

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 ~]$ cd test
[user@centos8 test]$ touch file
[user@centos8 test]$ ls
file
[user@centos8 test]$
```

Чтобы научиться работать с файлами, нам понадобятся тестовые файлы. Их мы можем создать с помощью команды `touch filename`. Вообще, основная задача `touch` – обновить время доступа к файлу, поэтому и такое название – потрогать. Если файл существует, применив `touch` на него ничего с ним не случится, только обновится информация в иноде, а если файла не существует – он создастся. Это может быть полезно в каких-нибудь скриптах, но и сейчас это помогает нам создать тестовые файлы.

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ touch file1 file2 file3
[user@centos8 test]$ ls
file file1 file2 file3
[user@centos8 test]$ touch file{5..15}
[user@centos8 test]$ ls
file      file11  file14  file3   file7
file1     file12  file15  file5   file8
file10    file13  file2   file6   file9
[user@centos8 test]$ mkdir dir_{a..f}
[user@centos8 test]$ ls
dir_a  dir_d  file      file11  file14  file3   file7
dir_b  dir_e  file1     file12  file15  file5   file8
dir_c  dir_f  file10    file13  file2   file6   file9
[user@centos8 test]$
```

С помощью `touch` можно разом создать несколько файлов – `touch file1 file2 file3`. Одна из фишек командной строки – возможность работать с регулярными выражениями. Например, мы можем написать `touch file{5..15}` и баш преобразит это в `touch file5 file6 file7` и т.д., в итоге мы получим кучу файлов. Тоже самое работает и с буквами, допустим, тот же `mkdir dir_{a..f}`. Но о регулярных выражениях мы еще поговорим.

```
user@centos8:~/test
File Edit View Search Terminal Help

[user@centos8 test]$ touch .file20
[user@centos8 test]$ mkdir .dir_h
[user@centos8 test]$ ls
dir_a  dir_d  file   file11  file14  file3  file7
dir_b  dir_e  file1  file12  file15  file5  file8
dir_c  dir_f  file10  file13  file2   file6  file9
[user@centos8 test]$ ls -a
.      dir_d  file1  file14  file5
..     dir_e  file10  file15  file6
dir_a  dir_f  file11  file2   file7
dir_b  .dir_h  file12  .file20  file8
dir_c  file   file13  file3   file9
[user@centos8 test]$ ls -a ~
.      .local
..     .mozilla
a      Music
.bash_history  .nanorc
.bash_logout   new
.bash_profile  Pictures
.bashrc        .pki
.cache         Public
```

Также можно создавать невидимые файлы и директории. Для этого в начале имени файла или директории следует указывать точку. Например, `touch .file20` или `mkdir .dir_h`. Если посмотреть на вывод `ls`, скрытых файлов и директорий не увидать. Чтобы их увидеть, для `ls` нужен ключ `-a`. `ls -a`. Тогда мы увидим все файлы в директории. Для того, чтобы увидеть скрытые файлы в файловом менеджере, можно нажать `Ctrl + h`. Тоже самое, чтобы скрыть. Обычно скрытые файлы нужны всяким программам для хранения настроек в домашней директории пользователя, но не только. В целом это позволяет скрыть ненужные пользователю файлы и директории.

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ cp file newfile
[user@centos8 test]$ ls
dir_a  dir_e  file10  file14  file5  file9
dir_b  dir_f  file11  file15  file6  newfile
dir_c  file   file12  file2   file7
dir_d  file1  file13  file3   file8
[user@centos8 test]$ cp -v file1 newfile1
'file1' -> 'newfile1'
[user@centos8 test]$
```

И так, что можно делать с файлами? Начнём с копирования. Для этого используется команда `cp` – `copy`. Чтобы скопировать файл в текущую директорию, используем `cp` имя нужного файла и имя для копии – `cp file newfile`. Команда что-то сделала, а чтобы увидеть, что произошло, нужно посмотреть (`ls`). Многие команды сами могут показывать, что происходит во время работы. Обычно для этого используется опция `verbose` (подробно) - вы часто будете наткаться на эту опцию. Обычно она пишется как `-v`. То есть, `cp -v file1 newfile1`. Тут мы видим, что файл скопировался.

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ cp -v file dir_a/newfile
'file' -> 'dir_a/newfile'
[user@centos8 test]$ cp -v file dir_a
'file' -> 'dir_a/file'
[user@centos8 test]$ cp -v dir_a/file dir_b/fileb
'dir_a/file' -> 'dir_b/fileb'
[user@centos8 test]$ cp -v dir_a/file dir_b/
'dir_a/file' -> 'dir_b/file'
[user@centos8 test]$ cp -v dir_b/fileb .
'dir_b/fileb' -> './fileb'
[user@centos8 test]$ cp -v dir_b/fileb filea
'dir_b/fileb' -> 'filea'
[user@centos8 test]$ cp dir_b/fileb
cp: missing destination file operand after 'dir_b/f
ileb'
Try 'cp --help' for more information.
[user@centos8 test]$ cp -v /home/user/test/file8 ~/
test/dir_c
'/home/user/test/file8' -> '/home/user/test/dir_c/f
ile8'
[user@centos8 test]$
```

Чтобы скопировать файл в директорию, используем `cp -v file dir_a/newfile`. Но вообще, если копировать в другую директорию, то необязательно указывать новое имя, можно просто написать `cp -v file dir_a`. Можно копировать из одной директории в другую – `cp -v dir_a/file dir_b/fileb`, либо также не указывая нового имени – `cp -v dir_a/file dir_b`. Ещё можно копировать из другой директории в текущую, используя ссылку для текущей директории – точку, как я говорил в прошлый раз. То есть, `cp -v dir_b/fileb .`, если вы не хотите менять имя, либо, если вы хотите новое имя, то достаточно просто указать его – `cp -v dir_b/fileb filea`. В общем, идея какая – для текущей директории путь не нужен, но нельзя просто написать `cp dir_b/fileb`, потому что синтаксис требует указать, куда файл нужно копировать. Если это не сделать, команда подумает, что недописана и просто выдаст ошибку. Ну и до этого мы использовали относительные пути, а так мы можем работать и с полными путями, например `cp -v /home/user/test/file8 ~/temp/dir_c`.

```
[user@centos8 test]$ cp -v .file20 dir_c
'.file20' -> 'dir_c/.file20'
[user@centos8 test]$ cp -v file6 file7 file8 dir_d
'file6' -> 'dir_d/file6'
[user@centos8 test]$ cp -v dir_a/file dir_b/fileb dir_d/file8 dir_e
'dir_a/file' -> 'dir_e/file'
'dir_b/fileb' -> 'dir_e/fileb'
'dir_d/file8' -> 'dir_e/file8'
[user@centos8 test]$ cp -v file{10..13} dir_e
'file10' -> 'dir_e/file10'
'file11' -> 'dir_e/file11'
'file12' -> 'dir_e/file12'
'file13' -> 'dir_e/file13'
[user@centos8 test]$ cp -v dir_d/* dir_f
'dir_d/file6' -> 'dir_f/file6'
'dir_d/file7' -> 'dir_f/file7'
'dir_d/file8' -> 'dir_f/file8'
[user@centos8 test]$ cp -v file1* dir_e
'file1' -> 'dir_e/file1'
'file10' -> 'dir_e/file10'
'file11' -> 'dir_e/file11'
'file12' -> 'dir_e/file12'
'file13' -> 'dir_e/file13'
'file14' -> 'dir_e/file14'
```

Копирование скрытых файлов ничем не отличается от копирования обычных файлов – просто не забывайте указывать точку в начале названия файла, потому что это часть названия файла. `cp -v .file20 dir_c`. Можно копировать несколько файлов разом, но тогда нужно копировать их в другую директорию. `cp -v file6 file7 file8 dir_d`. Ну и по той же логике, указывая пути – `cp -v dir_a/file dir_b/fileb dir_d/file8 dir_e`. Помните про регулярные выражения? Можем использовать и здесь. Допустим, `cp -v file{10..13} dir_e`. Кроме фигурных скобок, можно использовать и звёздочку – её часто называют wildcard – она означает, что подойдёт любое значение. Допустим, в директории `dir_d` есть три файла и я хочу скопировать все три. Пишу `cp -v dir_d/* dir_f`. Тогда все файлы копируются. Или, допустим, я хочу скопировать все файлы, названия которых начинаются с `file1`, а сюда подходят файл `file1`, `file10`, `file11`, `file12` и т.д. Я пишу `cp -v file1* dir_e` и все подходящие файлы копируются.

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ cp -i file* dir_d/
cp: overwrite 'dir_d/file6'? y
cp: overwrite 'dir_d/file7'? yes
cp: overwrite 'dir_d/file8'? y
[user@centos8 test]$
```

Давайте рассмотрим ещё пару ключей. Допустим, ключ `-i` – интерактивно. Что это значит? Когда вы копируете файл, если существует файл с таким названием в директории назначения, этот файл просто перезапишется. Это не всегда нас устраивает, иногда мы хотим лично решать, что перезаписывать, а что нет. Поэтому здесь нам поможет ключ `-i`. Допустим, скопируем все файлы в директорию `dir_d`. `cp -i file* dir_d`. Командная строка начнёт спрашивать, а что же делать с файлами, которые существуют там с тем же именем. Мы можем отказаться от перезаписывания файла, либо согласиться. Обычно когда какая-то команда спрашивает вопрос, на который нужно ответить да/нет, то подходят ответы `y`, либо `yes`, `n`, либо `no`. Такой вариант – интерактивный - требует, чтобы вы лично сидели и решали, что делать с каждым совпадающим файлом.

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ cp -v -i file newfile
cp: overwrite 'newfile'? no
[user@centos8 test]$ cp -vi file newfile
cp: overwrite 'newfile'? n
[user@centos8 test]$ cp -iv file newfile
cp: overwrite 'newfile'? y
'file' -> 'newfile'
[user@centos8 test]$
```

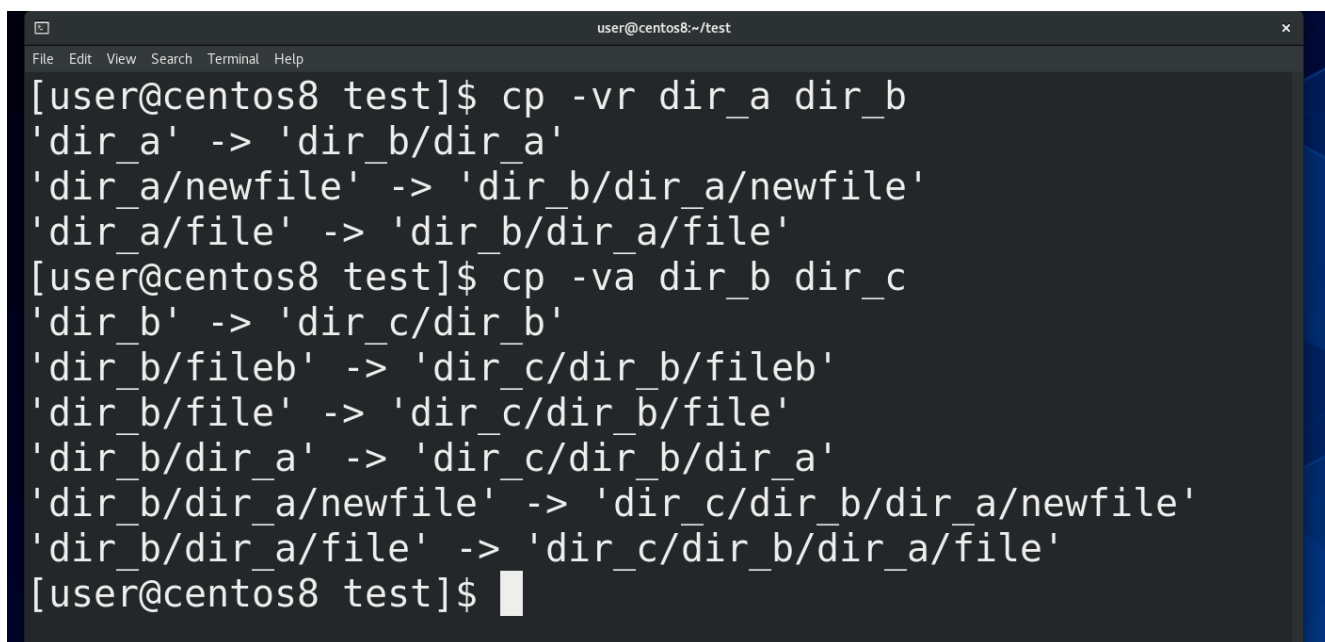
Кстати, ключи можно использовать вместе. Допустим, мы знаем ключ `-v` и ключ `-i`. Мы можем использовать их вместе – либо `cp -v -i file newfile`, либо `cp -iv file newfile`, либо `cp -vi file newfile`. Порядок – какой ключ вначале, какой потом – почти никогда не имеет значения.

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ cp -vn file* dir_d
[user@centos8 test]$ cp -vu file1* dir_e
'file1' -> 'dir_e/file1'
'file10' -> 'dir_e/file10'
'file11' -> 'dir_e/file11'
'file12' -> 'dir_e/file12'
'file13' -> 'dir_e/file13'
'file14' -> 'dir_e/file14'
'file15' -> 'dir_e/file15'
[user@centos8 test]$
```

Другой вариант – мы не хотим перезаписывать файлы – тогда сразу используем опцию -n. Например, `cp -vn file* dir_d`. Тогда файлы в директории не перезапишутся. А если, допустим, мы хотим пропустить совпадающие файлы, а скопировать только файлы новее или отсутствующие файлы – тогда опция -u – update. То есть, `cp -vu file1* dir_e`.

```
user@centos8:~/test
File Edit View Search Terminal Help
[user@centos8 test]$ cp -vl file filelink
'file' -> 'filelink'
[user@centos8 test]$
```

Помните про жёсткие ссылки? Мы можем вместо копирования создать жёсткую ссылку. Например, `cp -vl file filelink`.

A terminal window titled 'user@centos8:~/test' with a menu bar (File, Edit, View, Search, Terminal, Help). It shows two commands and their outputs. The first command is 'cp -vr dir_a dir_b', which copies 'dir_a' into 'dir_b'. The output shows 'dir_a' being copied to 'dir_b/dir_a', and its sub-items 'newfile' and 'file' being copied to 'dir_b/dir_a/newfile' and 'dir_b/dir_a/file' respectively. The second command is 'cp -va dir_b dir_c', which copies 'dir_b' into 'dir_c'. The output shows 'dir_b' being copied to 'dir_c/dir_b', and its sub-items 'fileb', 'file', 'dir_a', 'dir_a/newfile', and 'dir_a/file' being copied to 'dir_c/dir_b/fileb', 'dir_c/dir_b/file', 'dir_c/dir_b/dir_a', 'dir_c/dir_b/dir_a/newfile', and 'dir_c/dir_b/dir_a/file' respectively. The prompt '[user@centos8 test]\$' is shown at the end of the output.

```
[user@centos8 test]$ cp -vr dir_a dir_b
'dir_a' -> 'dir_b/dir_a'
'dir_a/newfile' -> 'dir_b/dir_a/newfile'
'dir_a/file' -> 'dir_b/dir_a/file'
[user@centos8 test]$ cp -va dir_b dir_c
'dir_b' -> 'dir_c/dir_b'
'dir_b/fileb' -> 'dir_c/dir_b/fileb'
'dir_b/file' -> 'dir_c/dir_b/file'
'dir_b/dir_a' -> 'dir_c/dir_b/dir_a'
'dir_b/dir_a/newfile' -> 'dir_c/dir_b/dir_a/newfile'
'dir_b/dir_a/file' -> 'dir_c/dir_b/dir_a/file'
[user@centos8 test]$
```

Также мы можем копировать директории. Для этого используем опцию `-r` – рекурсивно. Допустим, я хочу скопировать директорию `dir_a` в директорию `dir_b` - `cp -vr dir_a dir_b`. Но эту опцию можно использовать не только с директориями, но и с файлами, никому от этого хуже не станет. Забегая вперёд, скажу, что в большинстве случаев правильнее копировать с опцией `-a` вместо `-r`, по сути `-a` это опции `-r` и `-d`. Вкратце, это позволяет сохранить права и владельца файла у копии файла.

Как-то получилось, что я очень много времени уделил на копирование и лучше пока не перегружать вас информацией о других командах. Но то что мы сделали сегодня с одной командой, актуально и для большинства других команд – во многом ключи и подходы похожи.