Obsługa plików

Systemy Operacyjne 2 — laboratorium

Mateusz Hołenko

25 września 2011

Plan zajęć

1. Pliki w systemie Linux
   - i-węzły
   - deskryptory plików
2. Operacje na plikach
   - otwieranie i zamykanie
   - zapis i odczyt
   - skanowanie i skracanie
   - tworzenie i usuwanie
3. Operacje na deskryptorach
4. Zadania

Pliki w systemie Linux

I-węzły (ang. I-nodes)

- plik — podstawowa forma przechowywania danych w komputerze
- dla systemu operacyjnego plik jest ciągiem bajtów przechowywanych najczęściej w pamięci nieulotnej

i-węzeł struktura systemu plików w tradycyjnych Linux'ach zawierająca metainformacje o pliku, m.in.:

- właściciel pliku, prawa dostępu
- typ pliku, czasy dostępu
- ilość dowiązań, rozmiar pliku
- wskaźnik na fizyczną lokalizację pliku na dysku

- i-węzeł **nie zawiera** nazwy pliku ani ścieżki do niego !

I-węzły (ang. I-nodes)

- plik — podstawowa forma przechowywania danych w komputerze
- dla systemu operacyjnego plik jest ciągiem bajtów przechowywanych najczęściej w pamięci nieulotnej

i-węzeł struktura systemu plików w tradycyjnych Linux'ach zawierająca metainformacje o pliku, m.in.:

- właściciel pliku, prawa dostępu
- typ pliku, czasy dostępu
- ilość dowiązań, rozmiar pliku
- wskaźnik na fizyczną lokalizację pliku na dysku

- i-węzeł **nie zawiera** nazwy pliku ani ścieżki do niego !

I-węzły (ang. I-nodes)

- plik — podstawowa forma przechowywania danych w komputerze
- dla systemu operacyjnego plik jest ciągiem bajtów przechowywanych najczęściej w pamięci nieulotnej

i-węzeł struktura systemu plików w tradycyjnych Linux'ach zawierająca metainformacje o pliku, m.in.:

- właściciel pliku, prawa dostępu
- typ pliku, czasy dostępu
- ilość dowiązań, rozmiar pliku
- wskaźnik na fizyczną lokalizację pliku na dysku

- i-węzeł **nie zawiera** nazwy pliku ani ścieżki do niego !

Deskryptory plików

- system plików przechowuje informacje o plikach w globalnej tablicy i-węzłów
- użytkownik odwołuje się do plików przez **nazwy**, systemu operacyjny przez **indeks i-węzła w globalnej tablicy** (identyfikator i-węzła, deskryptor pliku)
- każdy proces posiada lokalną tablicę i-węzłów (tablicę deskryptorów plików) zawierającą pliki skojarzone z procesem
  - tablica jest niedostępna bezpośrednio dla programisty
  - odwołuje się do niej przez wywołania funkcji systemowych
  - domyślnie tablica zawiera trzy wpisy

0	stdin	standardowe wejście
1	stdout	standardowe wyjście
2	stderr	standardowe wyjście diagnostyczne

Deskryptory plików

- system plików przechowuje informacje o plikach w globalnej tablicy i-węzłów
- użytkownik odwołuje się do plików przez **nazwy**, systemu operacyjny przez **indeks i-węzła w globalnej tablicy** (identyfikator i-węzła, deskryptor pliku)
- każdy proces posiada lokalną tablicę i-węzłów (tablicę deskryptorów plików) zawierającą pliki skojarzone z procesem
  - tablica jest niedostępna bezpośrednio dla programisty
  - odwołuje się do niej przez wywołania funkcji systemowych
  - domyślnie tablica zawiera trzy wpisy

0	stdin	standardowe wejście
1	stdout	standardowe wyjście
2	stderr	standardowe wyjście diagnostyczne

Deskryptory plików

- system plików przechowuje informacje o plikach w globalnej tablicy i-węzłów
- użytkownik odwołuje się do plików przez **nazwy**, systemu operacyjny przez **indeks i-węzła w globalnej tablicy** (identyfikator i-węzła, deskryptor pliku)
- każdy proces posiada lokalną tablicę i-węzłów (tablicę deskryptorów plików) zawierającą pliki skojarzone z procesem
  - tablica jest niedostępna bezpośrednio dla programisty
  - odwołuje się do niej przez wywołania funkcji systemowych
  - domyślnie tablica zawiera trzy wpisy

0	stdin	standardowe wejście
1	stdout	standardowe wyjście
2	stderr	standardowe wyjście diagnostyczne

Operacje na plikach i deskryptorach

Otwieranie i zamykanie plików

- otwieranie plików
  - `pathname` — nazwa pliku (wraz ze ścieżką)
  - `flags` — flagi otwarcia pliku, m.in.: `O_WRONLY`, `O_RDONLY`, `O_RDWR`, `O_APPEND`, `O_CREAT`, `O_TRUNC`
  - `mode` — prawa dostępu tworzonego pliku

```
int open(const char* pathname, int flags)
int open(const char* pathname, int flags, mode_t mode)
```

- zamykanie pliku
  - `fd` — deskryptor zamykanego pliku

```
int close(int fd)
```

Otwieranie i zamykanie plików

- otwieranie plików
  - `pathname` — nazwa pliku (wraz ze ścieżką)
  - `flags` — flagi otwarcia pliku, m.in.: `O_WRONLY`, `O_RDONLY`, `O_RDWR`, `O_APPEND`, `O_CREAT`, `O_TRUNC`
  - `mode` — prawa dostępu tworzonego pliku

```
int open(const char* pathname, int flags)
int open(const char* pathname, int flags, mode_t mode)
```

- zamykanie pliku
  - `fd` — deskryptor zamykanego pliku

```
int close(int fd)
```

Otwieranie i zamykanie plików — przykładowy program

```c
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
int main(int argc, char* argv[])
{
  int fd;
  if (argc != 2)
  {
    printf("Usage: fexist [filepath]\n");
    return 1;
  }
  fd = open(argv[1], O_RDONLY);
  if (fd == -1)
  {
    printf("Couldn't open file %s (error code %d)\n", argv[1], errno);
    perror("Opening file");
    return 2;
  }
  else
  {
    printf("File %s opened successfully!\n", argv[1]);
    if (close(fd) == -1)
    {
      printf("Couldn't close file %s (error code %d)\n", argv[1], errno);
      perror("Closing file");
    }
  }
  return 0;
}
```

Odczyt i zapis danych

- odczyt danych z plików
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `buf` — adres początku obszaru pamięci, w którym zostaną umieszczone odczytane dane
  - `count` — liczba bajtów do odczytu

```
ssize_t read(int fd, void* buf, size_t count)
```

- zapis danych do pliku
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `buf` — adres początku obszaru pamięci, zawierającego blok danych do zapisania
  - `count` — liczba bajtów do zapisania

```
ssize_t write(int fd, const void* buf, size_t count)
```

Odczyt i zapis danych

- odczyt danych z plików
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `buf` — adres początku obszaru pamięci, w którym zostaną umieszczone odczytane dane
  - `count` — liczba bajtów do odczytu

```
ssize_t read(int fd, void* buf, size_t count)
```

- zapis danych do pliku
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `buf` — adres początku obszaru pamięci, zawierającego blok danych do zapisania
  - `count` — liczba bajtów do zapisania

```
ssize_t write(int fd, const void* buf, size_t count)
```

Odczyt i zapis danych — przykładowy program

```
...

char buf[20];
int n;

while ((n = read(fd, buf, 20)) > 0)
{
  write(1, buf, n);
}

...
```

Przesuwanie wskaźnika bieżącej pozycji i skracanie

- przesuwanie wskaźnika
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `offset` — wielkość przesunięcia
  - `count` — odniesienie: `SEEK_SET`, `SEEK_CUR`, `SEEK_END`

```
off_t lseek(int fd, off_t offset, int whence)
```

- skracanie pliku
  - `path` — nazwa pliku (wraz ze ścieżką)
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `length` — docelowa długość pliku

```
int truncate(const char* path, off_t length)
int ftruncate(int fd, off_t length)
```

Przesuwanie wskaźnika bieżącej pozycji i skracanie

- przesuwanie wskaźnika
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `offset` — wielkość przesunięcia
  - `count` — odniesienie: `SEEK_SET`, `SEEK_CUR`, `SEEK_END`

```
off_t lseek(int fd, off_t offset, int whence)
```

- skracanie pliku
  - `path` — nazwa pliku (wraz ze ścieżką)
  - `fd` — deskryptor pliku (uzyskany z funkcji `open`)
  - `length` — docelowa długość pliku

```
int truncate(const char* path, off_t length)
int ftruncate(int fd, off_t length)
```

Tworzenie i usuwanie plików

- tworzenie pliku
  - `pathname` — nazwa pliku (wraz ze ścieżką)
  - `mode` — prawa dostępu tworzonego pliku

```
int creat(const char* pathname, mode_t mode)
```

- usuwanie pliku
  - `pathname` — nazwa pliku (wraz ze ścieżką)

```
int unlink(const char* pathname)
```

Tworzenie i usuwanie plików

- tworzenie pliku
  - `pathname` — nazwa pliku (wraz ze ścieżką)
  - `mode` — prawa dostępu tworzonego pliku

```
int creat(const char* pathname, mode_t mode)
```

- usuwanie pliku
  - `pathname` — nazwa pliku (wraz ze ścieżką)

```
int unlink(const char* pathname)
```

Operacje na deskryptorach

- tworzenie duplikatu deskryptora
  - `oldfd` — deskryptor do powielenia
  - `newfd` — numer nowo przydzielanego deskryptora

```
int dup(int oldfd)
int dup2(int oldfd, int newfd)
```

Zadania

Zadania

Zadanie 1

Napisz program kopiujący zawartość pliku o nazwie podanej jako pierwszy parametr do pliku, którego nazwa podana jest jako drugi parametr.

Zadanie 2

Napisz program zmieniający kolejność znaków w każdej linii pliku o nazwie podanej jako parametr.

Zadanie 3

Napisz program wyszukujący najdłuższą linię w pliku i podający ilość znaków w tej linii.

Zadania

Zadanie 1

Napisz program kopiujący zawartość pliku o nazwie podanej jako pierwszy parametr do pliku, którego nazwa podana jest jako drugi parametr.

Zadanie 2

Napisz program zmieniający kolejność znaków w każdej linii pliku o nazwie podanej jako parametr.

Zadanie 3

Napisz program wyszukujący najdłuższą linię w pliku i podający ilość znaków w tej linii.

Zadania

Zadanie 1

Napisz program kopiujący zawartość pliku o nazwie podanej jako pierwszy parametr do pliku, którego nazwa podana jest jako drugi parametr.

Zadanie 2

Napisz program zmieniający kolejność znaków w każdej linii pliku o nazwie podanej jako parametr.

Zadanie 3

Napisz program wyszukujący najdłuższą linię w pliku i podający ilość znaków w tej linii.

Zadania

Zadanie 4

Napisz program określający rozmiar pliku/kilku plików o nazwach podanych jako parametry wejściowy.

Zadanie 5

Napisz program wypisujący od końca zawartość pliku (1) znakami, (2) liniami.

Zadania

Zadanie 4

Napisz program określający rozmiar pliku/kilku plików o nazwach podanych jako parametry wejściowy.

Zadanie 5

Napisz program wypisujący od końca zawartość pliku (1) znakami, (2) liniami.