**Crack**

Implemente un programa que crackee passwords, como se muestra a continuación.

$ ./crack 50fkUxYHbnXGw

rofl

[**Background**](http://docs.cs50.net/problems/crack/crack.html#background)

En la mayoría de los sistemas corriendo Linux estos días hay un archivo llamado /etc/shadow, que contiene usernames y passwords. Afortunadamente, los passwords contenidos ahí no son almacenados "en claro" sino que, en cambio son encriptados usando una " función hash de una sola vía o one-way hash " Cuando un usuario hace log en estos sistemas introduciendo un username y password, el último es encriptado con la misma función hash, y el resultado es comparado contra la entrada del username en /etc/shadow. Si los dos hashes concuerdan, el usuario es aceptado. Si alguna vez olvidas un password, quizás le hayan dicho que los técnicos de Soporte no pueden ver su password pero pueden cambiarlo para usted. Eso es probable porque los técnicos de soporte solo pueden ver en todo caso, un hash de su password, no su password en sí mismo. Pero podrán crear un nuevo hash para usted.

Aún cuando los passwords en /etc/shadow están como hashes, la función hash no siempre es tan fuerte. Muy a menudo adversarios, al obtener de alguna manera ese archive, son capaces de descifrar (y revisar) los passwords de los usuarios o de crackearlos utilizando fuerza bruta (ej., tratando todos los passwords posibles). Abajo tenemos como luciría /etc/shadow aunque simplificado, en dónde en cada línea está formateada como username:hash.

andi:50.jPgLzVirkc

jason:50YHuxoCN9Jkc

malan:50QvlJWn2qJGE

mzlatkova:50CPlMDLT06yY

patrick:50WUNAFdX/yjA

rbowden:50fkUxYHbnXGw

summer:50C6B0oz0HWzo

stelios:50nq4RV/NVU0I

wmartin:50vtwu4ujL.Dk

zamyla:50i2t3sOSAZtk

**[Especificación](http://docs.cs50.net/problems/crack/crack.html" \l "specification)**

Diseñe e implemente un programa, crack, que crackee passwords.

* Implemente su programa en un archivo llamado crack.c en un directorio llamado crack.
* Su programa deberá aceptar un solo argumento de línea de comando: un password hash.
* Si su programa es ejecutado sin ningún argumento de línea de comandos o con más de un argumento de línea de commandos, su programa deberá imprimir un mensaje de error (de su elección) y salir inmediatamente, con main retornando 1 (significando esto un error).
* De lo contrario su programa deberá proceder a crackear el password dado, idealmente lo más rápido posible, por ultimo imprimiendo el password descifrado seguido de \n, nada más, nada menos, con main retornando 0.
* Asuma que cada password ha sido hasheado con la función de C basado en DES (no basado en MD5) para encriptar.
* Asuma que cada password no es más lago que (ejemmm) cuatro caracteres
* Asuma que cada password está compuesto enteramente de caracteres alfabéticos (mayúsculas y/o minúsculas).

**[Walkthrough](http://docs.cs50.net/problems/crack/crack.html" \l "walkthrough)**

**[Us](http://docs.cs50.net/problems/crack/crack.html" \l "usage)o**

Su programa deberá comportarse como el ejemplo a continuación. Asumiendo que el texto subrayado es lo que un usuario ha introducido.

$ ./crack

Usage: ./crack hash

$ ./crack 50fkUxYHbnXGw

rofl

[**Testing**](http://docs.cs50.net/problems/crack/crack.html#testing)

¡No hay check50 para este! Pero las posibilidades son, de que si puede crackear todos los diez passwords anteriores, va por buen camino!

**[Consejos](http://docs.cs50.net/problems/crack/crack.html" \l "hints)**

Asegúrese de leer acerca de crypt, tomándo nota particularmente de las menciones de "salt":

man crypt

Según esa man page, usted querrá poner

#define \_XOPEN\_SOURCE

#include <unistd.h>

En la parte superior de su archive a fin de utilizar crypt. Además usted querrá enlazar con -lcrypt, al no compilar con make sino con:

clang -ggdb3 -O0 -std=c11 -Wall -Werror -Wshadow crack.c -lcrypt -lcs50 -lm -o crack

[**FAQs**](http://docs.cs50.net/problems/crack/crack.html#faqs)

*Ninguna hasta ahora! Recargue esta págin periódicamente para revisar si llega alguna!*