabe cjevovoda...

Uzmimo za primjer da smo napisali program koji podatke za rad čita isključivo sa standardnog ulaza, i rezultat obrade šalje isključivo na standardni izlaz. Nije baš neka zvijer od programa, ali ima i takvih. Postavlja se pitanje kako takvom programu ipak poslati nekakve podatke koje generira naš program i koje ne želimo svaki puta ručno utipkavati da bi on to mogao pročitati sa standardnog ulaza. To je vrsta problema gdje se služimo cjevovodima. Potrebno je stvoriti cjevovod koji ćemo povezati između našeg programa i standardnog ulaza tog programa. Tada će naš program podatke zapisivati u cjevovod, a drugi će program podatke čitati sa standardnog ulaza koji više neće biti "standardni ulaz" već drugi kraj cjevovoda. Pitanje je samo kako ovo realizirati...

O cjevovodu...

Svaki cjevovod ima dva kraja: kraj za čitanje, i kraj za pisanje. Cjevovod si jednostavno možemo predočiti kao tunel. Na jednu stranu guramo nešto u tunel; na drugoj strani nešto izlazi iz tunela. Onaj kraj u koji nešto guramo naziva se kraj za pisanje. Kraj iz kojega nešto izlazi naziva se kraj za čitanje. Operacijski sustav cjevovod pamti pomoću dva rukovatelja. Jedan rukovatelj je rukovatelj kraja za čitanje i može se koristiti isključivo u funkcijama za čitanje (read i kompanija); pisanje u taj kraj nije dopušteno. Drugi rukovatelj je rukovatelj kraja za pisanje i on se može koristiti isključivo za pisanje (write & co). Funkcija kojom se stvara cjevovod je pipe i ona zahtjeva jedan argument: polje od dva elementa - cijela broja, u koje će pohraniti rukovatelje krajeva cjevovoda.

int pipe(int handles[2]);

Po uspješnom povratku iz funkcije element handles[0] sadrži rukovatelj kraja za čitanje, dok element handles[1] sadrži rukovatelj kraja za pisanje.

O funkciji dup...

Funkcija dup koristi se za dupliciranje rukovatelja datoteke. Broj koji će funkcija dup vratiti biti će različit od svih brojeva koji se trenutno koriste za rukovatelje. Ujedno će biti i prvi nenegativni broj koji je raspoloživ. Treba uočiti da ako se rukovatelj neke datoteke duplicira, zatvaranjem jednog rukovatelja (funkcija close) datoteka se fizični ne zatvara! Datoteka će biti zatvorena tek kada se svi rukovatelji te datoteke zatvore. Razlog zbog čega ovo govorim leži u činjenici da nam ovo sve treba za cjevovode.

Povezivanje cjevovoda na standardni ulaz...

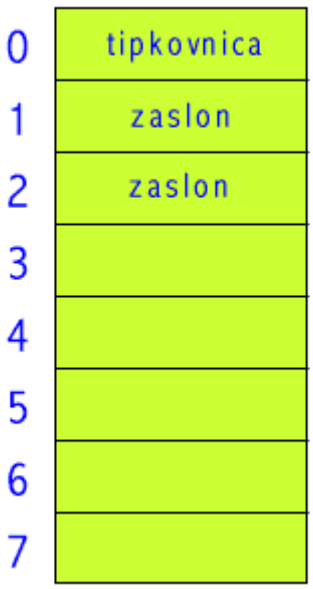
Povezivanje se vrši u nekoliko koraka. Pretpostavlja se da je cjevovod već kreiran pozivom funkcije pipe(cj); pri čemu je cj polje od dva integera. Dakle, kod poput ovog u nastavku već je trebalo izvršiti na početku programa:

```
int cj[2];

pipe( cj );
```

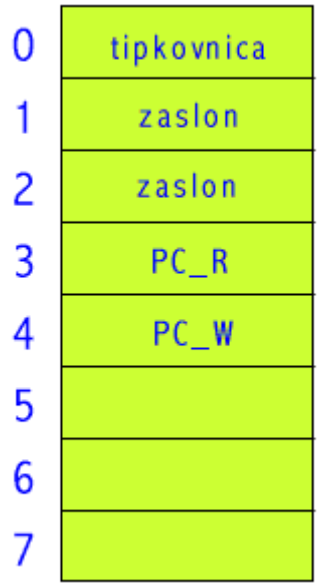
Tada se povezivanje cjevovoda na standardni ulaz vrši na slijedeći način:

1. `close(0);`
2. `dup( cj[0] );`
3. `close( cj[0] );`
4. `close( cj[1] );`

Kako i zašto ovo radi? Rukovatelj datotekom možemo si predočiti na slijedeći način. Pretpostavimo da svaki proces sadrži internu tablicu u kojoj sprema opis pojedine datoteke. Tada možemo rukovatelj datotekom zamisliti kao indeks onog elementa tablice u kojoj je spremljen opis datoteke. U tom slučaju, pokretanjem programa stvorila bi se tablica sa slike 1.

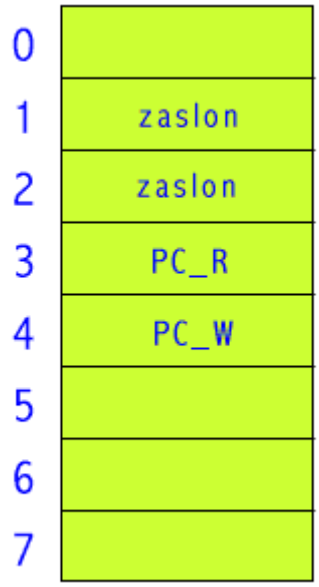


Dakle, rukovatelj 0 odgovara standardnom ulazu što je obično tipkovnica, itd. Nakon što smo izveli poziv pipe(cj); stanje u tablici je slijedeće:

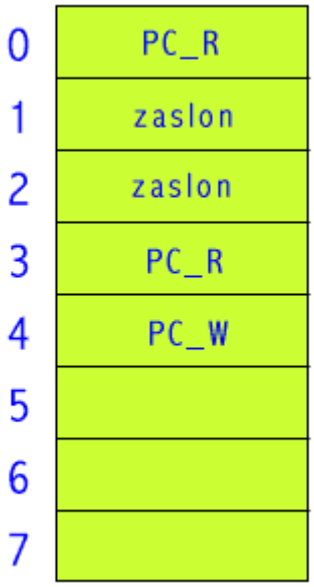


pri čemu PC_R označava kraj cjevovoda za čitanje (Read), dok PC_W označava kraj cjevovoda za pisanje (Write).

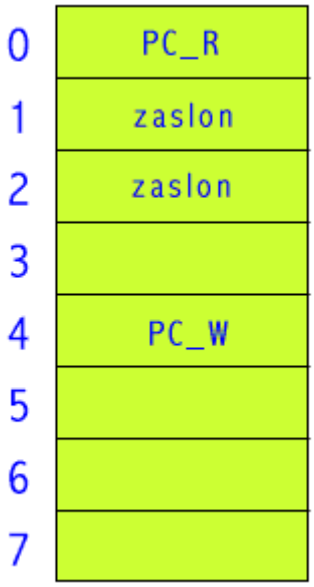
Izvedimo sada algoritam korak po korak. Korak 1 zatvara rukovatelj 0.



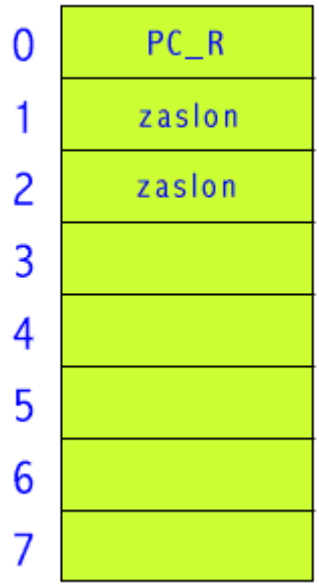
Korak 2 duplicira kraj cjevovoda za čitanje, a kako je prvi neiskorišteni rukovatelj sada upravo nulti, duplikat rukovatelja kraja cjevovoda za čitanje smješta se na nultu lokaciju:



Korak 3 zatvara original kraja za čitanje:



Korak 4 zatvara original kraja za pisanje:



iz ove tablice se vidi da je sada na mjestu standardnog ulaza (rukovatelj 0) zapravo kraj za čitanje cjevovoda! Korak 4 algoritma zatvorio nam je i kraj za pisanje jer je pretpostavka da je proces podatke o cjevovodu već nekome proslijedio i taj netko će koristiti taj kraj; kako on ne treba našem procesu, on ga zatvara.

Povezivanje cjevovoda na standardni izlaz...

Povezivanje se vrši u nekoliko koraka. Pretpostavlja se da je cjevovod već kreiran pozivom funkcije pipe(cj); pri čemu je cj polje od dva integera. Dakle, kod poput ovog u nastavku već je trebalo izvršiti na početku programa:

```
int cj[2];

pipe( cj );
```

Tada se povezivanje cjevovoda na standardni ulaz vrši na slijedeći način:

1. `close(1);`
2. `dup( cj[1] );`
3. `close( cj[1] );`
4. `close( cj[0] );`

Ukoliko ste shvatili kako radi prethodni algoritam, tada i ovaj mora biti sajan pa ga neću ponovno opisivati. Dovoljno je reći da je konačan rezultat algoritma rukovatelj kraja pisanja u cjevovod pod brojem 1, odnosno na standardnom izlazu.

Još malo teorije...

Cjevovodi kreirani funkcijom pipe ne mogu se dijeliti između proizvoljnih procesa. Cjevovode mogu naslijediti jedino djeca procesa. Nasljeđivanje se vrši automatski. Naime, onog trenutka kada proces P stvori dijete D pozivom funkcije fork, tada dijete D naslijedi sve rukovatelje koje je u tom trenutku imao i proces P (zapravo termin "naslijedi sve rukovatelje" trebalo bi zamijeniti sa "naslijedi duplikate svihrukovatelja"). Naime, ukoliko proces P najprije stvori cjevovod pozivom funkcije pipe, tada on u svojoj tablici ima i dva rukovatelja cjevovoda. Neka proces P zatim stvori dijete D. Dijete D u svojoj tablici ima također rukovatelje cjevovoda! I još bolje, neka sada dijete D zatvori jedan kraj cjevovoda. Taj isti kraj cjevovoda kod roditelja i dalje ostaje

otvoren! Treba zapamtiti da dijete ima duplikate svih opisnika a ne originale! Zato zatvaranjem svojeg duplikata original ostaje i dalje aktivan sve dok i roditelj ne zatvori isti kraj!

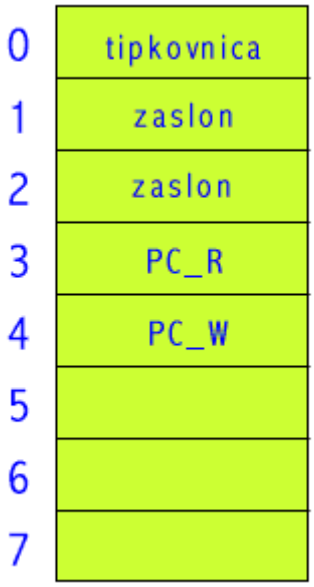
Jednostavan primjer...

Treba stvoriti cjevovod cj, i njime povezati proces P, i dijete D pri čemu cjevovod treba povezati na standardni ulaz djeteta. Evo kako:

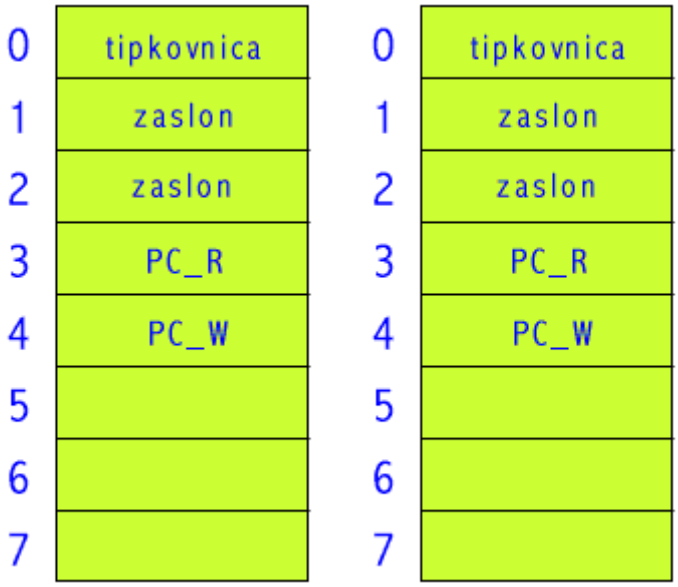
```
int main(int argc, char *argv[]) {
  int cj[2];

  if( pipe( cj ) == -1 ) {
   puts("Greska kod kreiranja prvog cjevovoda!");
   return 1;
  }

  switch( fork() ) {
   case -1:
     puts("Greska kod forkanja procesa!");
     return 1;
   case 0:
     // Povezi cjevovod cj na ulaz djeteta
     close(0); dup(cj[0]); close(cj[0]); close(cj[1]);

     ... radi nesto ...

     exit(0);
   default:
     break;
  }

  // U procesu roditelju zatvori ulaz iz kojeg djete cita
  close( cj[0] );

     ... radi nesto ...

  return 0;
}
```

Što se točno događa...

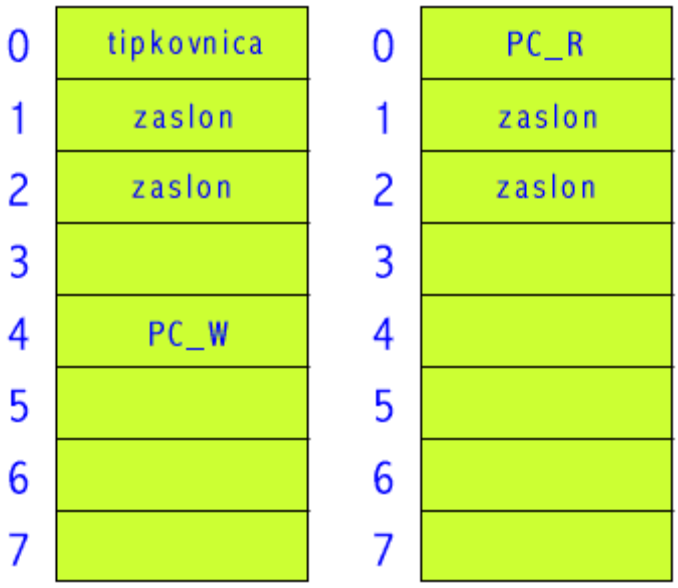
Do prije forka imamo situaciju:



Nakon forka imamo dvije tablice: lijeva je od glavnog procesa, desna je od djeteta.



Roditelj zatvara rukovatelj za citanje iz cjevovoda, dok dijete povezuje kraj za citanje iz cjevovoda na svoj standardni ulaz. Rezultat je:



Nakon svih ovih objašnjenja, mislim da labos stvarno ne bi smio biti težak. No ipak, kao i uvijek, source je ovdje: lv211x.cpp, test datoteka za drugi dio zadatka.

Ukoliko program pokrenete bez argumenata, odrađivati će se prvi dio vježbe. Ako zadate programu ime datoteke kao argument, tada ćete dobiti drugi dio vježbe.