11.04.2019

# Curs ~~6~~ 7 - Sisteme de operare

FIFO ♥

$$\begin{tikzcd}
& Pc \arrow[dr, bend left, "5"] \arrow[dr, "2"'] & \\
A \arrow[ur, bend left, "3"] \arrow[ur, "6"'] & & B \arrow[ll, bend left, "4"] \arrow[ll, "1"']
\end{tikzcd}$$

6 :(

round-count-fifo.c → round-count-fifo-a.c (A, B, C)

```
mkfifo c2b b2a a2c

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main(){

    int b2a, a2c, n;

    b2a = open ("b2a", O_RDONLY);
    a2c = open( "a2c", O_WRONLY);
    while ( read(b2a, &n, sizeof(int)) > 0 && n > 0) {
          n--;
          printf ("A: %d -> %d\n", n+1, n);
          write ( a2c, &n, sizeof(int));
    }
    close (b2a);
    close ( a2c);
    return 0;
}
```

round-count-fifo-b.c

```
#include <stdio.h>          #include <sys/types.h>
#include <stdlib.h>         #include <fcntl.h>
#include <unistd.h>         #include <sys/stat.h>
int main(){

    int b2a, c2b, n;

    b2a = open( "b2a", O_WRONLY);
    c2b = open( "c2b", O_RDONLY);
  while(read (c2b, &n, sizeof(int)) > 0 && n > 0) {
          n--;
          printf ("[illegible]: %d -> %d \n", n+1, n);
          write ( [illegible], &n, sizeof (int));
  }
  close (b2a);
  close (c2b);
  return 0;
}
```

round-count fifo-c.c

```
#include ...


int main() {

    int a2c, c26, n;
    a2c = open("a2c", O_RDONLY)
    c26 = open("c26", O_WRONLY);
    n=5;
    write(c26, &n, sizeof(int));
    while (read(a2c, &n, sizeof(int)) > 0 && n>0) {
          n--;
          printf("C : %d → %d\n", n+1, n);
          write(c26, &n, sizeof(int));
  }
  close(a2c);
  close(c26);
  return 0;
}
```

! Să avem o ordine (ordinea alfabetică ??)
apelul open poate primi flagul O_NDELAY
- flagul O_NDELAY poate fi activat pentru pipe folosind pipe2 sau fcntl.
- îi schimbă comportamentul, anulează blocajelor.

| | NORMAL | CU O_NDELAY |
|---|---|---|
| open FIFO - pentru citire | așteaptă ca fifo să fie deschis și pentru scriere | revine imediat, fără a semnala eroare |
| open FIFO - pentru scriere | așteaptă ca FIFO să fie deschis și pentru citire | revine imediat, semnalând eroare errno = ENXIO |
| read - din pipe, FIFO gol | așteaptă NIȘTE date sau până nu mai sunt scriitori<br>întoarce nr. de bytes citiți | revine imediat, întorcând 0 |
| write pipe, FIFO plin | așteaptă câtva spațiu sau până nu mai sunt cititori.<br>întoarce nr. de bytes scriși | revine imediat, întorcând 0 |

shmget → creează
shmat → pointer
shmdt → detach
shmctl → închidere + altele

$\boxed{\begin{array}{l}\text{91234}\\ \to \text{cu secțiuni id !!}\\ \text{chmod 600}\end{array}}$ → memorie partajată

shm-a.c

```
int main(int argc, char **argv) {
    int shm; struct data *p;
    shm = shmget(1234, sizeof(struct data), SHM-CREAT | 0600);
    p = shmat(shm, 0, 0);
    p->stop = 0;
    for (i=0; i<~~[illegible]~~; i++) {
            p->contor = i; sleep(1);
    }
    p->stop = 1;
    shmdt(p);
    shmctl(shm, IPC_RMID, 0);
    return 0;
}
```

```
struct data {
        ~~char msg[64];~~ int contor;
        int stop;
};
```

th-1.c

```
#include <stdio.h>
#include <pthread.h>
int n=0;
void *f(void *a) {
    int i;
    for(i=0; i<10000; i++){
        n++;
    }
    return NULL;
}

int main(int argc, char **argv) {
    pthread_t t[100];
    int i, n=0;
    for(i=0; i<100; i++){
        pthread_create(&t[i], NULL, f, NULL);
    }
    for (i=0; i<100; i++){
        pthread_join(t[i], NULL);
    }
    printf("%d\n", n);
    return 0;
```

```
gcc -Wall -g -o th1 th1.c -lpthread
```

→ de ce se întâmplă asta?
→ de ce nu afișează 1000000 mereu?