Launcher is the fundamental terminal user interface (TUI) which allows the user to run applications.
The TUI supports a few commands:

- `cd` (Unix)
- `pwd` (Unix)
- `ls` (Unix)
- `rm` (Unix)
- `rmdir`
- `cp` (Unix)
- `mv` (Unix)
- `mkdir` (Unix)
- `run`
- `setperms`
- `clearperms`
- `disable`
- `showperms`

Running programs

Users run programs with the `run` command.
The usage of `run` is `run [executable name] [permissions]`. `[executable name]` must be the name of an executable in the current working directory. `[permissions]` is a list of absolute paths, separated by commas, each ending with `(x)` where x is a two-bit binary number. For example:

```
run browser.exe
home.dir/browser.dir/tab1.txt(11),home.dir/browser.dir/tab2.txt(01)
```

is a valid call.

Program permissions

This is perhaps the most important and unique feature of MC. The user is able to set programs' permissions to directories.

There are 4 possible permissions a program has over a directory. By default, programs have `(00)` permission.

For files:
`(wr)`

- `w` is 1 iff the exe has write access to the file.
- `r` is 1 iff the exe has read access to the file.

So:

- `(00)` means the exe cannot read the file and cannot write to the file.
- `(01)` means the exe can read the file but not not write to the file.
- `(10)` means the exe can write to the file but cannot read the file (this should never be used).
- `(11)` means the exe can both read the file and write to the file.

For directories:
`(wr)`

- `w` is 1 iff the exe has write access to all files within the directory, can read the names of the contents, delete files, create files, rename files. The exe has `w=1` permission to every file and subdirectory within the directory.
- `r` is 1 iff the exe has read access to all files within the directory, can read the names of contents. The exe has `r=1` permission to every file and subdirectory within the directory.

So:

- `(00)` means the exe cannot read the names of the contents, cannot read any files within it, and cannot affect any of the files within it.
- `(01)` means the exe can read the names of the contents, can read the names of the contents of all subdirectories, and has `(01)` permissions over all files within it.
- `(10)` means the exe can read the names of the contents, can delete any files within it, can create new files within it, and rename files within it, and has `(10)` permissions over all files within it (this should never be used).
- `(11)` means the exe can do whatever it wants with any contents within the directory.

Exes have full permissions over files/directories they create.

If `a.exe` sets `(00)` as permission over `d.dir`, and then sets `(01)` over `d.dir/hello.txt`, then `(00)` applies to all contents within `d.dir` except for `d.dir/hello.txt` which has `(01)` (this rule should really never be used). When a file is moved to, copied to, or created in a directory, the file just inherits the permissions of its parent.

To set permissions:
`setperm a.exe home.dir/d.dir(01),home.dir/misc.dir(11)`

To clear perms:
`clearperms a.exe`
(makes all file permissions `(00)`)

Formal definition of file system tree:

Assume we have a file system tree, and an executable `e.exe`, and we have the command:
`setperms e.exe path1(x1),path2(x2),path3(x3),...,pathN(xN)`

This command is executed as follows:
We start by clearing all permissions.

- First we set the permission `x1` for `path1`. If `path1` is a directory, then `x1` is applied to all files within `path1`, even if `path1`'s parent is different from `x1`. If a file is moved/copied into `path1` then they inherit the permissions of its parent.
- Then, ***after we update the permissions for the previous*** `path1`, we set the permission `x2` for `path2`. If `path2` is a directory, then `x2` is applied to all files within `path2`, even if `path2`'s parent is different from `x2`. If a file is moved/copied into `path2` then they inherit the permissions of its parent.
- And so on for all paths `pathN`.