

Assignments

Assignment 1 - In progress

Honor Pledge Accepted
Draft - In progress
Submitted
Returned

Assignment Details

Title
Assignment 1
Due
Jan 31, 2023 11:55 PM
Number of resubmissions allowed
Unlimited
Accept Resubmission Until
Jan 26, 2023 11:55 PM
Status
Honor Pledge Accepted
Grade Scale
Points (max 100.00)

Instructions

Assignment overview

You have been provided with a program called `hackme`. To successfully hack the program, you need to provide the correct password. A random password would be impossible to guess. Thankfully, you have a few clues to discover the password:

- You know the password is definitely 12 characters in length
- The password could contain any character in the range 33 - 126 in [ASCII](#)
- You have a function called `checkPassword` which can be used to discover the password. Unfortunately, it can only check 3 consecutive characters at a time. You supply a three character guess (`const char * password`) and the position in the password you want to check (`int start`). The function will return 0 if the guess is correct, -1 if it is not correct.

To discover the password, you will have to use brute-force to check every possible 12 character combination. You will do this by checking three characters at a time. Characters from position 0-2, position 3-5, position 6-8, and position 9-11. When your program finds a match, it should print the match to the screen.

To make the discovery task faster, you will divide the task between 4 processes and you will demonstrate that forking accomplishes the task quicker.

Purpose

The goals of this assignment are the following:

- Learn about process creation and control in Linux environment
- Get experience with the `fork()`, `execl()`, `getpid()`, `getppid()` and `wait()` system functions
- Gain more experience with the C programming language from an OS perspective

Computing platform

You are welcome to develop your program on your own workstation if you wish, but you are responsible for ensuring that your program compiles and runs without error on the Gaul computing platform. Marks will be deducted if your program fails to compile, or your program runs into errors on Gaul.

- <https://wiki.sci.uwo.ca/sts/computer-science/gaul>

Instructions

Attached to this assignment is a tarball with the following files in it. **None of these files should be modified:**

```
checkPassword.h  <--- A header file containing the prototype for checkPassword()
checkPassword.o  <--- The object file for checkPassword(). Make requires this
hackme           <--- The program you need to hack!
Makefile         <--- A pre-packaged Makefile. This tells you how your program should be structured
run-assignment.sh <--- A shell script that will automatically run your program
```

Download this tarball and upload it to Gaul. Extract the tarball (`tar -xvf Assignment-1.tar`). Change to the `Assignment-1` directory.

You will write a program called `assignment-1.c`. This program:

- Can accept an optional command-line argument `-f` to signify that forking should be enabled or not.
- Can accept an optional command-line argument `-p` to signify that the parent process should automatically run `hackme` using `exec1`
- If forking is not requested, it will check characters 0-2, 3-5, 6-8, and 9-11 in that order.
- If forking is requested
 - The parent (1.0) will check characters 0-2, then wait on child 1 and child 2 to finish
 - Child 1 (1.2) will check characters 3-5
 - Child 2 (1.1) will check characters 6-8, then wait on Child 3 to finish
 - Child 3 (1.1.1) will check characters 9-11
 - Your program should print PIDs of all parent and child processes
- If running `hackme` automatically is not requested, simply return 0
- if running `hackme` automatically is requested, wait for all child processes if necessary, then run `hackme`
- When three characters of the password are found, just print them to the screen. In the case of forking, this means the password will probably be printed out of order. This is okay because some processes will find their part of the password faster than others. The point is speed. The order you can figure out manually.

Output

Assuming the password was `abcdef123456`, Executing `./assignment-1` should produce the following output:

```
abc
def
123
456
```

Executing `./assignment-1 -f -p` should produce the following output (356... is the pid using `getpid()` and `getppid()`):

```
PID 1.0 IS 356930. CHILD 1.1 IS 356932
PID 1.1 IS 356932. PPID IS 1.0 356930
PID 1.0 IS 356930. CHILD 1.2 IS 356933
PID 1.1 IS 356932. CHILD 1.1.1 IS 356934
PID 1.1.1 IS 356934. PPID 1.1 IS 356932
PID 1.2 IS 356933. PPID 1.0 IS 356930
def
abc
123
456
Please enter a password: abcdef123456
ACCESS GRANTED!
```

and `./run-assignment.sh 1` should produce the following output:

```
ASSIGNMENT 1 STARTED - Dow Mon  ## ##:##:## AM/PM EST 2023
```

Cleaning environment

```
-----
```

```
rm -f assignment-1.o assignment-1
```

Checking environment

```
-----
```

```
748327eb2da0d4371368f72bd42583b0  ./run-assignment.sh
```

```
*****UNIQUE ID***** assignment-1.c
```

```
checkPassword.o: OK
```

```
hackme: OK
```

```
Makefile: OK
```

```
checkPassword.h: OK
```

Building environment

```
-----
```

```
make all
```

```
make[1]: Entering directory '/home/wbeldman/3305/Projects/Assignment 1/Assignment-1'
```

```
gcc -c assignment-1.c -Wall -Wpedantic -Wextra -std=gnu17
```

```
gcc -o assignment-1 checkPassword.o assignment-1.o -Wall -Wpedantic -Wextra -std=gnu17
```

```
make[1]: Leaving directory '/home/wbeldman/3305/Projects/Assignment 1/Assignment-1'
```

Assignment 1 (without forking)

assignment-1,5032

abc

def

123

456

104 Seconds

Assignment 1 (with forking)

PID 1.0 IS 356930. CHILD 1.1 IS 356932

PID 1.1 IS 356932. PPID IS 1.0 356930

PID 1.0 IS 356930. CHILD 1.2 IS 356933

PID 1.1 IS 356932. CHILD 1.1.1 IS 356934

PID 1.1.1 IS 356934. PPID 1.1 IS 356932

PID 1.2 IS 356933. PPID 1.0 IS 356930

assignment-1,356930 -f

└─assignment-1,356932 -f

| └─assignment-1,356934 -f

└─assignment-1,356933 -f

def

456

123

abc

43 Seconds

Enter your guess: abcdef123456

Trying: abcdef123456

Please enter a password: ACCESS GRANTED!

Cleaning environment

rm -f assignment-1.o assignment-1

ASSIGNMENT 1 COMPLETED - Dow Mon ## #:##:## AM/PM EST 2023

Helpful hints

- If you cannot remember how to read command-line arguments in C, see https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm
- To use the checkPassword function
 1. make sure you #include "checkPassword.h" in your source file
 2. make sure you include the checkPassword.o object file when you compile. This can be done one of two ways:
 1. gcc checkPassword.o assignment-1.c
 2. Use the Makefile by issuing the command make
 1. The make command will read the Makefile and execute the rule it finds first (make default) which will in turn run the all rule

Submitting

When you are finished your assignment, follow these steps


1. From inside the Assignment-1 directory, run the following command: `script -c 'run-assignment.sh 1' assignment-1.out`
Your directory should now contain the following files:

```
assignment-1.c    <--- Your program
assignment-1.out  <--- The output produced by running the script command above
checkPassword.h   <--- A header file containing the prototype for checkPassword()
checkPassword.o   <--- The object file for checkPassword(). Make requires this
hackme            <--- The program you need to hack!
Makefile          <--- A pre-packaged Makefile. This tells you how your program should be structured
run-assignment.sh <--- A shell script that will automatically run your program and put the results in assignment-1.out
```

2. Assuming the command was successful, run the follow command to get out of the Assignment-1 directory: `cd ..`
3. Package your assignment into a tarball: `tar -cvf Assignment-1.tar Assignment-1`

4. Verify the contents of your tarball (`tar -tvf Assignment-1.tar`) (`du -sh Assignment-1.tar`). **If your tarball is 10kb in size** you have an empty tarball and you made an error on this step. Make sure you are properly creating your tarball with the right files in it.
5. Use an SFTP program to download the tarball and then upload it to OWL.

Additional resources for assignment

-  [Assignment-1.tar](#) (40 KB; Jan 2, 2023 1:25 pm)

Grading Rubric

[Preview Rubric](#)

Submission

Attachments

No attachments yet

Select a file from computer [Choose File](#) No file chosen

[Proceed](#)

[Preview](#)

[Save Draft](#)

[Cancel](#)



Don't forget to save or proceed!