problem	solution(s)	notes
you deleted a folder and recreated another folder with the exact same name, but you're in the old folder	- `cd $PWD` will cd to the new version of the folder (or `cd ../foldername`) - `cd .` too	
you accidentally created a folder with literally 1 million files in it and now you can't list the folder, it takes like 2 minutes	- use find | xargs or find -exec - try `/bin/ls -f -1` (unsorted, single column)	
even before you hit millions of files in a directory, you can easily have too many files for the maximum argument length to an executable	- use find | xargs or find -exec	
navigate to a folder on an external volume, then that external volume goes away (eg the disk is unplugged)		not sure what was meant by this tbh
write to a mount point that isn't actually mounted at that time, then fail to find the apparently vanished file later (because it's been mounted over)	- umount it to find the file - (on Linux) get the file with a bind mount: `mkdir other; sudo mount -o bind . other; ls -l other/mountpoint`	

how '..' works when you are in a directory which you got to via a symlink: `cd ..` gives you a different directory than `ls ..`	- `set -o physical` in bash will make them match
	- `pwd -P` will resolve the path to the current dir (though this isn't really a solution)
	- the equivalent of `set -o physical` in fish is:
	
	functions --copy cd _fish_cd
	function cd
	  if test "$argv" = "-"
	  _fish_cd -
	  else
	  _fish_cd (realpath $argv)
	  end
	end
		repro instructions:
		
		cd ~
		mkdir -p foo/bar/baz foo/bar/quux
		ln -s foo/bar/baz .
		cd baz # follows the symlink
		echo $PWD # outputs ~/baz
		ls .. # shows the contents of ~/foo/bar, not ~
		cd ..
		echo $PWD # outputs ~, not ~/foo/bar

in bash: when I'm in a folder via a symlinked path and I try to tab-complete a .../sibling , it uses the real path during tab-completion and the symlink one when evaluating it. So you can end up in a situation where a path you tab completed doesn't exist		not sure how to reproduce, would love ideas
if you follow a directory symlink, you may not be able to backtrack	- `cd -` (uses $OLDPWD in bash/zsh)	not sure how to reproduce, would love ideas
ls' is aliased to something with colour or other annotations that's really slow over a network share because it stat()s every file	- in bash/zsh: `\ls` will bypass the alias - `command ls` will bypass the alias	
what `mv file.txt dest` does is totally different depending on whether `dest` is a file or a directory (rename vs move to another folder)	- you could try to get in the habit of typing `dest/` when you mean a directory, but this can backfire (see #14) - for GNU mv: `mv -t dest source` will force dest to be a folder, `mv -T source dest` will force dest to be a file	
If you are using a FUSE filesystem and it hangs, you can end up with a black hole directory where any programs that touch it get stuck in "uninterruptable IO wait" which creates a spreading contagion that can make your system unusable until you reboot.	- kill -9 the fuse process, then unmount	
`ls dir*` will list the contents of the directories, especially annoying if you just want to check where a symlink points to	- use `ls -d`	
some things act differently depending on whether there's a `/` at the end, which can be confusing, for example: - on Mac OS, `cp -R a/ b` will copy the *contents* of a to b, different from `cp -R a b` - rsync	- rsync has a dry run option	