

## システムソフトウェア 第2回課題

学生番号 20B30790 藤井 一喜

2022 年 12 月 8 日

# 1 ファイルシステム API 強化

## 1.1 open への O\_APPEND フラグの追加

### 実装

kernel/fcntl.h に O\_APPEND フラグを追加した。また kernel/sysfile.c に

#### ソースコード 1 kernel/fcntl.h

```
1 if((omode & O_APPEND) && ip->type == T_FILE){  
2     // seek to end of file  
3     f->off = ip->size;  
4 }
```

を追加した。これにより、ファイルをオープンする際に O\_APPEND フラグが指定されている場合、オフセットがファイルの末尾に設定されるようになった。

次に、リダイレクション >> を追加するために user/sh.c を以下のように変更した。

#### ソースコード 2 user/sh.c

```
1 case '+': // >>  
2     cmd = redircmd(cmd, q, eq, O_WRONLY|O_CREATE|O_APPEND, 1);  
3     break;
```

### テスト

以下のように、テストプログラムを作成し、make qemu 後に usertests を実行することで、O\_APPEND フラグが正しく機能していることを確認した。また、>> (リダイレクション) に関しては、make qemu 後に、あらかじめ作成しておいた test.txt に対して、実際に echo hello >> test.txt を実行し、ファイルの末尾に文字列が追加されていることを確認した。

#### ソースコード 3 user/usertests.c

```
1 // test O_APPEND.  
2 void  
3 append1(char *s)  
4 {  
5     char buf[32];  
6  
7     unlink("appendfile");  
8     int fd1 = open("appendfile", O_CREATE|O_WRONLY|O_TRUNC);  
9     write(fd1, "abcd", 4);
```

```

10  close(fd1);
11
12  int fd2 = open("appendfile", O_RDONLY);
13  int n = read(fd2, buf, sizeof(buf));
14  if(n != 4){
15      printf("%s: read %d bytes, wanted 4\n", s, n);
16      printf("file=%s\n", buf);
17      exit(1);
18  }
19  close(fd2);
20
21  fd1 = open("appendfile", O_WRONLY|O_APPEND);
22  write(fd1, "efgh", 4);
23  close(fd1);
24
25  fd2 = open("appendfile", O_RDONLY);
26  n = read(fd2, buf, sizeof(buf));
27  if(n != 8){
28      printf("%s: read %d bytes, wanted 8\n", s, n);
29      printf("file=%s\n", buf);
30      exit(1);
31  }
32  close(fd2);
33
34  fd1 = open("appendfile", O_WRONLY|O_APPEND);
35  write(fd1, "ijklmnop", 8);
36  close(fd1);
37
38  fd2 = open("appendfile", O_RDONLY);
39  n = read(fd2, buf, sizeof(buf));
40  if(n != 16){
41      printf("%s: read %d bytes, wanted 16\n", s, n);
42      printf("file=%s\n", buf);
43      exit(1);
44  }
45  close(fd2);
46
47  unlink("appendfile");
48 }
49
50 ... (abbreviated) ...
51

```

```
52 { "append1", append1 },
```

テストコードの内容としては、O\_APPEND フラグが正しく機能しているかどうかを確認するために、文字列をファイル末尾に追加する動作を繰り返し、ファイルの内容が正しいかどうかを確認するというものである。

usertests を実行した結果、問題なく PASS した。

## 1.2 システムコール lseek の追加

### 実装

system call を追加するために、user/usys.pl に entry("lseek"); を追加し、user/user.h に int lseek(int, int, int); を追加、kernel/syscall.h に以下を追加した。

#### ソースコード 4 kernel/syscall.h

```
1 #define SYS_lseek 23
```

また、kernel/syscall.c に

#### ソースコード 5 kernel/syscall.c

```
1 extern uint64 sys_lseek(void);
2 ...
3 [SYS_lseek] sys_lseek,
```

を追加、kernel/fcntl.h に以下を追加した。

#### ソースコード 6 kernel/fcntl.h

```
1 #define SEEK_SET 0
2 #define SEEK_CUR 1
3 #define SEEK_END 2
```

また、kernel/sysfile.c に以下を追加した。

#### ソースコード 7 kernel/sysfile.c

```
1 uint64
2 sys_lseek(void)
3 {
4     int fd, offset, whence;
5
6     struct file *f; // file pointer
7     struct proc *p = myproc(); // current process
8
9     argint(0, &fd);
```

```

10     argint(1, &offset);
11     argint(2, &whence);
12
13     // fd < 0 : invalid file descriptor
14     // fd >= NOFILE : invalid file descriptor
15     // p->ofile[fd] == 0 : file is not opened
16     if(fd < 0 || fd >= NOFILE || (f = p->ofile[fd]) == 0)
17         return -1;
18
19     // 操作対象が通常のファイルの場合
20     if(f->type == FD_INODE){
21         ilock(f->ip);
22
23         // 更新前のoffsetを保存
24         uint old_offset = f->off;
25
26         if(whence == SEEK_SET) {
27             f->off = offset; // offset from the beginning of the file (offset の位置)
28         } else if(whence == SEEK_CUR) {
29             f->off += offset; // offset from the current position (現在の
30                             // offset に offset を加えた値)
31         } else if(whence == SEEK_END) {
32             f->off = f->ip->size + offset; // offset from the end of the file (ファイルサ
33                             // イズにoffset を加えた値)
34         } else {
35             iunlock(f->ip);
36             return -1;
37         }
38
39         // 計算された新しいオフセットがファイルサイズより大きい場合
40         // new_offset > file_size
41         if(f->off > f->ip->size) {
42             // file size を更新
43             f->ip->size = f->off;
44         }
45
46         // new_offset < 0 OR new_offset > file_size limit
47         if(f->off < 0 || f->off > MAXFILE * BSIZE) {
48             // offset を更新前の値に戻す
49             f->off = old_offset;
50             iunlock(f->ip);
51             return -1;

```

```

50     }
51
52     iunlock(f->ip);
53     return f->off;
54 }
55 return -1;
56 }

```

---

方針としては、引数として `fd`, `offset`, `whence` を受け取った後、`type` が `FD_INODE` の場合に、`f->off` を変更することで、`lseek` を実現することにした。

オフセットを変更する際は、`ilock` により `inode` をロックする処理を加えた。

また、問題文にあるように `whence` には、`SEEK_SET`、`SEEK_CUR`、`SEEK_END` のいずれかが入ることになるので、それぞれの場合に対応するように場合分けを施し、新しいオフセットを計算するようにした。

また求められている仕様に従い、新しいオフセットがファイルサイズより大きい場合は、ファイルサイズを更新するようにした。

実装全体は、私の個人リポジトリにある。<https://github.com/okoge-kaz/xv6-riscv>

## テスト

以下のようなテストコードを追加した。

---

### ソースコード 8 user/usertests.c

---

```

1  // test lseek.
2  void
3  lseek1(char *s){
4
5      unlink("seekfile");
6      int fd1 = open("seekfile", O_CREATE|O_RDWR|O_APPEND);
7      if (fd1 < 0) {
8          printf("open_failed\n");
9          exit(1);
10     }
11     write(fd1, "hello", 6);
12
13     int off1 = lseek(fd1, 0, SEEK_SET);// オフセットをファイルの先頭に戻す
14     // off1 = 0
15     if (off1 != 0) {
16         printf("lseek_failed\n");
17         printf("off1=%d, expected_0\n", off1);
18         exit(1);
19     }

```

```

20
21 char buf[6];
22 int n = read(fd1, buf, 6);
23 if (n != 6) {
24     printf("read_failed\n");
25     printf("buf=%s, n=%d, expected=%d\n", buf, n, 6);
26     exit(1);
27 }
28
29 int off2 = lseek(fd1, 200, SEEK_SET);// オフセットをファイルの先頭から 200バイト目に
    セットする
30 // off2 = 200
31 if (off2 != 200) {
32     printf("lseek_failed\n");
33     printf("off2=%d, expected_200\n", off2);
34     exit(1);
35 }
36
37 int off3 = lseek(fd1, -200, SEEK_CUR);// オフセットを現在の位置から 200バイト分戻す
38 // off3 = 0
39 if (off3 != 0) {
40     printf("lseek_failed\n");
41     printf("off3=%d, expected_0\n", off3);
42     exit(1);
43 }
44
45 int off4 = lseek(fd1, 0, SEEK_CUR);// オフセットを変更しない。現在のオフセットを得る
46 // off3 == off4
47 if (off3 != off4) {
48     printf("lseek_failed\n");
49     printf("off3=%d, off4=%d, expected_off3==off4\n", off3, off4);
50     exit(1);
51 }
52
53 int off5 = lseek(fd1, 0, SEEK_END);// オフセットをファイルの末尾まで進める。返値はフ
    ァイルの大きさ
54 // off5 = 200
55 if (off5 != 200) {
56     printf("lseek_failed\n");
57     printf("off5=%d, expected_200\n", off5);
58     exit(1);
59 }

```

```

60
61     int off6 = lseek(fd1, 100, SEEK_END); // ファイル末尾に 100バイト追加する
62     // off6 = 300
63     if (off6 != 300) {
64         printf("lseek_failed\n");
65         printf("off6=%d, expected_300\n", off6);
66         exit(1);
67     }
68
69     int off7 = lseek(fd1, -100, SEEK_CUR); // オフセットを 100バイト戻す
70     // off7 = 200
71     if (off7 != 200) {
72         printf("lseek_failed\n");
73         printf("off7=%d, expected_200\n", off7);
74         exit(1);
75     }
76
77     unlink("seekfile");
78 }
79
80 void
81 lseek2(char * s){
82     unlink("seekfile");
83     int fd1 = open("seekfile", O_CREATE|O_WRONLY);
84     if (fd1 < 0) {
85         printf("open_failed\n");
86         exit(1);
87     }
88     write(fd1, "Goodbye!\n", 11);
89     close(fd1);
90
91     fd1 = open("seekfile", O_WRONLY);
92     if (fd1 < 0) {
93         printf("open_failed\n");
94         exit(1);
95     }
96
97     lseek(fd1, 5, SEEK_SET);
98     write(fd1, "night!\n", 7);
99     close(fd1);
100
101     fd1 = open("seekfile", O_RDONLY);

```



```

102     if (fd1 < 0) {
103         printf("open_failed\n");
104         exit(1);
105     }
106
107     char buf[20];
108     int n = read(fd1, buf, 20);
109     if (n != 12) {
110         printf("read_failed\n");
111         printf("buf=%s, n=%d, expected=%d\n", buf, n, 12);
112         exit(1);
113     }
114
115     if (strcmp(buf, "Good_night!\n") != 0) {
116         printf("read_failed\n");
117         printf("buf=%s, expected=%s\n", buf, "Good_night!\n");
118         exit(1);
119     }
120
121     unlink("seekfile");
122 }
123
124 void
125 lseek3(char * s){
126     unlink("seekfile");
127     int fd1 = open("seekfile", O_CREATE|O_WRONLY);
128     if (fd1 < 0) {
129         printf("open_failed\n");
130         exit(1);
131     }
132     write(fd1, "Hello_World", 12);
133     close(fd1);
134
135     fd1 = open("seekfile", O_RDWR);
136     int offset = lseek(fd1, 200, SEEK_SET);// オフセットをファイルの先頭から 200バイト目
        にセットする
137     if (offset != 200) {
138         printf("lseek_failed\n");
139         printf("offset=%d, expected_200\n", offset);
140         exit(1);
141     }
142

```

```

143     int file_size = lseek(fd1, 0, SEEK_END);// オフセットをファイルの末尾まで進める。返値
        はファイルの大きさ
144     if (file_size != 200) {
145         printf("lseek_1failed\n");
146         printf("file_size=%d,_expected_200\n", file_size);
147         exit(1);
148     }
149
150     int error = lseek(fd1, -250, SEEK_CUR);// オフセットを現在の位置から 250バイト分戻す
        (オフセットが負になる)
151     if (error != -1) {// error
152         printf("lseek_1failed\n");
153         printf("error=%d,_expected_-1\n", error);
154     }
155
156     // error が起きた場合は、オフセットは変化しない
157     int offset2 = lseek(fd1, 0, SEEK_CUR);
158     if (offset2 != 200) {
159         printf("lseek_1failed\n");
160         printf("offset2=%d,_expected_200\n", offset2);
161         exit(1);
162     }
163
164     close(fd1);
165     unlink("seekfile");
166 }

```

---

lseek1 のテストについては、ファイルディスクリプタのオフセットを移動させ、想定した通りの挙動であるかどうかを確認している。この際、SEEK\_SET, SEEK\_CUR, SEEK\_END の挙動すべてを確認している

また、lseek2 では、ファイル書き込みの際に、オフセットを移動させて書き込みを行うことで、想定する文字列が得られるかどうかを確認することを通じて、lseek の挙動を確認している。

lseek3 では、仕様として求められている要件が充足されているかを確かめるために、新しいオフセットが負になるような lseek の呼び出しや、ファイルサイズよりも大きなオフセットを指定するような lseek の呼び出しを行い、検証を行っている。

この実装についても、個人リポジトリにて全体を公開している。(https://github.com/okoge-kaz/xv6-riscv)

### 1.3 得られた知見

ファイルディスクリプタのオフセット値を変更することで、ファイルの読み書きを行う位置を変更できることが実感を伴って理解できた。

また、ファイルホールという概念についても理解を深めることができた。

### 1.4 環境

講義にて用意されている Docker Image を使用している。(docker-compose を用いて `docker compose up` にて起動できるようにして使用した。)

- PC: MacStudio 2021 M1 Max
- Memory: 64 GB
- OS: macOS 13.0.1

### 1.5 参考文献

UNIX/Linux プログラミング教室 富永和人, 権堂克彦

Linux プログラミングインターフェース Michael Kerrisk