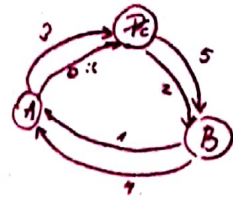


Burs - Systeme de operate

FIFO ♥

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



mtfifo c2b b2a a2c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
int main() {
    int b2a, a2c, m;
    b2a = open("b2a", O_RDONLY);
    a2c = open("a2c", O_WRONLY);
    while (read(b2a, &m, sizeof(int)) > 0 && m > 0) {
        m--;
        printf("A: %d → %d\n", m+1, m);
        write(a2c, &m, sizeof(int));
    }
    close(b2a);
    close(a2c);
    return 0;
}
```

round-count-fifo-b.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <fcntl.h>
#include <sys/stat.h>
int main() {
    int b2a, c2b, m;
    b2a = open("b2a", O_WRONLY);
    c2b = open("c2b", O_RDONLY);
    while (read(c2b, &m, sizeof(int)) > 0 && m > 0) {
        m--;
        printf("B: %d → %d\n", m+1, m);
        write(b2a, &m, sizeof(int));
    }
    close(b2a);
    close(c2b);
    return 0;
}
```

#include . . .

```
int main() {
    int a2c, c2b, M;
    a2c = open("a2c", O_RDONLY);
    c2b = open("c2b", O_WRONLY);
    n = 5;
    write(c2b, &n, sizeof(int));
    while (read(a2c, &n, sizeof(int)) > 0 & &n < 0) {
        n--;
        printf("C: %d -> %d\n", n+1, n);
        write(c2b, &n, sizeof(int));
    }
    close(a2c);
    close(c2b);
    return 0;
}
```

! Ia avem o ordine (ordinea alfabetica ??)

apelul open poate primi flagul O_NODELAY

- flagul O_NODELAY poate fi activat pentru pipe folosind pipe2 sau fcntl.
- ii schimbă comportamentul, anulează blocajelor.

	NORMAL	CU O_NODELAY
open ^{FIFO} pentru citire	asteaptă ca fifo să fie deschis și pentru scriere	revine imediat, fără a semnaliza eroare
open ^{FIFO} pentru scriere	asteaptă ca FIFO să fie deschis și pentru citire	revine imediat, semnalizând eroare errno = ENXIO
read - din pipe, FIFO gol	asteaptă NIȘTE date sau până nu mai sunt scriitori întoarce nr. de bytes citiți	revine imediat, întorcând 0
write pipe, FIFO plin	asteaptă câțiva spații sau până nu mai sunt cititori. întoarce nr. de bytes scriși	revine imediat, întorcând 0

shmget → creează
 shmat → pointer
 shmctl → detaș
 shmctl → include + altă

51234
 → cu secțiș id !!
 chmod 600

→ 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<1000; i++) {
```

```
        p->count = i; sleep(1);
```

```
        p->stop = 1;
```

```
        shmctl(p, 0);
```

```
        shmctl (shm, IPC-RLMID, 0);
```

```
    return 0;
```

```
struct data {
```

```
    char msg[64]; int count;
```

```
}; int stop;
```

shm-b.c

```
#include <stdio.h>
```

```
#include <pthread.h>
```

```
int n = 0;
```

```
void *f(void *a) {
```

```
    int i;
```

```
    for (i=0; i<1000; 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 shm shm-a.c -lpthread

→ de ce se întâmplă asta?

→ de ce nu afișează 1000000 marea?