

Forum: [Hack Forums](#)

Thread: [how do I make this bash script fast?](#)

rjusticemi - September 14, 2025, 1:22 pm

The first script completes in less than a quarter of a second, and the CompilationOfManyBreaches directory contains more than 20GB of data.

The second script ran in 3 minutes and 35 seconds, and it actually didn't even complete; I had to close the script manually because it was taking too long.

Why is the first script so fast, and how do I make the second script faster?

I can't have every single file in the letter format (/data/letter1/letter2) because I am planning to add more files with different names that aren't in that specific format. I want it to search through every single file and search for the specified email.

The first script was written by the person who I downloaded the data breaches from, but the second script was written by me.

I asked AI, but it was no help.

```
Code hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$ time bash query.sh
coolbrolin@yahoo.com
coolbrolin@yahoo.com:si0904
```

```
real 0m0.202s
user 0m0.186s
sys 0m0.016s
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$ time bash asdf.sh
coolbrolin@yahoo.com
^C
```

```
real 3m35.970s
user 0m54.626s
sys 0m8.193s
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$ # 3 minutes and 35 seconds
and no results at all
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$
```

```

hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$ cat query.sh # first script (fast
script)
#!/bin/bash
dir=$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )

if [ "$1" != "" ]; then
    letter1=$(echo ${1,,}|cut -b1)
    if [[ $letter1 == [a-zA-Z0-9] ]]; then
        if [ -f "$dir/data/$letter1" ]; then
            grep -ai "^$1" "$dir/data/$letter1"
        else
            letter2=$(echo ${1,,}|cut -b2)
            if [[ $letter2 == [a-zA-Z0-9] ]]; then
                if [ -f "$dir/data/$letter1/$letter2" ]; then
                    grep -ai "^$1" "$dir/data/$letter1/$letter2"
                else
                    letter3=$(echo ${1,,}|cut -b3)
                    if [[ $letter3 == [a-zA-Z0-9] ]]; then
                        if [ -f "$dir/data/$letter1/$letter2/$letter3" ]; then
                            grep -ai "^$1" "$dir/data/$letter1/$letter2/$letter3"
                        fi
                    else
                        if [ -f "$dir/data/$letter1/$letter2/symbols" ]; then
                            grep -ai "^$1" "$dir/data/$letter1/$letter2/symbols"
                        fi
                    fi
                fi
            fi
        else
            if [ -f "$dir/data/$letter1/symbols" ]; then
                grep -ai "^$1" "$dir/data/$letter1/symbols"
            fi
        fi
    fi
else
    echo "[*] Example: ./query name@domain.com"
fi
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$ cat asdf.sh # second script
(very slow script)
#!/bin/bash

dir=$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )

if [ "$1" != "" ]; then
    grep -ir "$1" .
else
    echo "[*] Example: ./query name@domain.com"
fi

```

```
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$  
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$  
hidden:/media/sh/Elements/Breaches/CompilationOfManyBreaches$ I want the second script to  
complete within 2 seconds max while also searching through files that aren't in the data/letter/letter  
format
```

DNA - *September 21, 2025, 10:25 pm*

Not sure what prompt you used to query the difference with your LLM (out of curiosity I tried with claude and it managed to spot it), but there are two main reasons:

1. The first script targets specific folders data/c/o/o based on the search term, so it only looks in one small file
2. The second script uses the recursive flag, your -r parameter after -i on the grep command which is killing performance, as it's recursively searching through every single folder and file in the directory

Basically the first script is smart and goes straight to where the data should be, while the second one brute forces through your entire breach collection. That's why you're getting 0.2 seconds vs 3+ minutes.