# Linux find files while excluding / ignoring some files - Show all files on UNIX excluding hidden . (dot) files

**Author :** admin

A colleague of mine (Vasil) asked me today, how he can **recursively chmod to all files in a directory while exclude unreadable files for chmod** (returning ***permission denied***). He was supposed to fix a small script which was supposed to change permissions like :

```
chmod 777 ./
chmod: cannot access `./directory': Permission denied
chmod: cannot access `./directory/file': Permission denied
chmod: cannot access `./directory/onenote': Permission denied
```

First thing that came to my mind was to loop over it with for loop and grep out only /directory/ and files returning permissioned denied.

```
for i in $(find . -print | grep -v 'permission denied'); do echo chmod 777 $i; done
```

This works but if chmod has to be done to few million of files, this could be a real resource / cpu eater.

The better way to do it is by only using Linux ***find* command native syntax to omit files**.

```
find . -type f ( -iname "*" ! -iname "onenote" ! -iname "file" )
```

Above find will print all files in **.** - current directory from where find is started, except files: **onenote** and **file**.
To exclude

## Search and show all files in Linux / UNIX except hidden . (dot) files

Another thing he wanted to do is **ignore printing of hidden . (dot) files like .bashrc, .profile and .bash_history while searching for files** - there are *plenty of annoying .* files.*

To **ignore printing with find all filesystem hidden files from directory**:

```
find . -type f ( -iname "*" ! -iname ".*" )
```

on **web hosting webservers most common files which is required to be omitted on file searches is *.htaccess***

```
find . -type f ( -iname "*" ! -iname ".htaccess" )
```

In order to **print only all hidden files in directory except .bashrc and .bash_profile**:

```
find . -type f ( -iname '.*' ! -iname '.bashrc' ! -iname '.bash_profile' )
```

Another **useful Linux find use** for scripting purposes is listing only all files presented in current directory (simulating **ls**), in case if you wonder *why on earth to use find and not a regular ls command?*, this is useful for scripts which has to walk through millions of files (for reference see how to delete million of files in same folder with Linux find):

```
find . ! -name . -prune
```

```
./packages
./bin
./package
```

"***! -name .*** " - *means any file other than current directory*

***prune*** - prunes all the directories other than the current directory.

A more readable way to list only files in current folder with find is - identical to what above cmd:

```
find ./* -prune
```

```
 ./packages
./bin
./mnt
```

If you want to exclude **/mnt** folder and its sub-directories and files with find by using *prune* option:

```
find . -name tmp -prune -o -print
```