Операции над файлами в Linux

Наумов Д.А., доц. каф. КТ

Операционные системы и системное программное обеспечение, 2019

Содержание лекции

1. Операции над файлами
   - Удаление файла
   - Перемещение файла
   - Создание ссылок
   - Создание и удаление каталогов

Удаление файла: unlink()

Для удаления файла служит системный вызов unlink(), объявленный в заголовочном файле unistd.h следующим образом:

```
int unlink (const char * FNAME);
```

- Аргумент FNAME — это имя удаляемого файла.
- unlink() возвращает 0 при успешном завершении.
- В случае ошибки возвращается –1.

Удаление файла: unlink()

```
#include <stdio.h>
#include <unistd.h>

int main (int argc, char ** argv)
{
  if (argc < 2) return 1;

  if (unlink (argv[1]) == -1) {
    fprintf (stderr, "Cannot unlink file (%s)", argv[1]);
    return 1;
  }

  return 0;
}
```

Файл

комплексное понятие, состоящее из следующих компонентов:

- данные (data);
- индексы (inodes);
- ссылки (links).

Индексы

специальные ячейки памяти, зарезервированные файловой системой для разделения данных на файлы.

- Каждый индекс имеет уникальный (в рамках данной файловой системы) номер.
- Индексы содержат информацию о том, в каких блоках файловой системы хранятся данные конкретного файла.
- В индексах содержатся сведения о дате и времени открытия и модификации файла.
- Сылка - имя индексного узла файловой системы.

Индексы

Программа ls, вызванная с флагом -i, позволяет увидеть номер индексного узла, на который указывает ссылка:

```
$ mkdir idemo
$ cd idemo
$ touch file1
$ touch file2
$ ls -i

952139 file1
952141 file2
```

- file1 — это ссылка на индекс с номером 952139
- file2 указывает на другой индексный узел с номером 952141

Индексы

Продолжим:

```
$ ln -s file1 symlnkf1
$ ls -i

952139 file1
952141 file2
952190 symlnkf1
```

- символическая ссылка является также ссылкой на индекс с номером 952190
- символические ссылки указывают не на индекс, а на имя файла

Индексы

Теперь воспользуемся командой ln без флага -s:

```
$ ln file1 hardfile1
$ ls -i

952139 file1 952141 file2 952139 hardfile1 952190 symlnkf1
```

- hardfile1 является жесткой ссылкой на файл file1. Вывод команды ls показывает, что file1 и hardfile1 указывают на один и тот же индекс с номером 952139.
- жесткие ссылки (в отличие от символических) не носят подчиненный характер. file1 и hardfile1 — это полноценные ссылки на один и тот же индексный узел.

Индексы

Если две ссылки указывают на один и тот же индекс, то можно сказать, что они имеют доступ к одним и тем же данным:

```
$ echo hello > file1
$ cat hardfile1
hello
```

Индексы являются промежуточным звеном, связывающим ссылку с данными на блочном устройстве.

Команда rm

Программа rm работает следующим образом:

- если удаляемый файл является последней ссылкой на соответствующий индексный узел в файловой системе, то данные и индекс освобождаются;
- если в файловой системе еще остались ссылки на соответствующий индексный узел, то удаляется только ссылка.

```
$ rm file1
$ cat hardfile1
hello
```

Теперь обратите внимание на вывод программы ls с флагом -l:

```
$ ls -l
-rw-r--r-- 1 kt kt 0 2011-05-07 10:00 file2
-rw-r--r-- 1 kt kt 6 2011-05-07 10:00 hardfile1
lrwxrwxrwx 1 kt kt 5 2011-05-07 10:00 symlnkf1 -> file1
```

Команда rm

Символическая ссылка symlnkf1 по-прежнему указывает на файл file1, которого уже не существует:

```
$ cat symlnkf1
cat: symlnkf1: No such file or directory
```

Числа во втором столбце вывода программы ls - счетчики ссылок на соответствующие индексные узлы.

```
$ ls -l
-rw-r--r-- 2 kt kt 0 2011-05-07 10:00 file2
-rw-r--r-- 1 kt kt 6 2011-05-07 10:00 hardfile1
-rw-r--r-- 2 kt kt 0 2011-05-07 10:00 hardfile2
lrwxrwxrwx 1 kt kt 5 2011-05-07 10:00 symlnkf1 -> file1
```

Команда df

Если вызвать команду df с флагом -i, то на экран будет выведена информация по индексным узлам смонтированных файловых систем:

```
$ df -i

Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/sda6 1311552 264492 1047060 21% /
udev 96028 487 95541 1% /dev
/dev/sda1 66264 58 66206 1% /boot
/dev/sda7 3407872 149600 3258272 5% /home
```

В файловых системах может присутствовать ограниченное число индексов (столбец Inodes). Столбец IUsed показывает число используемых индексов, а в столбце IFree содержится число свободных индексных узлов файловой системы

Команда df

Каждый раз при создании файла в файловой системе выделяется индекс:

```
$ df -i .
Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/sda7 3407872 149586 3258286 5% /home

$ touch file3
$ df -i .

Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/sda7 3407872 149587 3258285 5% /home
```

Команда df

Создание жесткой ссылки не приводит к появлению в файловой системе нового индекса:

```
$ df -i .
Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/sda7 3407872 149587 3258285 5% /home

$ ln file3 hardfile3
$ df -i .
Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/sda7 3407872 149587 3258285 5% /home
```

Пример statvfsinode.c

В рассматриваемом ранее примере программа использовала функцию statvfs() для вывода информации о смонтированных файловых системах.
Структура statvfs содержит также следующие поля, которые мы не рассматривали:

- f_files — общее число индексов для данной файловой системы;
- f_free — число свободных индексов файловой системы;
- f_favail — число доступных индексов в файловой системе.

Пример statinode.c

Структура stat содержит еще одно поле st_ino, в котором находится номер индексного узла, на который ссылается файл.

```
if (stat (argv[1], &st) == -1) {
        fprintf (stderr, "stat() error\n");
        return 1;
}

printf ("FILE:\t\t%s\n", argv[1]);
printf ("UID:\t\t%d\n", (int) st.st_uid);
printf ("GID:\t\t%d\n", (int) st.st_gid);
printf ("SIZE:\t\t%ld\n", (long int) st.st_size);
printf ("AT:\t\t%s", ctime (&st.st_atime));
printf ("MT:\t\t%s", ctime (&st.st_mtime));

printf ("INODE:\t\t%ld\n", (long int) st.st_ino);

return 0;
```

Системный вызов unlink() (unlink2.c) удаляет ссылку на индексный узел. Если эта ссылка была последней, то индекс освобождается.

```
$ touch file1
$ ln file1 file2
$ ls -i file1 file2
1048740 file1 1048740 file2
$ ./unlink2 file1
$ ls file1
/bin/ls: file1: No such file or directory
$ cat file2
Hello World
```

Над открытым файлом можно успешно осуществлять операции ввода-вывода, даже если последняя ссылка на этот файл удалена:

```
$ ./unlink2 file2
$ ls file2
/bin/ls: file2: No such file or directory
```

Перемещение файлов

Системный вызов rename() позволяет переименовывать или перемещать файл в пределах одной файловой системы.

```
int rename (const char * OLDF, const char * NEWF);
```

- При успешном завершении rename() возвращает 0.
- В случае ошибки возвращается –1

Пример (rename1.c):

```
if (rename (argv[1], argv[2]) == -1) {
        fprintf (stderr, "rename() error\n");
        return 1;
}
```

Перемещение файлов: rename2.c

Теперь проведем небольшой эксперимент:

```
$ touch file1
$ cat file1
$ ./rename2 file1 file2
$ ls file1
/bin/ls: file1: No such file or directory
$ cat file2
Hello World
```

Перемещение открытого файла никак не отражается на операциях ввода-вывода.

Создание ссылок

Ссылки в файловой системе Linux бывают двух типов:

- символические ссылки (symbolic links);
- жесткие (прямые) ссылки (hard links).

В распоряжении программиста имеются следующие системные вызовы:

```
int link (const char * FROM, const char * TO);
int symlink (const char * FROM, const char * TO);
```

Оба вызова возвращают 0 при успешном завершении и –1, если произошла ошибка. Примеры:

- linkdemo.c
- symlinkdemo.c

Создание каталога

Для создания каталога используется системный вызов mkdir():

```
int mkdir (const char * NAME, mode_t MODE)
```

Системный вызов mkdir() создает каталог с именем NAME и режимом MODE. При успешном завершении mkdir() возвращает 0. В случае ошибки возвращается –1. Примеры:

- mkdir1.c
- mkdir2.c

Создание каталога

Программа mkdir работает не так, как мы ожидали::

```
$ ./mkdir2 mydir
$ ls -l | grep mydir
drwxr-xr-x 2 kt kt 4096 2011-05-07 10:09 mydir
```

В системный вызов mkdir() передавался аргумент mode, в котором все биты базовых прав доступа установлены в единицу. Но вывод программы ls показывает, что созданный каталог имеет права доступа 0755.

К каждому процессу в Linux привязана маска прав доступа, которая наследуется потомком от родительского процесса (аналогично текущему каталогу, окружению и т. п.).

Маска прав доступа

число, представляющее собой набор битов прав доступа, которые никогда не будут устанавливаться для создаваемых процессом файлов или каталогов.

Команда umask позволяет узнать текущую маску прав доступа командной оболочки:

```
$ umask
0022
```

Маска прав доступа 0022 разрешает при создании файлов или каталогов устанавливать любые права доступа для владельца, но не разрешает права на запись для группы и остальных пользователей.

Маски прав доступа

Текущий процесс вправе изменять свою копию маски прав доступа:

```
$ umask 0044
$ umask
0044
```

Дочерние процессы наследуют копию маски прав доступа родительского процесса:

```
$ umask 000
$ umask
0000
$ bash
$ umask
0000
$ exit
exit
```

Создание каталога

Попробуем запустить программу mkdir2 с измененной маской прав доступа оболочки:

```
$ umask 0000
$ umask
0000
$ ./mkdir2 mydir
$ ls -l | grep mydir
drwxrwxrwx 2 kt kt 4096 2011-05-07 10:15 mydir
```

Системный вызов umask()

Программа может изменить маску прав доступа текущего процесса при помощи системного вызова umask():

```
mode_t umask (mode_t MASK);
```

Этот системный вызов изменяет текущую маску прав доступа и возвращает предыдущее значение маски. Примеры:

- mkdir3.c

```
$ umask 0022
$ umask
0022
$ ./mkdir3 mydir
$ ls -l | grep mydir
drwxrwxrwx 2 kt kt 4096 2011-05-07 10:17 mydir
```

Создание каталога

Пример создания каталога с "липким битом":

- mkdir4.c

```
$ ./mkdir4 mydir
$ ls -l | grep mydir
drwxrwxrwt 2 kt kt 4096 2011-05-07 10:20 mydir
```

Удаление каталога

Для удаления каталога служит системный вызов rmdir():

```
int rmdir (const char * DIR)
```

- Аргумент DIR — это имя (путь) к каталогу, который следует удалить.
- При успешном завершении rmdir() возвращает 0. В случае ошибки возвращается –1.

Пример:

- rmdirdemo.c

Системный вызов rmdir() удаляет только пустые каталоги. Если каталог не пуст, то rmdir() завершится неудачей.