VMware Workstation Pro 17.5.5 Build 37701024 With Keys Download [PATCHED] Pc

Q: How do I know if a bash command is encoded? I have a bash script that runs a program in a loop, but the program only accepts input if it's encoded in UTF-8. The problem is that I can't make bash automatically detect the encoding, so I have to use a technique like this: if ["`id -a -t $1`" = "8"] then if ["$2"!= ""] then eval `sudo $2` else echo $1 fi else echo "You typed an unencoded version of $1" fi (Specifically, I'm trying to detect if the name of the script and the input to it are both encoded in UTF-8, and if not, print a warning message.) The problem with this is that whenever the length of the $2 parameter is greater than 1, the file is unencoded. For example, if I run this: ./file_that_doesnt_know_utf8.sh "hello" It prints the following: file_that_doesnt_know_utf8.sh: You typed an unencoded version of 'hello' But I actually want it to print: file_that_doesnt_know_utf8.sh: You typed an unencoded version of 'hello' Note that it's not a typo. It's actually the correct output, but I want to be able to automatically detect when the input parameter is of size greater than 1. How can I get this to work without using eval in the code? A: You can do something like this: for c in hello world do echo "wrd". $c if (echo -n "$c" | iconv -t utf-8 -c -o /dev/null) >/dev/null 2>&1 then echo "$c" else echo "Encoding error in $c" fi done If your output is always the same as the input I guess you can do