



Laborator 4

Device Drivere în Linux

Sisteme de Operare 2 (SO2)

Departamentul de Calculatoare

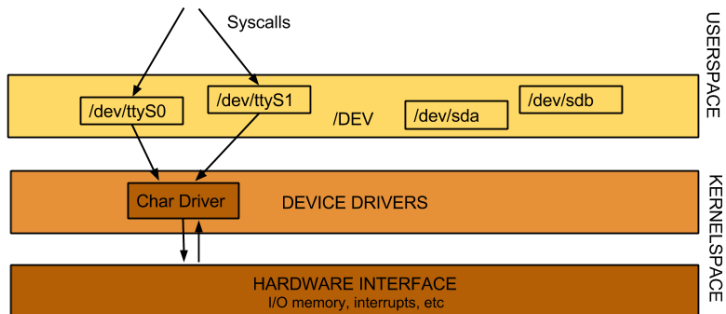
Introducere

Înregistrare/Deînregistrare

Operații

Sincronizare - Cozi de Așteptare

Keywords



- ▶ Construit pentru dispozitive lente
- ▶ Datele sunt(în general) prelucrate în ordine
- ▶ De multe ori prezintă comportament de pipe (se citește dintr-un capăt, se scrie din celălalt)
- ▶ Mouse, serială, tastatură, ...

Introducere

Înregistrare/Deînregistrare

Operații

Sincronizare - Cozi de Așteptare

Keywords

- ▶ fișiere speciale de tip 'c'
- ▶ de obicei în devfs (dev)
- ▶ Identificatori
 - ▶ Major number - tipul de dispozitiv (driver-ul)
 - ▶ minor number - dispozitivul
- ▶ Create cu mknod
 - ▶ # mknod /dev/myfirstdev c 42 0

```
crw-rw---- 1 root dialout 4, 64 feb 12 21:48 /dev/ttyS0
crw-rw---- 1 root dialout 4, 65 feb 12 21:48 /dev/ttyS1
crw-rw---- 1 root dialout 4, 74 feb 12 21:48 /dev/ttyS10
crw-rw---- 1 root dialout 4, 75 feb 12 21:48 /dev/ttyS11
crw-rw---- 1 root dialout 4, 76 feb 12 21:48 /dev/ttyS12
```

- ▶ **register_chrdev_region** alocă o zonă din spațiul (major,minors) driver-ului ce apelează
- ▶ **alloc_chrdev_region** nu necesită specificarea unui major (alocă dinamic)
- ▶ Deînregistrarea se face cu **unregister_chrdev_region**

- ▶ Inițializată cu `cdev_init(..)`
 - ▶ Structura conține și `file_operations`, listă cu operațiile implementate de driver-ul device-ului
- ▶ Adăugată în sistem cu `cdev_add(..)`
 - ▶ Marchează momentul din care driver-ul poate fi folosit
- ▶ Ștergerea din sistem cu `cdev_del(..)`

Introducere

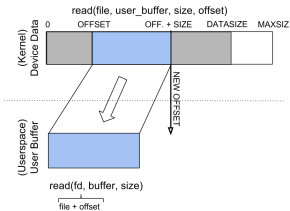
Înregistrare/Deînregistrare

Operații

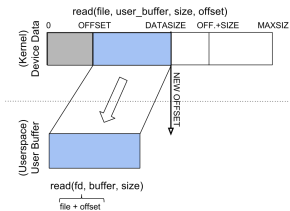
Sincronizare - Cozi de Așteptare

Keywords

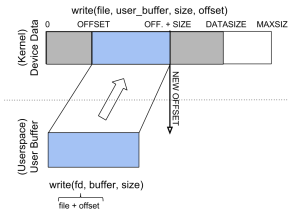
- ▶ **open** - corespondentul apelului de sistem open din userspace, verifică drepturi de utilizare și asigură eventuala arbitrarea accesului
- ▶ **release** - corespondentul syscall-ului close, realizează opusul funcției open (eliberează lock-uri, resurse)
- ▶ **read** - realizează transferul de date dinspre kernel înspre userspace (inter-buffer, nu neapărat cu dispozitivul în sine)
- ▶ **write** - analog READ, doar că sensul este dinspre userspace spre kernel
- ▶ **ioctl** - comenzi speciale efectuate asupra dispozitivului



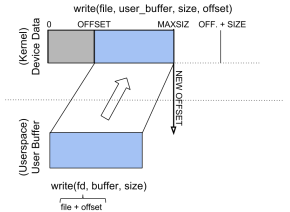
(a) read



(b) read less



(c) write



(d) write less

Inode

- ▶ Un fișier din perspectiva filesystem-ului
- ▶ dimensiune
- ▶ drepturi
- ▶ timp de acces
- ▶ pointer către dispozitivul caracter (pentru fișierele specifice)

File

- ▶ Un fișier deschis, practic o structură prezentă numai între o pereche open-release
- ▶ Modul de deschidere - `f_mode`
- ▶ Flag-uri de deschidere - `f_flag`
- ▶ Poziția în fișier - `f_pos`
- ▶ Operațiile posibile - `f_op`

Ce trebuie făcut în userspace

- ▶ Pasarea unui pointer valid
- ▶ Verificarea erorii returnate (if any)
- ▶ Verificarea numărului de octeți citiți/scriși (un driver nu este obligat să scrie tot în buffer-ele proprii)

Ce trebuie făcut în kernelspace

- ▶ Verificarea pointer-ului din userspace (se folosesc macro-urile de copiere)
- ▶ Returnarea unui cod de eroare coerente cu motivul erorii
- ▶ Marcarea numărului de octeți scriși/citiți (e bine să se încerce procesarea întregii comenzi)
- ▶ Deplasarea index-ului în fișier (câmpul offset)

- ▶ Prin copierea datelor între buffer-e
- ▶ Accesul direct la paginile din userspace este interzis! (De ce?)
- ▶ Accesul se face prin macro-uri specializate
 - ▶ `put_user` pune o singură valoare (până în 64 biți) la o locație
 - ▶ `get_user` citește o singură valoarea de la o locație din userspace
 - ▶ `copy_to_user(dest_user,sursă_kernel, n)`
 - ▶ `copy_from_user(dest_kernel,sursă_user, n)`

Introducere

Înregistrare/Deînregistrare

Operații

Sincronizare - Cozi de Așteptare

Keywords

- ▶ Oferă o soluție de sincronizare ce nu necesită busy-waiting
- ▶ Un proces se înscrie într-o coadă de așteptare (o listă) pentru a fi planificat doar atunci când o condiție este îndeplinită
- ▶ Mecanism de wait/wake-up

- ▶ **init_waitqueue_head()**
 - ▶ inițializarea cozii
- ▶ **wait_event** și **wait_event_interruptible**
 - ▶ adaugă thread-ul la coadă cât timp condiția nu este adevărată
- ▶ **wait_event_timeout** și **wait_event_interruptible_timeout**
 - ▶ adaugă thread-ul la coadă cât timp condiția nu este adevărată sau până la expirarea timeout-ului
- ▶ **wake_up()**
 - ▶ pune toate thread-urile în starea TASK_RUNNING (pot fi planificate)
- ▶ **wake_up_interruptible()**
 - ▶ pune doar thread-urile din starea INTERRUPTIBLE în starea TASK_RUNNING (pot fi planificate)

Introducere

Înregistrare/Deînregistrare

Operații

Sincronizare - Cozi de Așteptare

Keywords

- ▶ device node
- ▶ major
- ▶ minor
- ▶ file_operations
- ▶ file
- ▶ inode
- ▶ open/release
- ▶ read/write
- ▶ put/get_user
- ▶ copy_from/to_user
- ▶ wait_queue