Opis zastosowań i algorytmów dotyczących **tablic rozdzielczych** dla urządzeń w systemie **Linux**. Oparty o wersję jądra **2.2.17**.

Przygotowanie:
Paweł Witkowski
pw181102@zodiac.mimuw.edu.pl
III rok

6 stycznia 2001

Rozdział 1

Obsługa tablic rozdzielczych

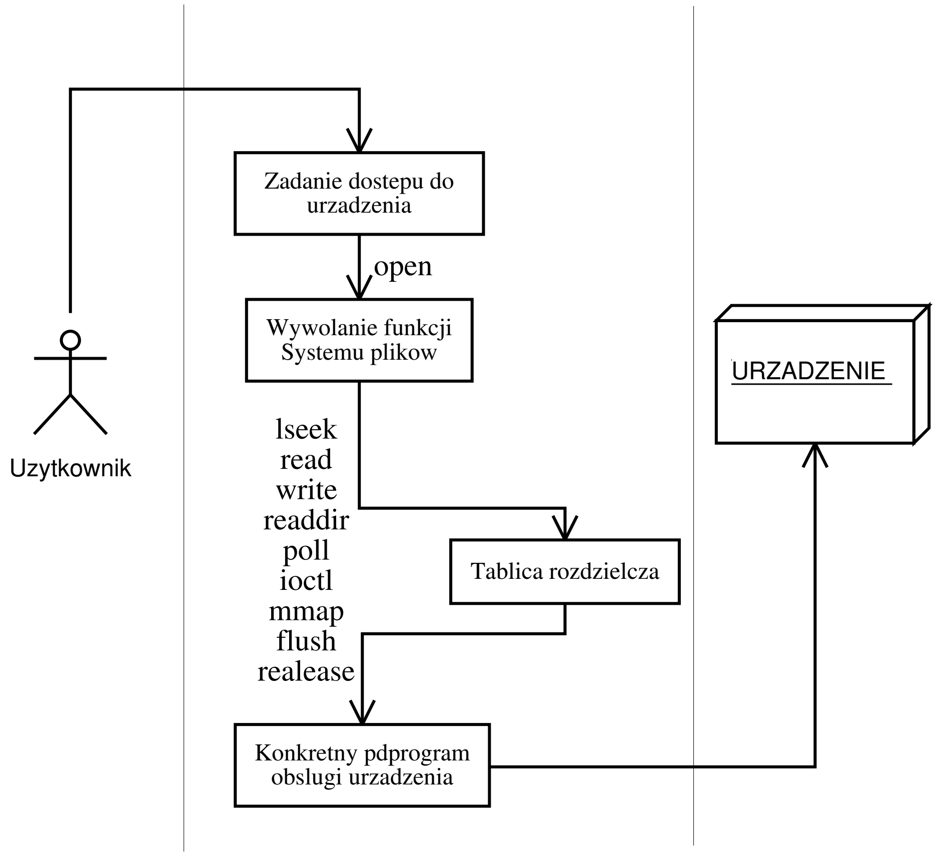
1.1 Wstęp

Każde urządzenie do prawidłowego i zgodnego z przeznaczeniem działania potrzebuje sterownika, czyli programu, który udostępni funkcjonalność urządzenia za pomoca interfejsu funkcji. Powinien on być tak napisany aby nie było możliwości nieprawidłowego wykorzystywania urządzenia. Tablice rozdzielcze to struktury służące do gromadzenia informacji o funkcjach implementujących poszczególne operacje realizowane przez sterownik. Mamy dwie takie tablice odpowiednio dla urządzeń znakowych **chrdevs** i blokowych **blkdevs** . Urządzenie identyfikowane jest przez jego numer nadrzędny **MAJOR**, który jest jednocześnie indeksem w tablicy, oraz nazwę pliku specjalnego odpowiadającego urządzeniu. Te informacje w zupełności wystarczą aby jednoznacznie wskazać sterownik danego urządzenia. Sterownik potrzebuje jeszcze numeru drugorzędnego **MINOR** aby jednoznacznie wskazać urządzenie, ale oczywiście tego nie należy trzymać w tablicy rozdzielczej. Dla każdego urządzenia tablica zawiera adresy funkcji implementowanych przez sterownik wraz z nazwą tego urządzenia.

1.2 Sposób korzystania z tablic rozdzielczych

Każde urządzenie w systemie **Linux** jest reprezentowane przez plik specjalny i znane są dobrze operacje jakie można na takim pliku wykonywać. Są to np. *read*, *write*, *open*, *close* itd. Trzeba jednak pamietać, że do prawidłowego dzialania operacje te muszą być zaimplementowane z myślą o konkretnym urządzeniu, tzn. nie są uniwersalne, tylko charakterystyczne dla każdego sterownika. System udostępnia za to uniwersalny interfejs do tych funkcji co znacznie ułatwia programowanie i czyni sterowniki bardziej przenośnymi.
Schemat korzystania z tablicy rozdzielczej jest przedstawiony na rysunku 1.1.

Na rysunku rozdzieliliśmy operacje na pliku specjalnym umieszczając funkcje *open* przed pozostałymi. Jest bowiem konieczne aby wykonać tę funkcję zanim zaczniemy korzystać z urządzenia. Ma ona za zadanie pobrać z tablicy rozdzielczej odpowiednią strukturę zawierającą wskaźniki na operacje dozwolone na pliku specjalnym. Adresy tych operacji zapisywane są w strukturze *file* pliku specjalnego i od tej pory wywoływanie innych operacji jest już dozwolone.



Rysunek 1.1: Wywolanie funkcji sterownika

1.3 Inicjalizowanie tablic rozdzielczych

Jak wiadomo w systemie **Linux** część podprogramów obsługi urządzen jest wkompilowana w jądro, a kolejne mozna ładować i odwoływać w trakcie działania systemu. Zatem nasz opis inicjalizacji tablic rozdzielczych również podzielimy na dwie części. Wygodniej jednak będzie zacząć od opisu rejestrowania pojedynczego urządzenia dynamicznie, gdyż jest to jakby opis jednej z cegiełek, z których składa się inicjalizacja tablic rozdzielczych podczas uruchamiania systemu.

Każdy moduł sterownika urządzenia ładowany dynamicznie musi mieć zdefiniowaną funkcję **init_module()**, która wywoływana jest podczas ładowania (wywołanie **insmod**). Jest to odpowiednie miejsce aby zarejestrować operacje na urządzeniu w odpowiedniej tablicy rozdzielczej. Odbywa się to przy pomocy funkcji:

- **register_blkdev()** lub
- **register_chrdev()**

odpowiednio dla urządzeń blokowych lub znakowych.

Algorytm funkcji rejestrującej urządzenie:

Algorytm 1 register_blkdev

Dane wejściowe: usigned int major, const char * name, struct file_operations *fops
Wynik: 0 : gdy major != 0 i miejsce o indeksie major bylo wolne
indeks w tablicy : gdy major = 0 i bylo wolne miejsce

```
jeśli major == 0 to
  major = MAX_BLKDEV
  dopóki major > 0 wykonuj
    jeśli miejsce wolne o indeksie major to
      blkdevs[major].name = name
      blkdevs[major].fops = fops
      zwróć(major)
    w przeciwnym przypadku
      major = major - 1
    koniec jeśli
  koniec dopóki
w przeciwnym przypadku
  jeśli major > MAX_BLKDEV to
    zwróć -EINVAL
  w przeciwnym przypadku
    jeśli blkdevs[major] != NULL i blkdevs[major].fops != fops to
      zwróć -EBUSY
    w przeciwnym przypadku
      blkdevs[major].name = name
      blkdevs[major].fops = fops
    koniec jeśli
  koniec jeśli
koniec jeśli
```

Jak widać z opisu algorytmu kluczową rolę pełni numer **MAJOR** urządzenia. Moduł może pobrać swój numer nadrzędny z pliku **linux/major.h** lub poprosić o przydzielenie tego numeru dynamicznie. Ten drugi sposób zalecany jest w przypadku gdy nie ma standardowego numeru dla danego urządzenia. Lepiej nie szukać wolnych miejsc w tablicy na własną rękę. Przydzielony numer pojawi się w pliku **devices.txt**. Trzeba ten numer znać również podczas usuwania modułu (wywołanie **rmmod**), gdyż w przeciwnym wypadku miejsce w tablicy nie zostanie zwolnione. Faktycznie spowoduje to więcej komplikacji, gdyż miejsce to będzie nadal zajęte przez moduł, który już nie istnieje, więc nie tak prosto będzie to miejsce zwolnić.

Inicjalizowanie tablic rozdzielczych podczas startu systemu odbywa się za pomocą funkcji **chr_dev_init()** oraz **blk_dev_init()**. Są one bardzo podobne i różnią się właściwie tylko początkową częścią, która jest specyficzna zarówno dla urządzeń znakowych jak i blokowych. Obie funkcje korzystają z pliku konfiguracyjnego jądra (**linux/config.h** w celu wykrycia urządzeń, które należy zarejestrować w tablicach rozdzielczych. Dla urządzeń blokowych istnieje poza tablicą **blkdevs** druga tablica rozdzielcza nazwana **blk_dev**. Jest ona potrzebna aby przechowywać informacje dotyczące żądań odczytu i zapisu do urządzenia blokowego. Zawarty jest w niej adres funkcji kolejkującej te żądania. Ma ona ten sam rozmiar co opisana wcześniej tablica **blkdevs** i również jest indeksowana numerami nadrzędnymi urządzeń.

Opis funkcji inicjalizujących urządzenia blokowe i znakowe:

- chr_dev_init()
  Najpierw próbuje się zarejestrować sterowniki dotyczące obsługi pamięci podając *major = 1* jako indeks w tablicy **chrdev**. Jeśli to się nie powiedzie wypisywany jest komunikat o błędzie, ale inicjalizowanie jest kontynuowane. Dalej sprawdzane są urządzenia z listy w pliku **config.h** i inicjalizowane gdy są wkompilowane w jądro (to oznacza, że odpowiednie stałe są zdefiniowane w tym pliku). Inicjalizacja odbywa się poprzez wywołanie funkcji inicjalizującej dla danego urządzenia. Ta funkcja wywołuje dopiero **register_chrdev()**.
- blk_dev_init() Najpierw inicjalizowana jest tablica **blk_dev** zawierająca informacje potrzebne do kolejkowania żądań odczytu i zapisu. Następna część wygląda tak samo jak w przypadku tablicy urządzeń znakowych, czyli wywoływane są funkcje inicjalizujące dla urządzeń, ktorych podprogramy obsługi są wkompilowane w jądro.

Gdy moduł zostaje usunięty z systemu to musi zadbać o zwolnienie odpowiadającej mu pozycji w tablicy rozdzielczej. Dobrym (i chyba jedynym odpowiednim) miejscem do uczynienia tego jest funkcja **cleanup_module()** wywoływana zawsze przy usuwaniu modułu. Należy w niej umieścić jedną z funkcji:

- **unregister_chrdev()**
- **unregister_blkdev()**

podając jako parametr numer nadrzędny urządzenia lub indeks, który został przydzielony podczas wywoływania funkcji **register_xxxdev()**, oraz nazwę pliku specjalnego.

Rozdział 2

Analiza kodu źródłowego

Deklaracje i definicje tablic rozdzielczych oraz struktur w nich zawartych znajdują się w pliku **fs/devices.c**. W obecnej wersji jądra ilość urządzeń znakowych i blokowych jest definiowana przez następujące stałe, umieszczone w pliku **linux/major.h**:

- **MAX_CHRDEV** 255
- **MAX_BLKDEV** 255

Warto zwrócić uwagę, że to oznacza, że naprawdę możemy mieć 254 urządzenia każdego z typów, gdyż pozycja 0 nie jest używana w tablicach rozdzielczych. Jest jednak wskazanie twórców jądra, aby nie zmieniać tych liczb na 256 (czyli tyle ile możemy zapisać na ośmiu bitach) w celu zostawienia pola dla przyszłych rozszerzeń.

- Struktura przechowywana w tablicy:
  name: nazwa pliku specjalnego urządzenia.
  fops: struktura zawierająca wskaźniki na funkcje implementujące operacje na urządzeniu.

  ```
  struct device_struct {
  const char * name;
  struct file_operations * fops;
  };
  ```

- Tablica rozdzielcza urządzen znakowych i jej zawartość początkowa:

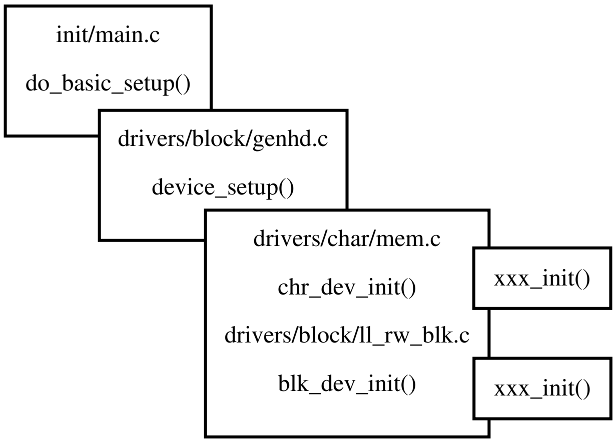
  ```
  static struct device_struct chrdevs[MAX_CHRDEV] = {
  { NULL, NULL },
  };
  ```

- Tablica rozdzielcza urządzeń blokowych i jej zawartość początkowa:

```
static struct device_struct blkdevs[MAX_BLKDEV] = {
{ NULL, NULL },
};
```

W pliku **fs/devices.c** znajdują się również definicje funkcji **chrdev_open()** oraz **blkdev_open()**. Jak już było powiedziane wcześniej, przed wywołaniem jakiejkolwiek operacji na pliku specjalnym reprezentującym urządzenie, trzeba wywołać jedną z tych funkcji w celu zainicjalizowania struktury **file**. W strukturze tej w polu **f_op** umieszcza się strukturę **file_operations** pobraną z tablicy rozdzielczej. Następnie próbuje się wywołać funkcję **open()** której adres mamy z pobranej struktury **file_operations**. Sposób pobierania tej struktury jest opisany w dalszej części.

Kolejność wywoływania funkcji inicjalizujących podczas startu systemu:



Rysunek 2.1: Inicjalizacja tablic rozdzielczych

Rozdział 3

Dodatkowe funkcje dla tablic rozdzielczych

- **get_device_list(char *)**
  Na podaną zmienną wypisuje po kolei wszystkie urządzenia, najpierw znakowe, później blokowe, podając numer główny i nazwę urządzenia.
- **get_blkfops(unsigned int)**
  Zwraca strukurę **file_operations** dla urządzenia blokowego o podanym numerze głównym
- **get_chrfops(unsigned int, unsigned int)**
  Zwraca strukturę **file_operations** dla urządzenia znakowego o podanych numerach: głównym i drugorzędnym.