U5ZAJ15P0 : When I use lein, where will the clojure stdlib be located on my file system?
U5ZAJ15P0 : .clj files that is
U11BV7MTK : ```(let [nav [:a :b]]
```
  (reduce (fn [m [k v]] (assoc-in m (conj nav k) v))
          {:a {:b {}}}
          [[:b/field0 ""]
           [:b/field1 ""]
           [:b/field2 ""]]))
{:a {:b {:b/field0 "", :b/field1 "", :b/field2 ""}}}
```

U06CM8C3V : <@U5ZAJ15P0> You mean the jar files?
U5ZAJ15P0 : ah right, so the std lib will be within jar files. Nevermind then
U050MP39D : jar files are just zip files and you can unzip them and the .clj files will be inside, if that matters to you
U0NCTKEV8 : <@U5ZAJ15P0> lein runs 2 jvms, lein's jvm and your project's jvm, the clojure runtime that lein uses is part of the lein jar and lives in ~/.lein, the clojure runtime for your project is Just Another Jar(tm)
U06CM8C3V : lots of interesting jar files in `~/.m2/repository` too
U066TMAKS : what's a good way to xor two sets?
U050MP39D : (set/union (set/difference s1 s2) (set/difference s2 s1))
U051SS2EU : a silly way to do it ```=&gt; (into #{} (comp cat (take 2)) (clojure.data/diff #{1 2 3} #{2 3 4}))#{1 4}```

U0CDMAKD0 : is it possible to use clojure.java.shell to invoke a shell command like `tr &lt; infile.txt -d '\000' &gt; outfile.txt` ?
U0CDMAKD0 : <https://github.com/clojure/clojure/blob/master/src/clj/clojure/java/shell.clj>
U050ECB92 : <https://twitter.com/clojure_conj/status/889899202767147008>
U050ECB92 : ^ How cool is that?
U11BV7MTK : yeah got the email earlier. super super cool
U051SS2EU : <@U0CDMAKD0> in order to use things like `&lt;` you need to invoke /bin/sh
U051SS2EU : clojure.java.shell/sh, despite the name, is a raw system command that doesn't use sh
U051SS2EU : of course that's not portable, and then you've created code that only works on a *nix system or reasonable facsimile
U0CDMAKD0 : yeah, <@U051SS2EU> there is a thorny translation that `tr` handles well at the command line that I'd like to shell out within a clojure program (that does a whole laundry list of things). Haven't yet found the right syntax for calling `tr` via clojure.java.shell/sh though
U051SS2EU : <@U0CDMAKD0> you send a normal shell command to sh `(clojure.java.shell/sh "/bin/sh" "-c" "tr &lt; infile.txt -d '\000' &gt; outfile.txt")`
U0CDMAKD0 : wow, thank-you. That works. I tried a lot of other combinations but without the "/bin/sh" "-c"
U051SS2EU : right, sh -c says "find and run the sh executable, tell it to run this"
U0CDMAKD0 : that class of example should probably be added to the other examples (<https://github.com/clojure/clojure/blob/master/src/clj/clojure/java/shell.clj#L130-L142>)
U0CDMAKD0 : thanks again
U051SS2EU : np, glad I could help, I wonder if it would be worth submitting a patch to JIRA for adding another println to that comment block
U0CDMAKD0 : agreed - I think that would help others. Not sure how easy that is to do...
U051SS2EU : <@U0CDMAKD0> there is an example of using sh -c as an arg to sh on the clojuredocs page <https://clojuredocs.org/clojure.java.shell/sh>
U0CDMAKD0 : true enough, perhaps a few words there describing why it is useful/needed would work too
U6DJH8TCL : <@U051SS2EU> thanks for your help yesterday, you were correct !!!!
U17DY48BW : I know this is a clojure channel but maybe someone will know the answer. I'm trying to ssh into a server, run a shell script, and the disconnect from the server and have the script continue to run
U0954HGDQ : ssh + nohup?
U3JURM9B6 : <#C03RZGPG3|off-topic> :slightly_smiling_face:
U4PUTN69G : tmux or screen
U06GS6P1N : <@U5ZAJ15P0> 'what makes a good REPL?' - what value do REPLs bring, what features make for good REPL, what programming language features enable them
U06GS6P1N : <@U064X3EF3> a bit too long :) I need a 2-minutes thing
U068SUJNT : Thanks, that helped me
U5ZAJ15P0 : Hello! I am reading clojure.core's source cdoe and the beginning confuses me. Could someone explain this out?```
;during bootstrap we don't have destructuring let, loop or fn, will redefine later

```
(def
  ^{:macro true
    :added "1.0"}
  let (fn* let [&amp;form &amp;env &amp; decl] (cons 'let* decl)))
```

U5ZAJ15P0 : there are a number of those declarations
U5ZAJ15P0 : <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj#L31>
U61HA86AG : `fn*` is a compiler internal thing: <https://github.com/clojure/clojure/blob/master/src/jvm/clojure/lang/Compiler.java#L47>
U5ZAJ15P0 : <@U61HA86AG> what is the difference with `fn`? Also, why is `let` defined as a macro? I thought it was a special form
U61HA86AG : `fn` is defined in clojure, and just calls out to `fn*`: <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj#L42>
U61HA86AG : likewise, `let` is a macro that calls out to the special form `let*`
U5ZAJ15P0 : <@U61HA86AG> ah, I see. So special forms are always followed by a `*`, but for easy of use there is a non-starred version
U5ZAJ15P0 : Follow up question: what is the use of adding a name to an anonymous function?
U5ZAJ15P0 : e.g. `(fn foo [x] ...)`