Programming Assignment 3

Due Tuesday, September 29 before midnight

NAME
crack – multithreaded brute-force password hash cracker

USAGE
crack threads keysize target

DESCRIPTION
crack should attempt to find the password associated to the target DES hash. It does this by trying all possible lowercase alphabetic (a-z) passwords of length up to keysize. The program should run with threads concurrent threads for speed.

Linux/Unix user passwords are never stored on the system. Instead, a function called a hash is applied to the password. Then, the hashed password is stored, traditionally in /etc/passwd or more recently in /etc/shadow.

The classic hash function on Unix systems is crypt(3). To make things harder on crackers, crypt also uses a two character string called a salt which it combines with the password to create the hash. Schematically:

password + salt $\rightarrow$ hash

The salt is visible in the hash as the first two characters. As an example, a password ‘apple’ and salt ‘na’ become the hash ‘na3C5487Wz4zw’.

The crack program should extract the salt from the first two characters of target, then repeatedly call crypt using all possible passwords built of up to keysize lowercase alphabetic characters.

When a match to target is found, the program should print the cracked password and exit immediately. If the entire space of passwords is searched with no match, the program should exit with no output.

HINTS
See /export/mathcs/home/public/clair/bin/crack for a working version. There is also a useful utility encode in the public os/demo_sec directory. encode will encode passwords and salt into hashes. You can also encode using the one line perl command: perl -e 'print crypt("apple","na")'

Don’t forget to compile with the -lpthread and -lcrypt options.

The maximum allowed keysize is 8 (since crypt only uses the first 8 characters of the password anyway).
You might want to write this first as a single threaded program. Just remember that `crypt` won't work with multiple threads - you need to switch to `crypt_r`.

You need to check passwords of length `keysize`, but don’t forget you also need to check the shorter ones. The simplest way is probably to write a function that checks all passwords which are exactly a given length, and then have `main` call it in a loop from 1 to `keysize`.

For extra credit, add optional command line switches that allow searching for lowercase, uppercase, symbols, or combinations of them.

**USEFUL MAN PAGES**

- `pthread_create(3)`
- `pthread_join(3)`
- `pthread_exit(3)`
- `pthread_self(3)`
- `pthread_mutex_init(3)`
- `pthread_mutex_lock(3)`
- `pthread_mutex_unlock(3)`
- `crypt_r(3)`
- `strcmp(3)`
- `strcpy(3)`
- `strncpy(3)`
- `strncpy(3)`