Sisteme de operare - lab10

Comunicatii intre procese Unix: pipe

Aspecte teoretice

Una dintre modalitățile de comunicare între procese în Unix este cea prin intermediul canalelor de comunicație (numite pipes, în limba engleză). Practic este vorba despre o "conductă" (un buffer) prin care pe la un capăt se scriu mesajele, iar pe la celălalt capăt se citesc - deci este vorba despre o structură de tip coadă, adică o listă FIFO (First-In,First-Out).

Aceste canale sunt de două categorii:

- pipe-uri interne: aceste "conducte" sunt create în memoria internă a sistemului Unix;
- pipe-uri externe: aceste "conducte" sunt fișiere de un tip special, numit fifo, deci sunt păstrate în sistemul de fișiere (aceste fișiere fifo se mai numesc și pipe-uri cu nume).

În continuare ne vom referi la canalele interne.

Canale interne. Primitiva pipe
Deci un canal intern este un canal aflat în memorie, prin care pot comunica două sau mai multe procese.

Crearea unui canal intern se face cu ajutorul funcției pipe:

```
int pipe(int *p)
```
unde: p = parametrul efectiv de apel trebuie să fie un tablou int[2] ce va fi actualizat de funcție astfel:

- p[0] va fi descriptorul de fișier deschis pentru capatul Read al canalului;
- p[1] va fi descriptorul de fișier deschis pentru capatul Write al canalului;

iar valoarea int returnată este 0, în caz de succes (dacă s-a putut crea pipe-ul), sau -1, în caz de eroare.

Efect: în urma execuției funcției pipe se creează un canal intern și este deschis la ambele capete - în scriere la capătul referit prin p[1], respectiv în citire la capătul referit prin p[0].

După crearea unui canal intern, scrierea în acest canal și citirea din el se efectueaza la fel ca pentru fișierele obișnuite. Și anume, citirea din canal se va face prin intermediul descriptorului p[0] folosind funcțiile de citire uzuale (read, respectiv fread sau fscanf dacă se folosește un descriptor de tip FILE*), iar scrierea în canal se va face prin intermediul descriptorului p[1] folosind funcțiile de scriere uzuale (write, respectiv fwrite sau fprintf dacă se folosește un descriptor de tip FILE*).

Observație
Pentru ca două sau mai multe procese să poată folosi un pipe pentru a comunica, ele trebuie să aibă la dispoziție cei doi descriptori p[0] și p[1] obținuți prin crearea pipe-ului, deci procesul care a creat pipe-ul va trebui să le "transmită" cumva celuilalt proces.

De exemplu, în cazul când se dorește să se utilizeze un canal intern pentru comunicarea între două procese de tipul părinte-fiu, atunci este suficient să se apeleze primitiva pipe de creare a canalului înaintea apelului primitivei fork de creare a procesului fiu. În acest fel în procesul fiu avem la dispoziție cei doi descriptori necesari pentru comunicare prin intermediul acelui canal intern.
La fel se procedează și în cazul apelului primitivelor exec (deoarece descriptorii de fișiere deschise se moștenesc prin exec).
De asemenea, trebuie reținut faptul că dacă un proces își închide vreunul din capetele unui canal intern, atunci nu mai are nici o posibilitate de a redeschide ulterior acel capăt al canalului.

Citirea dintr-un canal intern (cu primitiva read)

- Apelul read va citi din canal și va returna imediat, fără să se blocheze, numai dacă mai este suficientă informație în canal, iar în acest caz valoarea returnată reprezintă numărul de octeți citiți din canal.
- Altfel, dacă canalul este gol, sau nu conține suficientă informație, apelul de citire read va rămâne blocat până când va avea suficientă informație în canal pentru a putea citi cantitatea de informație specificată, ceea ce se va întimpla în momentul când alt proces va scrie în canal.
- Alt caz de excepție la citire, pe lângă cazul golirii canalului: dacă un proces încearcă să citească din canal și nici un proces nu mai este capabil să scrie în canal vreodată (deoarece toate procesele și-au închis deja capătul de scriere), atunci apelul read returnează imediat valoarea 0 corespunzatoare faptului că a citit EOF din canal.

În concluzie, pentru a se putea citi EOF din pipe, trebuie ca mai întâi toate procesele să închidă canalul în scriere (adică să închidă descriptorul p[1]).

Scrierea într-un canal intern (cu primitiva write)

- Apelul write va scrie în canal și va returna imediat, fără să se blocheze, numai dacă mai este suficient spațiu liber în canal, iar în acest caz valoarea returnată reprezintă numărul de octeți efectiv scriși în canal (care poate să nu coincidă întotdeauna cu numărul de octeți ce se doreau a se scrie, căci pot apare erori I/O
- Altfel, dacă canalul este plin, sau nu conține suficient spațiu liber, apelul de scriere write va rămâne blocat până când va avea suficient spațiu liber în canal pentru a putea scrie informația specificată ca argument, ceea ce se va întâmpla în momentul când alt proces va citi din canal.
- Alt caz de excepție la scriere, pe lângă cazul umplerii canalului: dacă un proces încearcă să scrie în canal și nici un proces nu mai este capabil să citească din canal vreodată (deoarece toate procesele și-au închis deja capatul de citire), atunci sistemul va trimite acelui proces semnalul SIGPIPE, ce cauzează întreruperea sa și afișarea pe ecran a mesajului "Broken pipe".

Note

1. Cele afirmate mai sus, despre blocarea apelurilor de citire sau de scriere în cazul canalului gol, respectiv plin, corespund comportamentului implicit, *blocant*, al canalelor interne.
2. Funcțiile de nivel înalt (fread/fwrite, fscanf/fprintf, etc.) lucrează buffer-izat. (vezi fflush)
3. Conversia descriptorilor de fișiere de la tipul int la tipul FILE* se realizează cu ajutorul primitivei fdopen.

popen și pclose

Unix oferă funcțiile de bibliotecă popen și pclose. Biblioteca standard <*stdio.h*> descrie aceste apeluri:

```
FILE* popen (const char *comanda, const char *tip);
int pclose (FILE *pointer);
```

comanda este un string ce conține o comandă shell, iar *tip* este un string ce poate avea fie valoarea "r", fie valoarea "w".

Efectul apelului sistem *popen* este următorul: deschide un pipe, apoi execută un fork. Procesul fiu execută prin exec comanda. Procesele tată și fiu comunică prin pipe astfel:

- dacă tipul este "r", atunci procesul tată citește din pipe, folosind pointerul la fișier întors de popen, ieșirea standard dată de comanda
- dacă tipul este "w", atunci procesul tată scrie în pipe, folosind pointerul la fișier întors de popen, iar ceea ce scrie se constituie în intrarea standard pentru comandă.

Dacă comanda a fost lansată cu intrarea redirectată dinspre procesul apelant, atunci în continuare putem trimite date comenzii lansate prin fwrite() sau fprintf(). Dacă comanda a fost lansată cu ieșirea redirectată spre procesul apelant, atunci ieșirea comenzii se poate citi din procesul apelant cu functiile fread() sau fscanf(). În ambele cazuri, încheierea se face prin apelul funcției *pclose()*.
Apelul *pclose* închide pipe și întoarce codul de retur al comenzii. La eșec întoarce -1.