Cozi de mesaj

• IPC - posedă un apel de forma - xxxget

- msg - ~~cozi~~ de msg
- sem - ~~vector~~ de semafoare
- shm

- xxxctl - apel de sistem

• Deschiderea se face cu ajutorul apelului
• int msgget(key_t cheie, int opt)

- esec - -1
- succes - >1

cheie - poate fi recuperată cu ~~ftok~~
- poate să aibe valoarea IPC_PRIVATE ↳ valabil pentru proces și descendenți

Opțiuni - se creată prin disjuncție cu sau logic (bit cu bit)
- opt IPC_CREAT - dacă nu există îl creată
- dacă există îl deschide

ENOENT (de nu există)

- opt IPC_EXCL - are efect numai în prezența lui IPC_CREAT
- vrea exclusiv create

EEXIST (dacă există)

- 0666

• → permite punerea de mesaje în coadă
• int msgsnd (int id, void *p, int lg, int opt)

adr. din memorie unde este msg. / lungimea msg (sizeof) / opțiuni

- 0 - succes
- -1 - insucces → id-ul nu este recuperat cu msgget; EINVAL (param. invalid)

opt. - IPC_NOWAIT - cod de eroare "-1" - EAGAIN ↳ mai încearcă odată

- mesajul trebuie să înceapă cu un "unsigned long typ;" → tipul msg.

- lg este sizeof (struct.) - sizeof (long)

• Extragem msg din coadă
• int msgrcv (int id, void *p, int lg, long tip, int opt)

{tip >0 - se ia cel mai vechi msg cu tipul specificat
tip =0 - se ia cel ——— ——— indiferent de tip

opt - IPC-NOWAIT | MSG-NOERROR (specific ~~cozilor~~ de msg.)
↳ cod retur "-1" - E2BIG (mesaj prea mare)

Citirea din coadă este distructivă: O dată ce msg a fost citit, el nu se mai află în coadă

- int msgctl (int id, int op, struct msqid_ds *p)
  - ↳ utilizatorul care a creat mesajul
  - → drept de acces
  - ↳ câte msg sunt în coadă

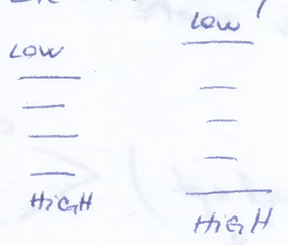
op - IPC_STAT < msgctl umple câmpurile lui p (recuperată caract.)
- IPC-SET - modif setările IPC-ului
- IPC-RMID < al treilea argument nu mai are importanță; op. de a sterge IPC-ul

Vector de semafoare

Semafoare - ~~[illegible]~~ "Dijkstra", val semaforului nu poate fi negativă

Operații: HIGH(S) - incrementarea nr.
LOW(S) - decrementarea nr.

Ex de semafor → mutex - are ca val doar "0" și "1"
→ fiecare proces are o zonă critică / nu vrem să fie întreruptă
- dc un proces intră în zonă critică să nu intre și celălalt



Vector de semafoare

- Apeluri pt vect de sem. unix
  - int semget (key_t cheie, int nr, int opt)
  - struct sembuf - descrie o operație asupra unui sem. atomic
    ↳ unsigned short sem_num; // pe al câtelea sem fac operații direct cu
    short sem_op; // ce operatie fac asupra sem, / putem incrementa ele *, s
    /* >0 - incrementare de semop
    <0 - decrementare
    =0 - aștept ca sem să ajungă pe 0 */
    short sem_flg; /* IPC-NOWAIT */ → ERROR-EAGAIN
    sem-undo - toate proc se anulează la

◦ fcelum care fac op asupra sem.

• int semop (int id, struct sembuf ~~tab~~ *tab, int nr) ← arbitrarea proceselor !

se recupereaza cu semget

câte elem sunt in tab (câte op se fac)

• int semctl (int id, int num, int op, ...) → apel cu signatură variabilă

apel de exploatare

nr sem individual

↳ tipul celui de-al patrulea arg este dat de op

! initializari

◦ IPC_STAT

◦ struct *semid_ds - struct se umple cu caract.

◦ IPC_SET

◦ IPC_RMID

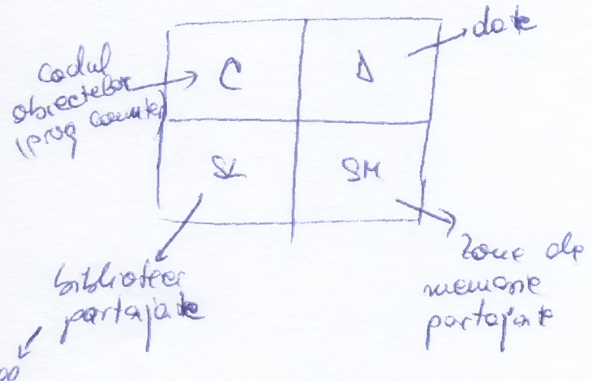
Operatii:

arg 4 int
- GETVAL - imi returnează val sem; în caz de succes
- SETVAL - forțez val sem.

arg 4 pointer catre short
- GETALL - recuperez val tuturor sem.
- SETALL

! SEMCTL nu este folosit pt arbitrarea proceselor, ci SEMOP !

Segmente de memorie partajate



◦ int shmget (key_t cheie, int dim, int opt) la fel ca msgget și semget

dimensiune

- Apeluri specifice de exploatare:

◦ void *(int id, void *adr, int opt)

shmat

↳ null OBLIGATORIU

opt – SHM_RDONLY – read only

• int shmdt (void *adr) → detașare

↳ ce am recuperat la shmat

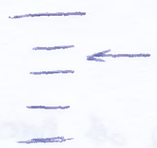
Semnale

signal.h – definește o mulțime de constante

NSIG – nr de semnale disponibile

– fiecare sem. – este identificat printr-un nr întreg, pt.

```
#define SIGCONT 6
# define SIGKILL 9
```



s-a primit un sem., proc intră în kernel mod, se execută o rutină handler (a semnalului)

Dc. handler se termină cu return & întoarce în kernel mode & se execută prog de unde a rămas