Zaawansowane operacje wejścia-wyjścia dla plików		
Dominik Wróbel	14 III 2019	Czw. 17:00

Spis treści

1 Pozyskiwanie i wyświetlanie metadanych pliku

1.1. Odpowiedz na pytania

Pytanie

Proszę przejrzeć manual do funkcji z rodziny stat(2). Czym różnią się te funkcje ?

- Funkcja stat przyjmuje ścieżkę jako argument i szuka i-node podążając tą ścieżką.
- Funkcja lstat jest identyczna jak stat, ale w przypadku gdy ścieżka prowadzi do linku symbolicznego, wyświetla metadane powiązane z tym linkiem, a nie z tym do czego on wskazuje, jak w przypadku stat.
- Funkcja fstat przyjmuje deskryptor, a nie ścieżkę do pliku, znajduje i-node w tablicy aktywnych i-node w jądrze systemu

Pytanie

Co reprezentuje flaga S_IFMT zdefiniowana dla pola st_mode ?

Jest to makro reprezentujące maskę dzięki której można sprawdzić typ pliku. Aby przy pomocy tego makra uzyskać typ pliku należy wykonać logiczny AND z polem st_mode struktury stat.

Pytanie

Zmienna sb jest wypełnioną strukturą typu struct stat. Czy można sprawdzić typ pliku (np. czy plik jest urządzeniem blokowym) w następujący sposób ? Odpowiedź uzasadnij.

```
if (( sb.st_mode & S_IFBLK) == S_IFBLK) { /* plik jest urzadzeniem blokowym */}
```

Nie, nie można sprawdzić typu pliku w taki sposób. Najpierw należy wykonać logiczny AND z maską pliku w celu uzyskania typu pliku, a następnie przyrównać wynik do odpowiedniej flagi. Prawidłowa wersja to:

```
if (( sb.st_mode & S_IFMT) == S_IFBLK) { /* plik jest urzadzeniem blokowym */}
```

1.2. Program stat_info.c

Zadanie

Proszę pobrać, skompilować i uruchomić poniższy program. stat_info.c

Zadanie

4. Proszę zmodyfikować funkcję print_type tak aby badała wszystkie możliwe typy pliku i wyświetlała odpowiednią informację.
5. Proszę zmodyfikować funkcję print_perms tak aby wyświetlała prawa dostępu do pliku w postaci numerycznej, np. 755.
6. Proszę zmodyfikować funkcję print_owner tak aby wyświetlała nazwę właściciela i grupy właścicieli oraz podawała identyfikatory w nawiasach, np. wta(1234) iastaff(5678)
7. Proszę zmodyfikować funkcję print_perms tak aby wyświetlone zostały prawa dostępu do pliku użytkownika uruchamiającego program np. Your permisions: read: yes, write: no, execute: no.
8. Proszę zmodyfikować funkcję print_size tak aby wyświetlała informacje o rozmiarze pliku w sformatowany sposób - w kilo/megabajtach itp., np. 1MB zamiast 1024bytes
9. Proszę zmodyfikować funkcje print_laststch, print_lastacc i print_lastmod tak aby wyświetlały czas, który minął od daty ostatniej zmiany statusu/dostępu/modyfikacji, np. 3 day ago
10. Proszę zmodyfikować funkcje print_name tak, aby w przypadku gdy podany jako argument plik jest linkiem symbolicznym wyświetlał jego nazwę w formacie: nazwa_linku nazwa_plik_na_ktory_wskazuje.

Listing 1: Zadania 4-10

```
static void print_file_info(struct stat *sb, char *name){

    // 4.
      printf("File type:                ");
      switch (sb->st_mode & S_IFMT) {
      case S_IFBLK:  printf("block device\n");            break;
      case S_IFCHR:  printf("character device\n");        break;
      case S_IFDIR:  printf("directory\n");               break;
      case S_IFIFO:  printf("named or unnamed pipe\n");   break;
      case S_IFLNK:  printf("symbolic link\n");           break;
      case S_IFREG:  printf("regular file\n");            break;
      case S_IFSOCK: printf("Socket\n");                  break;
      default:       printf("unknown?\n");                break;
      }


      // 5. i 7.
      char *perms = (char *) malloc(sizeof(char) * 128);
    strcat(perms, "Your permissions: read: ");

    int usr = 0;
    int grp = 0;
    int oth = 0;
      printf("Mode:                     %lo (octal)\n", (unsigned long) sb->st_mode);
    if( (sb->st_mode & S_IRUSR) == S_IRUSR){
      usr += 4;
    strcat(perms, "yes, write: ");
    }
    else{
      strcat(perms, "no, write: ");
    }
    if( (sb->st_mode & S_IWUSR) == S_IWUSR){
      usr += 2;
    strcat(perms, "yes, execute: ");
    }
    else{
      strcat(perms, "no, execute: ");
    }
    if( (sb->st_mode & S_IXUSR) == S_IXUSR){
```

```
40        usr += 1;
41      strcat(perms, "yes");
42      }
43      else {
44        strcat(perms, "no");
45      }
46      if( (sb->st_mode & S_IRGRP) == S_IRGRP){
47        grp += 4;
48      }
49      if( (sb->st_mode & S_IWGRP) == S_IWGRP){
50        grp += 2;
51      }
52      if( (sb->st_mode & S_IXGRP) == S_IXGRP){
53        grp += 1;
54      }
55      if( (sb->st_mode & S_IROTH) == S_IROTH){
56        oth += 4;
57      }
58      if( (sb->st_mode & S_IWOTH) == S_IWOTH){
59        oth += 2;
60      }
61      if( (sb->st_mode & S_IXOTH) == S_IXOTH){
62        oth += 1;
63      }
64      printf("Permission: %d %d %d \n", usr, grp, oth);
65      printf("%s\n", perms);
66
67
68      // 6.
69      char *userName =(char *)malloc(64*sizeof(char));
70      char *groupName =(char *)malloc(64*sizeof(char));
71
72      long uid = (long) sb->st_uid;
73      long gid = (long) sb->st_gid;
74      struct passwd *pwd = getpwuid(uid);
75      struct group *grp = getgrgid(gid);
76
77      userName = pwd->pw_name;
78      groupName = grp->gr_name;
79
80      printf("Ownership:                UID=%ld   GID=%ld\n", (long) sb->st_uid, (long) sb
        ->st_gid);
81      printf("Ownership:                %s(%ld) %s(%ld)\n", userName, (long) sb->st_uid,
        groupName, (long) sb->st_gid );
82
83
84      // 7. rozwiazanie w 5.
85
86
87      // 8.
88      printf("Preferred I/O block size: %ld bytes\n", (long) sb->st_blksize);
89
90      printf("File size:                %lld bytes\n",(long long) sb->st_size);
91
92        printf("Kb File size:             %f kb\n",(float)sb->st_size/1024);
93
94      printf("MB File size:            %f Mb\n",(float)sb->st_size/(1024*1024));
95
96        printf("Blocks allocated:           %lld\n",(long long) sb->st_blocks);
97
98        // 9.
99      printf("Last status change:       %s", ctime(&sb->st_ctime));
100     time_t currentTime;
```

```
101       time(&currentTime);
102       long long diffTime;
103     diffTime = difftime(currentTime,sb->st_ctime);
104 
105     printf("Last status change:         %lld days ago\n",diffTime/(60*60*24) );
106 
107     printf("Last file access:         %s", ctime(&sb->st_atime));
108 
109     time_t currentTime;
110       time(&currentTime);
111       long long diffTime;
112     diffTime = difftime(currentTime,sb->st_atime);
113 
114     printf("Last file access:         %lld days ago\n",diffTime/(60*60*24) );
115 
116     printf("Last file modification:   %s", ctime(&sb->st_mtime));
117 
118     time_t currentTime;
119       time(&currentTime);
120       long long diffTime;
121     diffTime = difftime(currentTime,sb->st_mtime);
122 
123     printf("Last file modification:         %lld days ago\n",diffTime/(60*60*24) );
124 
125 
126     // 10.
127     char* bname = basename(name);
128       printf("Name of the file:          %s\n", bname);
129 
130     char* lname = (char*) malloc(256 * sizeof(char));
131     ssize_t bytesRead;
132 
133     if( (sb->st_mode & S_IFMT) == S_IFLNK ){
134     printf("Is Link");
135     if((bytesRead = readlink(name,lname, sizeof(lname)) != -1)){
136       if(lname != NULL){
137           lname[bytesRead] = '\0';
138           printf("SymLink %s -> File %s\n",lname,bname);
139       }
140       else{
141           exit(EXIT_FAILURE);
142       }
143       }
144   }
145 
146 
147 }
```

2 Wejście / wyjście asynchroniczne

Zadanie

Do programu z poprzedniego ćwiczenia dopisz funkcję print_content, której deklaracja może wyglądać następująco static void print_content(char *name). Powyższa funkcja implementuje następującą funkcjonalność:

- Pyta użytkownika czy chce wypisać zawartość podanego jako argument pliku.
- Jeżeli tak, to otwiera i przy pomocy funkcji czytania asynchronicznego aio_read(3) odczytuje zawartość pliku i wypisuje ją na ekran.

Powyższe zadanie najlepiej wykonać stosując taki oto scenariusz:

- Otwieramy plik open(2)
- Inicjalizujemy wczytywanie pierwszej porcji danych aio_read(3).
- Przy pomocy funkcji aio_error(3) czekamy do momentu aż odczyt się zakończy.
- Wyświetlamy wczytane dane i wracamy do 2 jeżeli nie osiągnęliśmy końca pliku.

Listing 2: print_content

```
static void print_content(char * name){

    char * choice = (char*) malloc(1*sizeof(char));

    printf("Would you like to print content of your file ?\n");
    printf("Press Y - Yes, any key otherwise");

    int fd;
    struct aiocb cb;
    char check[256];

    scanf("%c", choice);
    if(*choice == 'y' || *choice == 'Y'){

      fd = open(name, O_RDONLY); // open file
      memset(&cb, 0, sizeof(struct aiocb));
      cb.aio_fildes = fd; // configure struct
      cb.aio_buf = check;
      cb.aio_nbytes = 256;
      printf("reading aio");
      int n;
      while(1){
        n = aio_read(&cb); // read
        if(n==-1){
          printf("AIO Error"); // handle errors
          exit(EXIT_FAILURE);
        }
        int err;
        printf("reading aio 2");
        while((err = aio_error (&cb)) == EINPROGRESS); // wait for asynchronous read to
    complete
        printf("%s\n", check);
        if(n==0){
          break;
        }
```

```
35          }
36
37      } else {
38          printf("Printing cancelled");
39      }
40
41      return;
42
43  }
```