twocore.py – Read Me

A version of the multi-core program that uses only *two* processes. This program will alter how it splits up the space of all possible passwords among two processes, so that the workload is evenly balanced between the two processes.

crackfile.py – Read Me

crackfile.py attempts to crack a password database file using the output table files generated by the above programs.

This script takes one command-line argument that is the file name of a plaintext password database file. The format of this file is that each line contains a username, followed by a colon, followed by a hexadecimal SHA256 password hash. Here is an example:

```
bsmith:c9efc14482fc976da42c8f2af73551c31af35bcfac5267fbdca395b79a05d0fc
```

The file can have any number of lines of this form.

The script will load a rainbow table file from disk and creates a dictionary from it. Then, it will read the specified password database file from disk and attempts to find a password that matches each hash value contained in the file. The program should print out the (username, password) pair for any successfully cracked passwords.

The cracking program will not recompute any hashes! The point of a rainbow table is to precompute all hashes *once*, then use it as a database to crack quickly in the future. Also, this program will use dictionary lookup to find the matching hash (if any) quickly, rather than linearly searching through all the hashes. crackfile.py will finish searching for all five hashes within a few seconds.
The sample password database, “shadow.txt”, is used to crack.