howto, find, locate, bash, shell

How to find stuff

- https://stackoverflow.com/questions/6844785/how-to-use-regex-with-find-command

recursively find specific string in filename of files with specific suffix

- https://stackoverflow.com/questions/25693638/recursively-find-files-with-a-specific-extension

Case sensitive:

```
find . -name '*.jpg' -o -name '*.png' -print | grep Robert
```

Case insensitive:

```
find . -iname '*.jpg' -o -iname '*.png' -print | grep Robert
```

Using find's -regex argument:

```
find . -regex '.*/Robert\.\(h\|cpp\)$'
```

Or just using -name:

```
find . -name 'Robert.*' -a \( -name '*.cpp' -o -name '*.h' \)
```

The -o repreents an OR condition and you can add as many as you wish within the braces. So this says to find all files containing the word “Robert” anywhere in their names and whose names end in either “pdf” or “jpg”.

```
find -name "*Robert*" \( -name "*.pdf" -o -name "*.jpg" \)
```

As an alternative to using -regex option on find, since the question is labeled bash, you can use the brace expansion mechanism:

```
eval find . -false "-o -name Robert".{jpg,pdf}
```

This q/a shows how to use find with regular expression: How to use regex with find command? Pattern could be something like:

```
'^Robert\\.\\(h|cgg\\)$'
```

As a script you can use:

```
find "${2:-.}" -iregex ".*${1:-Robert}\.\(h\|cpp\)$" -print
```

- save it as findcc
- chmod 755 findcc

and use it as

```
findcc [name] [[search_direcory]]
```

e.g.

```
findcc                # default name 'Robert' and directory .
```

```
findcc Joe             # default directory '.'
findcc Joe /somewhere  # no defaults
```

note you cant use

```
findcc /some/where    #eg without the name...
```

also as alternative, you can use

```
find "$1" -print | grep "$@"
```

and

```
findcc directory grep_options
```

like

```
findcc . -P '/Robert\.(h|cpp)$'
```

Using bash globbing (if find is not a must)

```
ls Robert.{pdf,jpg}
```

Recurisvely with ls: (-al for include hidden folders)

```
ftype="jpg"
ls -1R *.${ftype}  2> /dev/null
```

For finding the files in system using the files database:

```
locate -e --regex "\.(h|cpp)$"
```

recursively find files with a specific suffix and copy them

- https://stackoverflow.com/questions/15617016/copy-all-files-with-a-certain-extension-from-all-subdirectories
- https://stackoverflow.com/questions/17334014/move-files-to-directories-based-on-extension

Under unix, I want to copy all files with a certain extension (all excel files) from all subdirectories to another directory. I have the following command:

```
cp --parents `find -name \*.xls*` /target_directory/
```

The problems with this command are:

- It copies the directory structure as well, and I only want the files (so all files should end up in /target_directory/)
- It does not copy files with spaces in the filenames (which are quite a few)

Any solutions for these problems?

`--parents` is copying the directory structure, so you should get rid of that.

The way you've written this, the find executes, and the output is put onto the command line such that cp can't distinguish between the

spaces separating the filenames, and the spaces within the filename. It's better to do something like

```
$ find . -name \*.xls -exec cp {} newDir \;
```

in which cp is executed for each filename that find finds, and passed the filename correctly. Here's more info on this technique.

Instead of all the above, you could use zsh and simply type

```
$ cp **/*.xls target_directory
```

zsh can expand wildcards to include subdirectories and makes this sort of thing very easy.

Bash 4.0+ and ksh93 also supports **. For bash, use shopt -s globstar to enable it. For ksh, it's set -G or set -o globstar

That exec is technically less efficient than passing into xargs, which will do it all in as few cp calls as possible:

```
find . -name '*.xls' -print0 | xargs -0 cp -t destdir
```

From all of the above, I came up with this version. This version also works for me in the mac recovery terminal.

```
find ./ -name '*.xsl' -exec cp -prv '{}' '/path/to/targetDir/' ';'
```

It will look in the current directory and recursively in all of the sub directories for files with the xsl extension. It will copy them all to the target directory.

cp flags are:

- p - preserve attributes of the file
- r - recursive
- v - verbose (shows you whats being copied)

I had a similar problem. I solved it using:

```
find dir_name '*.mp3' -exec cp -vuni '{}' "../dest_dir" ";"
```

The '{}' and “;” executes the copy on each file.

I also had to do this myself. I did it via the –parents argument for cp:

```
find SOURCEPATH -name filename*.txt -exec cp --parents {} DESTPATH \;
```

you may remove the –parents but there is a risk of collision if multiple files bear the same name.

```
find [SOURCEPATH] -type f -name '[PATTERN]' |
    while read P; do cp --parents "$P" [DEST]; done
```

In 2022 the zsh solution also works in Linux Bash:

```
cp **/*.extension /dest/dir
```

works as expected.

Find files newer than...

Thanks to liberanet - pablos - #kde 

> I use the following technique to find files: 1) cd 2) touch than_me 3) find . -newer than_me 4) “do the thing that needs doing” 5) repeat 3 and look for affected files.

```
cd && touch than_me
```

do whatever has to be done

```
find . -newer than_me
```

~~DISCUSSION~~