Obsługa kolejek fifo

Agata Kendziorek

19 grudnia 2000

Spis treści

1 Obsługa kolejek fifo **5**
1.1 Wstęp 5
1.2 Operacje na kolejkach fifo 5
1.3 Reprezentacja kolejek fifo w systemie 6
1.3.1 Inacjalizacja kolejki 6
1.3.2 Algorytm otwierania kolejki - **fifo_open** 7
1.3.3 Pozostałe operacje na kolejce 8

Rozdział 1

Obsługa kolejek fifo

1.1 Wstęp

Kolejki **fifo (łącza nazwane)** służą do komunikacji pomiędzy procesami. W odróżnieniu od **łączy nienazwanych (pipe)**, stworzona kolejka fifo występuje w systemie jako plik i jest dostępna dla każdego procesu. Implementacja kolejek fifo znajduje się w pliku *fs/fifo.c*.

1.2 Operacje na kolejkach fifo

Operacje na kolejkach fifo dostępne w systemie dla procesów użytkownika:

- stworzenie kolejki fifo
- otwarcie kolejki do pisania lub czytania
- pisanie do kolejki
- czytanie z kolejki
- zmiana trybu
- usunięcie kolejki

Kolejkę można stworzyć za pomocą funkcji
`int mkfifo(const char *pathname, mode_t mode)`,
gdzie `pathname` jest nazwą tworzonej kolejki, zaś `mode` określa prawa dostępu do kolejki.
Alternatywnym sposobem stworzenia kolejki jest wywołanie funkcji
`int mknod(const char *pathname, mode_t mode, dev_t dev)`.
Znaczenie pierwszych dwóch parametrów jest identyczne, jak w poprzednim przypadku, natomiast `dev`, który określa, jakiego rodzaju plik ma zostać stworzony, musi byc równy `S_IFIFO`.

Do otwarcia kolejki fifo służy funkcja
`int open(const char *pathname, int flags)`.
Parametr `pathname` to nazwa kolejki, którą chcemy otworzyc, a `flags` okresla w jakim celu otwieramy kolejkę (`O_RDONLY`, `O_WRONLY` lub `O_RDWR`). Kolejki fifo domyślnie są otwierane synchronicznie tzn. jeżeli proces chce otworzyc kolejkę np. do czytania, to zostanie zawieszony jeżeli jeszcze nikt nie czeka na otwarcie tejże kolejki do pisania, chyba że otwiera ją w trybie nieblokującym. Próba otwarcia kolejki do pisania w trybie nieblokujacym zakończy się błędem, jeśli żaden proces nie otworzył lub nie czeka na otwarcie danej kolejki do czytania. Wynikiem wywołania funkcji open jest deskryptor pliku reprezentującego kolejkę.

Pisanie do kolejki fifo odbywa sie za pomocą zwyklej komendy
`ssize_t write(int fd, const void *buf, size_t size)`
gdzie `fd` jest deskryptorem kolejki, `buf` wiadomością, którą chcemy przesłać, natomiast `size` wielkoscią bufora `buf`.

Czytanie z kolejki fifo odbywa się za pomocą zwykłej funkcji
`ssize_t read(int fd, void *buf, size_t count)`.
Parametr `fd` to deskryptor kolejki, `buf` jest zmienną, do której wczytujemy wiadomość, a `count` ilością bajtów, jaką chcemy przeczytać. Wynikiem funkcji jest liczba odczytanych bajtów.

Zmiany trybu kolejki można dokonać za pomocą funkcji
`int fcntl(int fd, int cmd, long arg)`.
Argument `fd` jest deskryptorem kolejki, `cmd` określa, jakiej operacji chcemy dokonać, zaś `arg` jest nowym trybem.

Kolejkę fifo można usunąć za pomocą funkcji
`int unlink(const char *pathname)` .
Parametr `pathname` to nazwa kolejki do usunięcia.

1.3 Reprezentacja kolejek fifo w systemie

W tym rozdziale opisane są struktury danych i algorytmy służące do obsługi kolejek fifo.

1.3.1 Inicjalizacja kolejki

Podczas tworzenia kolejki fifo wywoływana jest funkcja
`void init_fifo(struct inode * inode)`,
która inicjalizuje strukturę i-węzła kolejki. W strukturze inode iwęzła jest unia u, a w niej strutura `struct pipe_inode_info pipe_i`. Funkcja `init_fifo` inicjalizuje na zera pola struktury `pipe_i`, przechowującej dane o łączu (jest to

taka sama struktura, jak dla łącza nienazwanego).

Skruktura ta jest zadeklarowana w pliku źródłowym *include/linux/pipe_fs_i.h*

```
struct pipe_inode_info {
   struct wait_queue * wait;
       /* procesy oczekujące */
   char * base;
       /* miejsce na dane (bufor cykliczny) */
   unsigned int start;
       /* początek wolnego miejsca w buforze base */
   unsigned int lock;
       /* czy kolejka jest zajęta */
   unsigned int rd_openers;
       /* liczba procesów otwierających kolejkę do czytania */
   unsigned int wr_openers;
       /* liczba procesów otwierających kolejkę do pisania */
   unsigned int readers;
       /* liczba czytelników */
   unsigned int writers;
       /* liczba pisarzy */
};
```

Funkcja `init_fifo` inicjalizuje też operacje i-węzła. Jedyna operacja, jaka zostaje udostępniona to
`static int fifo_open(struct inode * inode, struct file * filp)`,
której kod znajduje się w pliku *fs/fifo.c*. Wynikiem działania tej funkcji może być 0 oznaczające, że otwarcie kolejki się udało lub kod błędu.

1.3.2 Algorytm otwierania kolejki - fifo_open

1. Sprawdzamy tryb otwarcia
2. Jeśli otwieramy tylko do czytania
   - Udostępniamy dla pliku `filp` operacje `connecting_fifo_fops`, które są zdefiniowane w pliku `pipe.c`.
   - Jeżeli nie było czytelników, to budzimy wszystkich, którzy spią na kolejce `wait`
   - Jeżeli otwieramy w trybie blokującym i nie ma pisarzy, zwiększamy `rd_openers` i zasypiamy.
   - Po obudzeniu się zmniejszamy `rd_openers` i budzimy pozostałych.
   - Czekamy, aż się obudzą wszyscy, którzy chcą otworzyć kolejkę do pisania
   - Jeśli są pisarze, udostępniamy dla pliku `filp` operacje `read_fifo_fops`, zdefiniowane w pliku *fs/pipe.c*.

3. Jesli otwieramy do pisania

    - Jeżeli nie ma czytelników i otwieramy w trybie nieblokującym, to wychodzimy z błędem
    - Udostępniamy dla pliku `filp` operacje `write_fifo_fops`, które sa zdefiniowane w pliku *fs/pipe.c*.
    - Jeżeli nie było pisarzy, to budzimy wszystkie procesy czekające na kolejce `wait`
    - Jeśli nie ma czytelników, to zwiększamy `wr_openers` i zasypiamy.
    - Po obudzeniu się zmniejszamy `wr_openers` i budzimy pozostalych.
    - Czekamy, aż się obudzą wszyscy, którzy chcą otworzyć kolejkę do pisania

4. Jeśli otwieramy do pisania i czytania

    - Udostępniamy dla pliku `filp` operacje `rdwr_fifo_fops`, które są zdefiniowane w pliku *fs/pipe.c*.
    - Jeżeli nie było czytelników lub pisarzy, to budzimy wszystkie procesy czekające na kolejce `wait`

5. Jeśli był podany jakiś inny tryb, zwracamy błąd

6. Bierzemy wolną stronę pamieci

7. Jeśli była już przydzielona pamięć dla tej kolejki, zwalniamy wziętą stronę i kończymy działanie (z sukcesem)

8. Jeśli nie dostaliśmy strony, zwracamy bład

9. Dokonujemy końcowej inicjalizacji

10. Kończymy dzialanie

1.3.3 Pozostałe operacje na kolejce

Funkcja `fifo_open` w zależności od trybu otwarcia kolejki udostępnia otwierającemu procesowi odpowiednie operacje. Możliwe są cztery zbiory operacji:

- connecting_fifo_fops
- read_fifo_fops
- write_fifo_fops
- rdwr_fifo_fops

Każdy z tych zbiorów jest typu `struct file_operations`. Struktura ta zawiera wskaźniki do wszystkich możliwych operacji, jakie można wykonywać na pliku. Jeżeli pewne pole tej struktury jest równe `NULL`, to znaczy, że dana operacja jest zabroniona na tym pliku, w przeciwnym wypadku znajduje się tam wskaźnik do właściwej funkcji obsługi.

Do obsługi operacji na kolejkach fifo wykorzystywane są te same funkcje jak w przypadku łączy nienazwanych, zadeklarowane w pliku *fs/pipe.c*.