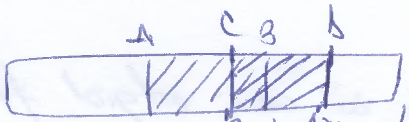
Blocaje



- să se pună un blocaj între A și B
- nimeni nu are voie să citească sau să scrie

Tipuri:

- partajat - un alt proces vrea să pună un blocaj în zonele C și D
  - accepta să coexiste cu alte blocaje
- exclusiv - nu accept un alt blocaj incompatibil cu al meu (care "mușcă" din zona mea)

• modul de acțiune:

- consultative - nu împiedică citirea și scrierea în zona
- imperativ - împiedică ... blocată până când se ridică blocajul

Numai procesul care a pus blocajul poate să-l ridice

Putem avea deadlock.

FCNTL → -1 → (EDEADLCK)

```
struct flock                b. part.     b. excl.    deblocare
{ short l_type;     /* F_RDLCK , F_WRLCK , F_UNLCK */
  short l_whence;   /* SEEK_SET, SEEK_CUR, SEEK_END */
  // descriem zona blocată       poz curentă   sf f.s
  int l_start;
  int l_len; // pozitiv
     lung zonei blocate
  int pid; // id-ul proc care a impus blocajul
}
```

int fcntl (int dest , int oper , struct flock *p)

F_SETLCK – blocaj de tip consultativ

F_SETLKW – blocaj facultativ

F_GETLCK

Ca blocajele să meargă tb să am setgid fără drept de execuție:

rwxrwSrwx

rw-r-Sr-- // nu are ce căuta x în drepturi

Exploatarea directoarelor

direct.h – conține toate def

DIR* opendir(char* nume) – deschidere
– eșec = NULL

struct direct *readdir(DIR *p) – intrare ; când ajung la sf returnează NULL

dă în ord în care au fost date fiș / ex în ord aff. returnat de opendir

Obs: char d_name [D_MAX] → valabil pe toate sistemele (singurul câmp)
→ ret și . și ..

int closedir (DIR *p) – închiderea directorului

void rewinddir (DIR *p) – resetez poz curentă la început

Examen: directoare / fără parcurgerea recursivă a dir

Parcurgerea dir:

1) chdir (dirname);
opendir (".");
while (readdir != NULL)
{
—— //parc. de nume
}
chdir ("..");

2) pun calea într-un string și concatenăm la opendir ----

Schimbarea poz curente într-un fișier:

int lseek(int desc, int orig, int lg)

↓ descriptor ↓ origine ↓ lungime

||

SEEK_SET, SEEK_CUR, SEEK_END

lg > 0 → origine

lg > 0 & lg < 0 → poz cur

lg < 0 → final

Tuburi (pipe-uri)

Caracteristici:

1. Nu există descriptori R/W
   Sunt ori ReadOnly ori WriteOnly
2. Citirile sunt distructive
3. Citirile sau scrierile pot fi blocante

Categorii:

1) Fișiere de tip tub cu ~~nume~~ (named pipes)
   - au ~~tre~~ cel puțin o leg fizică pe disc (p)

2) ——— || ——— fără nume (unnamed pipes)
   - nu au nici o leg fiz pe disc, dar au inodul în memorie

1) $ mkfifo nume → creare - din shell
   int mkfifo(char *nume, int dr) → creare din program

   dr → drepturi de acces (dacă nu le dăm sunt cele din default)

- open - rămâne blocant dacă nu avem nici un desc.
- read
- write
- close

2) int pipe(int *p) - creare < 0 - succes ; -1 - insucces

int p[2]; - min 2 elem.

p[0] descriptor la citire

p[1] ——— || ——— la scriere

descriptorii se moștenesc → în urma fork-ului (de toate proc. descendente)

Lucrăm cu read, write, close. NU avem voie să folosim open.

c1 | c2 | c3

stdoutput c1 = stdinput c2
stdoutput c2 = stdinput c3
stdinput ~~c3~~ = input c1
~~std~~ output c3 = output

ls -l / wc -l
af. dir ↵ fiș → af. nr. de linii

La open de testat neapărat
codul de retur

```
int p[2];
pipe(p);
if (fork()>0) // procesul tată (folosește ls -l)
  {
    close(1); // închide stdout
    dup(p[1]); // duplicat desc de scriere în desc 1
    close(p[0]);
    close(p[1]);
    execlp("ls","ls","-l",NULL);
  }
else  // proc fiu
  {
    close(0); // închid stdinput
    dup(p[0]); // duplicarea se face în cel mai mic desc. liber
    close(p[0]);
    close(p[1]);
    execlp("wc","wc","-l",NULL);
  }
```

Când avem c1 | c2 | ... | cn ~~fiecare~~ se crează n-1 pipe-uri și se lansează în fiecare își închid stdinput/output

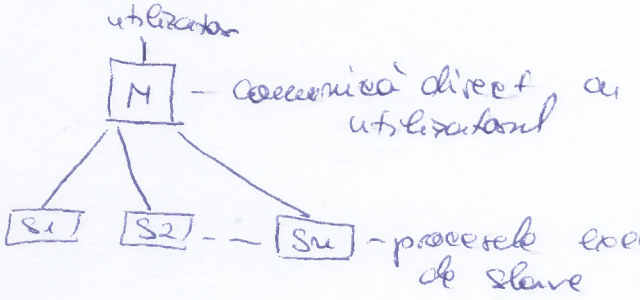
Putem avea blocaje când lucrăm cu pipe-uri.

Sisteme multiprocesor

Examen

Arhitecturi

1. master - slave



2. procesoare dedicate anumitor funcții

Exemplu: procesor dedicat gestiunii întreruperilor și registrelor → partajată unui proc dedicat gestiunii tuturor dispozitivelor periferice.

Procesoarele comunică prin zone de memorie comune. (au un pointer către o zonă de memorie comună)