ОП.14 **Основы функционирования UNIX - систем**

ЗАНЯТИЕ 07

Файловая система
Операции с дисковыми файлами

В этом разделе мы рассмотрим практические аспекты работы с объектами файловой системы, такими как дисковые файлы и каталоги.

Именно операции с файлами и каталогами, располо женными на жестком диске, в подавляющем большинстве случаев приходится выполнять пользователю.

Как известно, к объектам файловой системы относятся также и устройства ввода/вывода, но анализ выполнения операций с такими объектами требует специальных знаний, поэтому рассматривать эти вопросы здесь мы не будем.

Файловая система
Операции с дисковыми файлами

Операции над файлами можно выполнять двумя вариантами:

- с использованием команд UNIX;
- при разработке программ на языках высокого уровня, посредством системных вызовов или библиотечных функций.

Следует отметить, что все команды UNIX для манипуляций файлами реализованы с использованием системных вызо вов.

Файловая система
Операции с дисковыми файлами

Системные вызовы для работы с объектами файловой системы сгруппированы в таблице.

Функция API	Назначение
`open()`	Открывает файл для доступа к данным
`read()`	Чтение данных из файла
`write()`	Запись данных в файл
`close()`	Закрытие дескриптора открытого файла

Файловая система
Операции с дисковыми файлами

Системные вызовы. Таблица (продолжение).

Функция API	Назначение
`lseek()`	Позиционирование в открытом файле
`stat()`, `fstat()`	Получение атрибутов файла
`chmod()`	Изменение прав доступа к файлу

Файловая система
Операции с дисковыми файлами

Системные вызовы. Таблица (продолжение).

Функция API	Назначение
`chown()`	Изменение идентификатора пользователя (uid) или группы (gid) для файла
`utime()`	Изменение даты и времени последнего изменения содержимого файла и последнего доступа к нему

Файловая система
Операции с дисковыми файлами

Системные вызовы. Таблица (окончание).

Функция API	Назначение
`link()`	Создание жесткой ссылки на файл
`unlink()`	Удаление жесткой ссылки на файл
`umask()`	Установка маски

Файловая система
Операции с дисковыми файлами

Операционная система UNIX позволяет выполнять различные манипуляции над объектами файловой системы, включая:

- создание и удаление файлов;
- копирование, перемещение и создание ссылок на объекты файловой системы;
- чтение/запись данных;
- установку и изменение атрибутов файлов.

Файловая система
Операции с дисковыми файлами

Некоторые из этих операций, например, установка и изменение атрибутов файлов, а также чтение и запись данных, мы рассматривали ранее.

Сейчас же сосредоточим внимание на операциях:

- создания/удаления;
- копирования/перемещения дисковых файлов и каталогов.

Начнем с копирования файлов.

Файловая система
Операции с дисковыми файлами
Копирование файлов

Для копирования файлов используется команда `cp`. Она позволяет копировать файлы или каталоги, допуская копирование одного файла в другой, а также копирование группы файлов в заданный каталог.

Синтаксис команды `cp` можно представить следующим образом:

```
cp [опции]  файл  путь
cp [опции]  файл... каталог
```

Файловая система
Операции с дисковыми файлами
Копирование файлов

Если в качестве последнего параметра `cp` задан существующий каталог, то выполняется копирование исходных файлов в этот каталог с сохранением их имен.

В том случае, если параметрами являются имена файлов, `cp` копирует первый файл во второй.

Если командная строка содержит более двух параметров, не являющихся опциями самой команды, а последний параметр не является именем какого-либо каталога, то команда генерирует ошибку

Файловая система
Операции с дисковыми файлами
Копирование файлов

Попытка скопировать файл сам в себя ни к чему не приводит, кроме того, что выдается сообщение об ошибке.

Права доступа к скопированным файлам и каталогам вычисляются путем логического умножения (операция И) кода доступа исходных файлов на 0777, а также с учетом маски, установленной для пользователя.

Файловая система
Операции с дисковыми файлами
Копирование файлов

Вот некоторые примеры использования команды `cp`.

Для копирования одного каталога в другой можно выполнить команду:

```
# cp -r  DIR  DIR.OLD
```

Здесь каталог `DIR` вместе со своим содержимым копируется в каталог `DIR.OLD`.

Файловая система
Операции с дисковыми файлами
Копирование файлов

В следующем примере команда

```
# cp -r  DIR1  DIR2  DIR12
```

копирует содержимое каталогов `DIR1` и `DIR2` в каталог `DIR12`.

Операцию копирования несложно реализовать в приложении, написанном на одном из языков высокого уровня.

Исходный текст простейшего аналога UNIX-команды `cp`, написанный на языке C («Си»), представлен в листинге.

Копирование файлов с использованием функций API

```
iinclude <stdio.h>
iinclude <unistd.h>
iinclude <stdlib.h>
iinclude <sys/types.h>
iinclude <fcntl.h>

int main(int argc, char* argv[])
{
 if (argc != 3)
  {
   printf("Usage: %s [source] [dest]\n", argv[0]);
   exit(0);
  }

  char buf[1024];
  int bytes;
  int fsrc = open(argv[1], 0_RDONLY);
  if (fsrc == -1)
   {
    printf("Can't open %s\n", argv[1]);
    exit(1);
   }
```

Копирование файлов с использованием функций API

```
int fdst = open(argv[2], O_RDWR | O_CREAT, 7 64);
if  (fdst = -1)
 {
   printf("Can't create %s\n", argv[2]);
   exit(2);
 }
while (1)
{
  bytes = read(fsrc, buf, sizeof(buf));
   if  (bytes == 0)
     break;
   write (fdst, buf, bytes);
 }

 close (fsrc);
 close(fdst);
 return 0;
}
```

Файловая система
Операции с дисковыми файлами
Копирование файлов

Как видно из исходного текста программы, в ней используются уже знакомые нам системные вызовы UNIX `open()`, `read()`,`write()` и `close()`.

Программа принимает два параметра: первый параметр `argv[1]` указывает имя исходного файла, а второй `argv[2]` — имя файла назначения.

Файловая система
Операции с дисковыми файлами
Копирование файлов

Исходный файл открывается при помощи системного вызова `open()` для чтения, после чего все операции чтения выполняются посредством дескриптора `fsrc`.

Считанные из дескриптора `fsrc` данные (оператор `bytes = read (fsrc, buf, sizeof (buf));`) помещаются во временный буфер памяти `buf`, после чего записываются в дескриптор `fdst` вновь создаваемого файла функцией `write()`:

```
write (fdst, buf, bytes);
```

Файловая система
Операции с дисковыми файлами
Копирование файлов

Операция копирования завершается, если при чтении в буфер количество прочитанных байтов `bytes` становится равным 0/

При этом происходит выход из цикла `while` и закрытие дескрипторов файлов функцией `close()`.

Файловая система
Операции с дисковыми файлами
Перемещение файлов

Перемещение файлов в операционной системе UNIX выполняется с помощью команды `mv`, имеющей синтаксис:

`mv [`*опции*`...]  `*исходный_файл*`  `*файл_назначения*

`mv [`*опции*`...]  `*исходный_файл*`...  `*каталог*

Файловая система
Операции с дисковыми файлами
Перемещение файлов

Если последний параметр команды указывает на имя существующего каталога, то `mv` перемещает указанные файлы в этот каталог.

В том случае, если в качестве параметров заданы имена двух файлов, то имя первого файла будет изменено на имя второго.

Если же последний параметр не является каталогом, и за даны имена более чем двух файлов, то команда генерирует ошибку.

Файловая система
Операции с дисковыми файлами
Перемещение файлов

Когда *исходный_файл* и *файл_назначения* находятся в одной файловой системе, то изменяется имя файла, а владелец, права доступа, атрибуты времени остаются неизменными.

Если же они находятся в разных файловых системах, то *исходный_файл* копируется и затем удаляется.

Во время выполнения операции команда `mv` пытается скопировать время последней модификации, время доступа, идентификаторы пользователя и группы и права доступа к файлу.

Файловая система
Операции с дисковыми файлами
Перемещение файлов

Вот пример использования команды `mv`:

```
# mv test test.old
```

Здесь файл `test` переименовывается в файл `test.old`.

Программный аналог UNIX-команды `mv` несложно реализовать при помощи системных вызовов `link()` и `unlink()`, как это показано в листинге.

Перемещение файлов с использованием функций API

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

int main(int argc, char* argv[])
{
 if (argc != 3)
  {
    printf("Usage: %s [source] [dest]\n", argv[0]);
    exit(0);
  }

 if (link(argv[1], argv[2]) == 0)
   if (unlink(argv[1]) == -1)
      printf("Can't delete file %s\n", argv[1]);
 return 0;
}
```

Файловая система
Операции с дисковыми файлами
Перемещение файлов

В этой программе с помощью системного вызова `link()` создается дополнительная жесткая ссылка (параметр `argv[2]`) на исходный файл, имя которого указано первым параметром `argv[1]`.

Если эта операция выполнена успешно, то первая же сткая ссылка (она же является именем исходного файла) удаляется функцией `unlink()`.

Файловая система
Операции с дисковыми файлами
Удаление файлов и каталогов

Для удаления файлов и каталогов в операционной системе UNIX используются команды `rm` и `rmdir`.

С помощью команды `rmdir` можно удалить одиночный каталог, причем он **должен быть пустым**.

Если в каталоге имеются элементы, отличные от . и .. , то команда `rmdir` такой каталог не удаляет.

Файловая система
Операции с дисковыми файлами
Удаление файлов и каталогов

Синтаксис этой команды таков:

`rmdir [-p][-s]` *`каталог`*

Команда `rmdir` имеет две опции:

- `-p` — позволяет удалить пустой каталог вместе с его родительскими каталогами, отображая сообщение об успешном или неуспешном выполнении операции;
- `-s` — подавляет выдачу сообщений при использовании опции `-p`.

Файловая система
Операции с дисковыми файлами
Удаление файлов и каталогов

Команда `rm` функционирует иначе — с ее помощью можно удалить указанные файлы, но каталоги по умолчанию не удаляются.

При указании опций `-r` или `-R` будет удаляться все дерево каталогов нижезаданного каталога, включая и сам каталог, причем на глубину дерева не накладывается никаких ограничений.

Если последний компонент файла — символ . или .. , то генерируется ошибка (это помогает избежать неприятных сюрпризов при выполнении команды `rm -r . *` или ей подобных).

Файловая система
Операции с дисковыми файлами
Удаление файлов и каталогов

Общие для UNIX-систем опции POSIX данной команды имеют следующий смысл:

- `-f` — не запрашивается подтверждение операции и не выдаются диагностические сообщения. При завершении команды с ошибками код ошибки не возвращается, если ошибки вызваны отсутствием файлов;
- `-i` — выводится запрос на подтверждение удаления (при указании опций `-f` и `-i` одновременно используется последняя);
- `-r` или `-R` — позволяет рекурсивно удалять дерево каталогов (с описанием).

Файловая система
Операции с дисковыми файлами
Удаление файлов и каталогов

Команда `rm` довольно ***опасна!!!***

После ее выполнения **восстановить** удаленные файлы **невозможно**.

Поэтому нужно быть очень **внимательным** при ее использовании.

Файловая система
Операции с дисковыми файлами
Создание каталогов

Создать новый каталог в UNIX можно с помощью команды `mkdir`.

В простейшем варианте команда использует один пара метр (имя каталога), создавая каталог с указанным именем.

С помощью одной команды `mkdir` можно создавать несколько каталогов одновременно, перечисляя их в одной командной строке.

Синтаксис команды таков:

`mkdir [`*`опции`*`] [`*`список_каталогов`*`]`

Файловая система
Операции с дисковыми файлами
Создание каталогов

В команде `mkdir` можно использовать две опции:

- `-m` — позволяет задать в восьмеричной или символьной форме права доступа (как и для команды `chmod`), которые будут присвоены создаваемым каталогам;
- `-p` — кроме указанного каталога создаются любые требуемые промежуточные каталоги.

Если у пользователя нет прав на запись в родительский каталог, то новый каталог не создается, а если каталог уже существует (или файл с таким же име нем), то команда генерирует ошибку.

Файловая система
Поиск файлов и каталогов

Кроме **копирования/перемещения** и **создания/удаления** к часто выполняемым операциям с файлами и каталогами относится и ***поиск объектов файловой системы***.

Файловая система UNIX содержит десятки тысяч файлов, поэтому для быстрого поиска используются очень эффективные средства, одним из которых является команда `find`.

Команда имеет синтаксис:

`find` *`каталог`* `...` *`выражение`*

Файловая система
Поиск файлов и каталогов

Она просматривает иерархии каталогов в поисках файлов, удовлетворяющих критерию, задаваемому выражением выражение. Выражения строятся из элементов с помощью следующих конструкций:

- `-name` *`шаблон`* — условие истинно, если имя файла соответствует шаблону. При использовании метасимволов необходимо маскировать шаблоны от командного интерпретатора;

Файловая система
Поиск файлов и каталогов

- `-type` *`тип`* — условие истинно, если файл — указанного типа. Типы файлов задаются символами `b`, `c`, `d`, `f`, `l`, `p` и `s`, обозначающими, соответственно, специальное блочное устройство, специальное символьное устройство, каталог, обычный файл, символическую ссылку, именованный канал и сокет;
- `-user` *`пользователь`* — условие истинно, если файл принадлежит пользователю, указанному по идентификатору или регистрационному имени;

Файловая система
Поиск файлов и каталогов

- `-group` *`группа`* — условие истинно, если файл принадлежит группе, указанной по идентификатору или имени;
- `-perm [-]` *`права`* — если дефис не задан, то условие истинно, только если права доступа в точности соответствуют указанным (как в команде `chmod`).

Если задан дефис, то условие истинно, если в правах доступа файла, как минимум, установлены те же биты, что и в указанных правах;

Файловая система
Поиск файлов и каталогов

- `-size [+|-|=] n [c]` — условие истинно, если файл имеет длину *n* блоков (блок — 512 байтов) или символов (если указан суффикс `c`).

Перед размером можно указывать префикс `+` (не меньше), `-` (не больше) или `=` (в точности равен);

- `-atime [+|-|=]n` — условие истинно, если к файлу последний раз обращались *n* дней назад.

Перед *n* в элементах `-atime`, `-ctime` и `-mtime` можно указывать префикс `+` (не позже), `-` (не ранее) или `=` (ровно);

Файловая система
Поиск файлов и каталогов

- `-ctime` *n* — условие истинно, если файл создан *n* дней назад;
- `-mtime` *n* — условие истинно, если файл был изменен *n* дней назад;
- `-newer файл` — условие истинно, если файл более новый, чем указанный;
- `-ls` — условие истинно всегда (выдает информацию о файле, аналогичную длинному листингу);

Файловая система
Поиск файлов и каталогов

- `-print` — условие истинно всегда (выдает полное имя файла в стандартный выходной поток);
- `-exec` *`команда`* `{} \;` — условие истинно, если выполненная команда имеет код возврата 0.

Команда заканчивается замаскированной точкой с запятой.

В команде можно использовать конструкцию `{}`, заменяемую полным именем рассматриваемого файла;

Файловая система
Поиск файлов и каталогов

- `-ok` *`команда`* `{} \;` — аналогично `exec`, но полученная после подстановки имени файла вместо `{}` команда выдается с вопросительным знаком и выполняется, если пользователь ввел символ y;
- `-depth` — условие истинно всегда — требует так обходить иерархию каталогов, чтобы файлы любого каталога всегда обрабатывались раньше, чем сам каталог (обход "в глубину");

Файловая система
Поиск файлов и каталогов

- `-prune` — условие истинно всегда — требует не проверять файлы в каталоге, путевое имя которого присутствует в предыдущем выражении.

Не действует, если ранее указан элемент `-depth`.

В различных версиях операционной системы UNIX могут поддерживаться и другие компоненты выражений в команде `find`.

Если командная строка сформирована неправильно, команда немедленно завершает работу.

Файловая система
Поиск файлов и каталогов

Вот несколько примеров использования команды `find`:

Пример 1.

Для отображения списка файлов текущего каталога программы достаточно выполнить команду:

```
# find . -print
```

Пример 2.

Для получения содержимого произвольного каталога, например, `/home/developer` нужно выполнить команду:

```
# find /home/developer -print
```

Файловая система
Поиск файлов и каталогов

Пример 3.

Для поиска файлов в текущем каталоге с именами, которые заканчиваются на `tmp`, нужно выполнить команду:

```
# find . -name '*tmp' -print
```

Пример 4.

Здесь с помощью команды `find` выполняется поиск файлов с расширением `tmp` или `c`, находящихся в текущем каталоге:

```
# find . \( -name '*.tmp' -o -name '*.c' \) -print
```

Файловая система
Поиск файлов и каталогов

В команде `find` можно задавать временные критерии поиска файлов, причем в самых различных комбинациях.

Следующий пример демонстрирует это: в нем используется опция `-atime [+|-|=] n`.

Условие является истинным, если время последнего доступа к файлу **больше/меньше**, чем `n*24`.

Например, команда `# find . \( -name '*.tmp' -o -name '*.pl' \) -atime +3 -print`

выполняет поиск файлов с указанными шаблонами, к которым не было обращения больше трех суток.

Файловая система
Поиск файлов и каталогов

Например, команда:

```
# find . \( -name '*.tmp' -o -name '*.pl' \) -atime +3 -print
```

выполняет поиск файлов с указанными шаблонами, к которым не было обращения больше трех суток.

Нередко требуется найти файлы, принадлежащие определенному пользователю.

Например, следующая команда выполняет поиск файлов в каталоге `/usr`, владельцем которых является супер-пользователь `root`:

```
# find /usr -user root -print
```

Файловая система
Поиск файлов и каталогов

Если критерием поиска является размер файла, то можно использовать следующую опцию: `-size [+|-|=]n[c]`.

Условие, задаваемое этой опцией, истинно, если размер файла **больше/меньше** *n*.

При этом различают два случая: если присутствует опция `c`, то размер файла предполагается заданным в байтах, если опция `c` отсутствует — то в блоках по 512 байтов.

Следующая команда выполняет поиск файлов, размер которых превышает 2048 байтов, в каталоге `/developer`:

```
$ find /developer -size +2048c -print
```

Файловая система
Поиск файлов и каталогов

Команда `find` может выполнять другие команды или группы команд, принимающих в качестве параметра результат поиска файлов.

Для реализации такой возможности служит опция `-exec`.

В этом случае команда должна заканчиваться пробелом и символами `\;`.

В следующем примере из каталога `/developer` удаляются все файлы, размер которых не превышает 1000 байтов:

```
# find /developer -size -1000c -print -exec rm {} \;
```

Файловая система
Поиск файлов и каталогов

Для вывода на консоль атрибутов файлов (команда `ls -l`), удовлетворяющих шаблону `t*`, можно воспользоваться командой

```
# find /developer -name 't*' -exec ls -l {} \;
```

Расширить возможности команды `find` можно, перенаправив ее вывод не на стандартное устройство вывода, а в программный канал, как это показано в следующем примере:

```
# find TMP -name 't*' -print | grep tmp
```

Файловая система
Поиск файлов и каталогов

Здесь команда `find` выполняет поиск файлов в каталоге `TMP`, удовлетворяющих шаблону `t*`, в имени которых присутствует `tmp`.

Конвейер программ чаще всего применяется в операциях копирования, перемещения и создания резервных копий файловых систем — при этом вывод команды `find` служит вводом для команды архивирования, как правило, `cpio`.

Файловая система
Поиск файлов и каталогов

В следующем примере выполняется копирование файлов в другой каталог.

Для этого применяется команда `cpio -p`, которая принимает из стандартного входного потока список файлов и копирует или создает на них ссылки (опция `-l`) в каталоге `NEW` (к моменту выполнения копирования он должен существовать).

Опция `-d` требует создания каталогов при необходимости.

Опция `-m` запрещает модификацию времени изменения файла.

Файловая система
Поиск файлов и каталогов

Для генерации списка полных путевых имен файлов для `cpio` в команде `find` нужно задать опцию `-depth`.

Это позволяет создавать файлы в каталогах, доступных **только для чтения**.

Вот так выглядит командная строка для выполнения операции копирования:

```
# find . -depth -print | cpio -pdlmv NEW
```

Файловая система
Поиск файлов и каталогов

Заканчивая обзор возможностей операционной системы UNIX для работы с файлами, хочется добавить, что **дополнительную информацию** по данной теме можно почерпнуть:

- из **man**-страниц;
- или из многочисленных источников в **Интернете**.

Список литературы:

1. Юрий Магда. UNIX для студентов, Санкт-Петербург «БХВ-Петербург», 2007.
2. Unix и Linux: руководство системного администратора, 4-е издание, 2012, Э. Немет, Г. Снайдер, Т. Хейн, Б. Уэйли
3. Организация UNIX систем и ОС Solaris 9, Торчинский Ф.И., Ильин Е.С., 2-е издание, исправленное, 2016.

Спасибо за внимание!

Преподаватель: Солодухин Андрей Геннадьевич

Электронная почта: asoloduhin@kait20.ru