Podstawowe operacje wejścia-wyjścia dla plików		
Dominik Wróbel	07 III 2019	Czw. 17:00

Spis treści

1 Podstawy obsługi plików w systemie UNIX

1.1. Odpowiedz na pytania

Pytanie

Co to są deskryptory ?

To małe liczby całkowite, które stanowią dane powiązane z procesem, określające pliki otwarte przez proces. Deskryptory służą do realizowania operacji na plikach w wywołaniach systemowych.

Pytanie

Jakie są standardowe deskryptory otwierane dla procesów ?

Są to deskryptory o numerach 0-2, odpowiadają one standardowemu wejściu, wyjściu i wyjściu błędów.

Pytanie

Jakie flagi trzeba ustawić w funkcji open aby otrzymać funkcjonalność creat ?

O_WRONLY | O_CREAT | O_TRUNC

Pytanie

W wyniku wykonania polecenia umask otrzymano 0022. Jakie prawa dostępu będzie miał plik otwarty w następujący sposób: open(pathname, O_RDWR | O_CREAT, S_IRWXU | S_IRWXG | S_IRWXO)

Będą to prawa 0755

Pytanie

Co oznaczają flagi: O_WRONLY | O_CREAT | O_TRUNC?

O_WRONLY - otwarcie pliku tylko do zapisu, O_CREAT - flaga konieczna gdy chcemy utworzyć plik, O_TRUNC - flaga usuwająca zawartość pliku podczas otwierania go

Pytanie

Co oznacza flaga O_APPEND?

Przy każdym zapisie do pliku funkcją write() dane będą dodawane na jego koniec.

Pytanie

Co oznacza zapis: S_IRUSR | S_IWUSR?

Prawa dostępu dla usera: read oraz write.

1.2. Program copy1.c

Zadanie

Proszę rozbudować program o obsługę błędów. Proszę wyszczególnić następujące problemy:

- brak pliku
- złe prawa dostępu do pliku
- Dla tych błędów proszę wyświetlić odpowiedni komunikat
- Podpowiedź: sprawdź wartości zwracane przez funkcję oraz zmienną errno

Listing 1: copy1.c

```c
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h> /* add errno.h to get errno variable */
#include <stdlib.h>

#define BUFSIZE 512

void copy(char *from, char *to)
{
  int fromfd = -1, tofd = -1;
  ssize_t nread;
  char buf[BUFSIZE];

  fromfd = open(from, O_RDONLY);
  if(fromfd == -1){
    if(errno == ENOENT ){
      printf("fromfd - Errno is %d", errno); /* No such file or directory */
      perror("Error: ");
      exit(EXIT_FAILURE);
    }
    if(errno == EACCES){
      printf("fromfd - Errno is %d", errno); /* Permission denied */
      perror("Error: ");
      exit(EXIT_FAILURE);
    }
  }


  tofd = open(to, O_WRONLY | O_CREAT | O_TRUNC,
        S_IRUSR | S_IWUSR);
  if(tofd == -1){
    if(errno == ENOENT )
    {
      printf("to - Errno is %d", errno); /* No such file or directory */
      perror("Error: ");
      close(fromfd);
      exit(EXIT_FAILURE);
    }
    if(errno == EACCES)
    {
      printf("to - Errno is %d", errno); /* Permission denied */
      perror("Error: ");
      close(fromfd);
      close(tofd);
      exit(EXIT_FAILURE);
```

```
47      }
48    }
49
50
51
52    while ((nread = read(fromfd, buf, sizeof(buf))) > 0)
53        write(tofd, buf, nread);
54    if(nread == -1){
55      if(errno == ENOENT )
56      {
57        printf("read - Errno is %d", errno); /* No such file or directory */
58        perror("Error: ");
59        close(fromfd);
60        close(tofd);
61        exit(EXIT_FAILURE);
62      }
63      if(errno == EACCES)
64      {
65        printf("read - Errno is %d", errno); /* Permission denied */
66        perror("Error: ");
67        close(fromfd);
68        close(tofd);
69        exit(EXIT_FAILURE);
70      }
71    }
72
73      close(fromfd);
74    close(tofd);
75    return;
76  }
77
78  int main(int argc, char **argv){
79    if (argc != 3)
80    {
81      fprintf(stderr,"usage: %s from_pathname to_pathname\n", argv[0]);
82      return 1;
83    }
84    copy(argv[1], argv[2]);
85    return 0;
86  }
```

2 Operacje pisania i czytania z pliku

2.1. Odpowiedz na pytania

Pytanie

Czy w momencie powrotu z funkcji write dane są już zapisana na urządzenie wyjściowe ?

Nie, dane po wywołaniu write są przechowywane w buforze, zapisywane są dopiero w odpowiedniej dla systemu chwili.

Pytanie

Przeanalizować kod writeall.c, co robi ta funkcja ? Jakiej sytuacji dotyczy wartość EINTR ?

Funkcja *writeall(int fd, const * buf, size_t nbytes)* zapisuje ilość bajtów określoną przez nbytes z bufora buf do pliku o deskryptorze fd. EINTR dotyczy przerwania wykonywania funkcji, np. przez inny proces.

2.2. Funkcja readall.c

Zadanie

Proszę napisać funkcję readall. Funkcja ma zwracać wartości jak read ale obsługiwać błędy analogicznie jak writeall. Proszę uwzględnić możliwość przeczytania końca pliku.

Listing 2: readall.c

```c
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>

#define BUFSIZE 512

ssize_t readall(int fd, void *buf, size_t nbyte)
{
     ssize_t nread = 0;
   ssize_t n;

     do {
    if ( (n = read(fd, &((char *)buf)[nread], nbyte - nread)) == -1 ) {
              if (errno == EINTR)
                  continue;
              else
                  return -1;
         }
     if(n == 0){ // if end of file is reached
      return 0;
     }
         nread += n;
     } while (nread < nbyte);

  return nread;

}



    int main(int argc, char **argv){

    int toRead = -1;
    char buf[BUFSIZE];

      if (argc != 2)
      {
        fprintf(stderr,"usage: %s from_pathname to_pathname\n", argv[0]);
        return 1;
      }

```

```
44          toRead = open(argv[1], O_RDONLY);
45        if(toRead == -1){
46              if(errno == ENOENT){
47                  perror("File does not exists: ");
48                  exit(EXIT_FAILURE);
49              }
50              if(errno == EACCES){
51                  perror("Permission denied: ");
52                  exit(EXIT_FAILURE);
53              }
54          }
55
56          readall(toRead, buf, BUFSIZE);
57
58      printf("%s", buf);
59
60      return 0;
61  }
```

Pytanie

Jaki znak jest czytany gdy osiągnięty zostanie koniec pliku ?

Czytany jest wtedy EOF, wówczas wartość zwracana przed read to 0.

2.3. Program realizujący funkcjonalność cat - cat.c

Zadanie

Proszę napisać program realizujący funkcjonalność polecenia cat bez opcji.

- Polecenie powinno działać bez podania pliku oraz z podaniem pliku

Listing 3: cat.c

```
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <errno.h>
5  #include <unistd.h>
6
7  #define BUFSIZE 512
8
9  ssize_t readall(int fd, void *buf, size_t nbyte)
10 {
11      ssize_t nread = 0;
12     ssize_t n;
13
14      do {
15      if ( (n = read(fd, &((char *)buf)[nread], nbyte - nread)) == -1 ) {
16              if (errno == EINTR)
17                  continue;
18              else
19                  return -1;
20          }
21      if(n == 0){
22       return 0;
23      }
24          nread += n;
```

```
25      } while (nread < nbyte);
26
27    return nread;
28
29  }
30
31
32
33      int main(int argc, char **argv){
34
35      int toRead = -1;
36      char buf[BUFSIZE];
37
38        if (argc != 2)
39        {
40          printf("Podaj tekst dla cat: ");
41        scanf("%s", buf); // standardowe wejscie
42        printf("%s", buf);
43        return 0;
44        }
45
46          toRead = open(argv[1], O_RDONLY);
47        if(toRead == -1){
48              if(errno == ENOENT){
49                  perror("File does not exists: ");
50                  exit(EXIT_FAILURE);
51              }
52              if(errno == EACCES){
53                  perror("Permission denied: ");
54                  exit(EXIT_FAILURE);
55              }
56          }
57
58          readall(toRead, buf, BUFSIZE); // plik
59
60      printf("%s", buf);
61
62      return 0;
63  }
```

3 3. Wskaźnik pliku i sygnalizator O_APPEND

3.1. Odpowiedz na pytania

Pytanie

Dwa deskryptory: fd1 i fd2 użyto do otwarcia pliku podając tę samę ścieżkę dostępu do pliku. Wskaźnik pliku ustawiony jest na początku pliku. Następnie korzystając z deskryptora fd1 wykonano operację zapisania 100b do pliku. Następnie przy użyciu deskryptora fd2 wykonano operację czytania z pliku. Pytanie: Na jakiej pozycji jest wskaźnik pliku? Jakie dane odczytano przy użyciu fd2?

Wskaźnik pliku jest na pozycji po 100b. fd2 odczyta dane, które zapisał fd1, bo dwa różne deskryptory mają niezależne offsety plików.

Pytanie

Do otwarcia pliku użyto jednego deskryptora fd3. Następnie wykonano kolejno operację pisania 100b i czytania 100b. Na jakiej pozycji jest wskaźnik pliku? Co zostało przeczytane?

Zostały przeczytane dane, które znajdują się po 100b, które zapisaliśmy, EOF jeśli plik się kończył po 100b, ponieważ fd3 ma jeden offset pliku, który przy zapisaniu jest przesuwany o 100b, a read zaczyna od tego miejsca.

Pytanie

Czy każdorazowe poprzedzenie operacji pisania ustawieniem wskaźnika pliku na końcu pliku za pomocą funkcji lseek daje taki sam rezultat jak otwarcie pliku w trybie z ustawioną flagą O_APPEND? Odpowiedź uzasadnij.

Nie, O_APPEND gwarantuje każdorazowe zapisywanie na końcu pliku, nawet w sytuacji gdy na pliku działa wiele procesów, co nie jest prawdą w przypadku wywoływanie osobno lseek i write.

Pytanie

Jak wygląda wywołanie funkcji lseek które:

- ustawia wskaźnik na zadanej pozycji
- znajduje koniec pliku
- zwraca bieżącą pozycję wskaźnika

- ustawia wskaźnik na zadanej pozycji - lseek(fd, pos, SEEK_SET);
- znajduje koniec pliku - lseek(fd, 0, SEEK_END)
- zwraca bieżącą pozycję wskaźnika - lseek(fd, 0, SEEK_CUR)

3.2. Funkcja copyappend.c

Zadanie

Napisz funkcję copy, która będzie za każdym razem dodawać zawartość na końcu pliku.

Listing 4: copyappend.c

```
void copyAppend(char *from, char *to)
    {
      int fromfd = -1, tofd = -1;
      ssize_t nread;
      char buf[BUFSIZE];

      fromfd = open(from, O_RDONLY | O_APPEND); // dodanie flagi append
      if(fromfd == -1){
            if(errno == ENOENT){
                perror("File does not exists: ");
                exit(EXIT_FAILURE);
            }
            if(errno == EACCES){
                perror("Permission denied: ");
                exit(EXIT_FAILURE);
            }
        }


        tofd = open(to, O_WRONLY | O_CREAT | O_TRUNC,
            S_IRUSR | S_IWUSR | O_APPEND); // dodanie flagi append
        if(tofd == -1){
            if(errno == ENOENT){
                perror("File does not exists: ");
                exit(EXIT_FAILURE);
            }
            if(errno == EACCES){
                perror("Permission denied: ");
                exit(EXIT_FAILURE);
            }
        }


      while ((nread = read(fromfd, buf, sizeof(buf))) > 0)
          write(tofd, buf, nread);
        if(nread == -1){
            perror("Reading failed: ");
        }

        close(fromfd);
      close(tofd);
      return;
    }
```

Pytanie

Czy zmiana wywołań lseek i read na pread jest równoważna ?

Nie do końca, ponieważ pread ignoruje offset, jest on ustawiany jawnie przez argument offset.

4 4. Buforowanie operacji I/O

4.1. Porównanie czasu działania kopiowania

Zadanie

Proszę pobrać poniższy program wykorzystujący funkcje ze standardowej biblioteki wejścia-wyjścia.
Proszę stworzyć duży plik, np. plik 1MB wypełniony zerami:
dd if=/dev/zero of=/path/to/desired/big/file count=10k

Korzystając z funkcji mierzących czas z poprzedniego laboratorium proszę napisać program testujący ile czasu zajmują funkcje kopiujące copy2 (copy1 z napisaną przez nas obsługą błędów) i copy3

Proszę napisać program, który liczy czas wykonania funkcji copy2 przy różnych rozmiarach bufora (proszę przyjąć przynajmniej następujące wartości rozmiaru bufora: 1, 512, 1024, 1100)

Listing 5: timecmp.c

```
compare: t.o timecmp.o
	gcc timecmp.o t.o -o compare -I.

t.o : t.c t.h
	gcc -c -Wall -pedantic t.c -I.

timecmp.o: timecmp.c t.h
	gcc -c -Wall -pedantic timecmp.c -I.

#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h> /* add errno.h to get errno variable */
#include <stdlib.h>
#include <t.h>

#define BUFSIZE 1100

void copy(char *from, char *to)  /* has a bug */
{
  int fromfd = -1, tofd = -1;
  ssize_t nread;
  char buf[BUFSIZE];

  fromfd = open(from, O_RDONLY);
  if(fromfd == -1){
    if(errno == ENOENT ){
      printf("fromfd - Errno is %d", errno); /* No such file or directory */
      perror("Error: ");
      exit(EXIT_FAILURE);
    }
    if(errno == EACCES){
      printf("fromfd - Errno is %d", errno); /* Permission denied */
      perror("Error: ");
      exit(EXIT_FAILURE);
    }
  }

```

```
39
40    tofd = open(to, O_WRONLY | O_CREAT | O_TRUNC,
41          S_IRUSR | S_IWUSR);
42    if(tofd == -1){
43      if(errno == ENOENT )
44      {
45        printf("to - Errno is %d", errno); /* No such file or directory */
46        perror("Error: ");
47        close(fromfd);
48        exit(EXIT_FAILURE);
49      }
50      if(errno == EACCES)
51      {
52        printf("to - Errno is %d", errno); /* Permission denied */
53        perror("Error: ");
54        close(fromfd);
55        close(tofd);
56        exit(EXIT_FAILURE);
57      }
58    }
59
60
61
62    while ((nread = read(fromfd, buf, sizeof(buf))) > 0)
63        write(tofd, buf, nread);
64
65    if(nread == -1){
66      if(errno == ENOENT )
67      {
68        printf("read - Errno is %d", errno); /* No such file or directory */
69        perror("Error: ");
70        close(fromfd);
71        close(tofd);
72        exit(EXIT_FAILURE);
73      }
74      if(errno == EACCES)
75      {
76        printf("read - Errno is %d", errno); /* Permission denied */
77        perror("Error: ");
78        close(fromfd);
79        close(tofd);
80        exit(EXIT_FAILURE);
81      }
82    }
83
84      close(fromfd);
85    close(tofd);
86    return;
87  }
88
89
90  void copy3(char *from, char *to)
91  {
92    FILE *stfrom, *stto;
93    int c;
94
95    if ((stfrom = fopen(from, "r") ) == NULL){}
96    if(( stto = fopen(to, "w") ) == NULL) {}
97    while ((c = getc(stfrom)) != EOF)
98      putc(c, stto);
99    fclose(stfrom);
100   fclose(stto);
101   return;
```

```
102
103 }
104
105
106
107 int main(int argc, char **argv){
108   if (argc != 3)
109   {
110     fprintf(stderr,"usage: %s from_pathname to_pathname\n", argv[0]);
111     return 1;
112   }
113
114   timestart();
115
116   copy(argv[1], argv[2]);
117
118   timestop("copy2");
119
120
121   timestart();
122
123   copy3(argv[1], argv[2]);
124
125   timestop("copy3");
126
127
128   return 0;
129 }
```

Wyniki uzyskane z eksperymentu:

Listing 6: Logi z programu timecmp.c

```
1
2 26 lines (21 sloc) 433 Bytes
3 512:
4 copy2:
5   "Total (user/sys/real)", 0, 5, 12
6   "Child (user/sys)", 0, 0
7 copy3:
8   "Total (user/sys/real)", 50, 1, 78
9   "Child (user/sys)", 0, 0
10
11
12 1024:
13 copy2:
14   "Total (user/sys/real)", 0, 3, 8
15   "Child (user/sys)", 0, 0
16 copy3:
17   "Total (user/sys/real)", 50, 1, 74
18   "Child (user/sys)", 0, 0
19
20
21 1100:
22 copy2:
23   "Total (user/sys/real)", 0, 3, 8
24   "Child (user/sys)", 0, 0
25 copy3:
26   "Total (user/sys/real)", 53, 0, 78
27 "Child (user/sys)", 0, 0
```

Czas dla 1100 jest zgodnie z przypuszczeniami większy niż czas osiągany dla 512 czy 1024.