# Help Topic: Creating and Removing Directories

**Spring 2022**

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

---

## Creating/Removing Directories and Files on the Command Line

Now that we know how to see what directory we're in with `pwd`, and see the contents of the directory with `ls` and change between directories with `cd`, we discuss how to make and remove contents of directory. A directory has two types of elements: other directories, and files. We begin first with directories.

### Making a New Directory with the `mkdir` Command

Making a new directory is done with the `mkdir` command. It typically has only a single argument, the name of the new directory. For example, suppose your current directory has the following contents:

```
$ ls -F
mail/  public_html/   report_a.pdf   report_b.pdf   test.txt
```

You can make a new directory, `project_fango`, using `mkdir`:

```
$ mkdir project_fango
$ ls -F
mail/  project_fango/   public_html/   report_a.pdf   report_b.pdf   test.txt
```

That's it. In our example, we were in the directory where we created the new directory, but that's not necessary. You can provide a full path name of the directory you wish to create. For example:

```
$ mkdir ~/project_fango
```

will create the new directory in your home directory regardless of where you are in your file system. If the directory already exists, it will tell you so:

```
$ mkdir ~/project_fango
$ mkdir ~/project_fango
mkdir: project_fango: File exits
```

Lastly, to make a directory, you need write permissions in the directory where you're creating the

new directory. By default you always have write permission in your home directory and in the new directories you make. But eventually you may encounter the issue of permissions.

### Removing a Directory with the `rmdir` Command

A directory may be removed with the `rmdir` command:

```
$ rmdir project_fango
```

To use this command the directory must be empty. If you really wish to remove a directory containing a whole tree of other files and directories without having to crawl through the tree, you can do this with the `rm -rf` command discussed below. Be careful, this command wields great power.

### Making a File with the `touch` Command

There are many ways to make a new file; with an editor, as output from a program and so on. Assuming no knowledge yet of these things, the easiest way to create an empty file is with the `touch` command

```
$ touch file1 file2 file3
$ ls
file1 file2 file3
```

The above action creates three empty files. To check that they are empty it may be a good time to introduce the `-l` (for *long* format) option of the `ls` command:

```
$ touch file1 file2 file3
$ ls -l
-rw-r--r--   1 john  staff     0B Nov  3 09:51 file1.txt
-rw-r--r--   1 john  staff     0B Nov  3 09:51 file2.txt
-rw-r--r--   1 john  staff     0B Nov  3 09:51 file3.txt
```

The fifth column confirms that each new file created with `touch` has a size of zero bytes. The important thing is now we know a simple way to make a file to demonstrate the next essential command, `rm` for removing a file.

### Removing a File with the `rm` Command

Existing files may be removed with the `rm` command, which accepts one or more file names as arguments:

```
$ ls
file1 file2 file3
$ rm file3
$ ls
file1 file2
```

A note of caution. The `rm` command can be brutally powerful. It does not, by default, ask you if you're sure about what you've just requested to be removed. I highly recommend you change this default behavior. The `rm` command has a command line option `-i` that changes `rm` to ask for confirmation:

```
$ ls
file1 file2 file3
$ rm -i file3
remove file3?
```

To see how to make this interactive mode the default mode for the `rm` command, see the following help topic:

http://oceanai.mit.edu/ivpman/help/cmdline_interactive_rm_mv_cp

For now, just be really careful using the `rm` command.

### Removing Groups of Files with the `rm` Command

The command line offers a generic ability for matching groups of files or directories by naming a pattern. To remove all three of the files in the example above, one could instead just type:

```
$ ls
file1 file2 file3
$ rm file*
$ ls
$
```

The asterisk pattern matches any combination of characters against all file and directory names in the current directory. You could also remove, for example, all PDF files with `rm *.pdf`. Or any PDF file in any subdirectory with `rm */*.pdf`. Obviously this should be used with caution.

### Removing Directories with the `rm` Command

It was discussed above that a directory may be removed with the `rmdir` command. This only works when the directory being removed is empty. To remove an entire directory, including all its subdirectories, you can use two additional command line arguments:

```
$ rm -rf project
```

The `-r` options stands for *recursive* and will recursively remove the entire tree in the named directory.

The `-f` option stands for *force* and overrides any `-i` option, meaning no prompt for confirmation will be presented after the command is invoked. Extremely powerful, extremely dangerous. Use with caution.

As with `mkdir`, the `rm` command will not let you remove files or directories for which you do have write permisions. But you should certainly pause and think for a second whenever you begin a shell command with `rm -rf`.