SCALE FOR PROJECT NM-OTOOL (/PROJECTS/NM-OTOOL)

Introduction

Please respect the following rules:

- Remain polite, courteous, respectful and constructive
throughout the correction process. The well-being of the community
depends on it.

- Identify with the person (or the group) graded the eventual
dysfunctions of the work. Take the time to discuss
and debate the problems you have identified.

- You must consider that there might be some difference in how your
peers might have understood the project's instructions and the
scope of its functionalities. Always keep an open mind and grade
him/her as honestly as possible. The pedagogy is valid only and
only if peer-evaluation is conducted seriously.

Guidelines

- Only grade the work that is in the student or group's
GiT repository.

- Double-check that the GiT repository belongs to the student
or the group. Ensure that the work is for the relevant project
and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases was used to fool you
and make you evaluate something other than the content of the
official repository.

- To avoid any surprises, carefully check that both the correcting
and the corrected students have reviewed the possible scripts used
to facilitate the grading.

- If the correcting student has not completed that particular
project yet, it is mandatory for this student to read the
entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository,
non-functioning program, a norm error, cheating etc. In these cases,
the grading is over and the final grade is 0 (or -42 in case of
cheating). However, with the exception of cheating, you are
encouraged to continue to discuss your work (even if you have not
finished it) in order to identify any issues that may have caused
this failure and avoid repeating the same mistake in the future.

- For this project specifically, will not be eliminatory:
- Nm: An output formating or information order that differs from the
system binary.
- Otool: An output formating that differs from the system binary.

- For this project specifically, will be eliminatory:
- Output missing information from the system binary.

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/5930/ft_nm_otool.en.pdf)

Preliminaries

To check out the submitted project you can use the following code: $> cat simple_test.c.c #include <stdio.h> int main(void) { puts("Simple test"); return (0); } $> cat less_simple_test.c #include <stdio.h> int global_var = 40; int main(void) { printf("The global variable is: %d\n", global_var); return (0); } $> cc simple_test.c -o simple_test $> cc less_simple_test.c -o less_simple_test $> cc -m32 less_simple_test.c -o less_simple_test_32-bit $> If you are not sure, to identify a binary (type/architecture): man 1 file If in trouble to locate an universal binary research in the PATH: (IFS=$'\n'; for d in ${PATH//:/$IFS}; do find "$d" -type f -exec file '{}' \+ | grep -i -A3 universal ; done) To create an universal binary: clang/gcc: -m32 to cross-compile explicitly in 32-bit. lipo -create -output <universel> <binaire arch. 1> <binaire arch. 2> ... In case of difficulties to locate a dynamic library: (.so, .dylib): find /usr/lib -type f -iname '\.dylib' 2>/dev/null*

Preliminary tests

Check firstly the following elements:

- There is something in the git repository.
- A valid author file
- The Makefile is present and compiles correctly the executables
ft_nm et ft_otool

- No norm errors, Norminette is authoritative.
- No cheating (All functions are authorised, the student can explain the code, ...)

If an element of this list isn't respected, the grading ends.
Use the appropriate flag. You're allowed to debate some more about the project, but the grading will not be applied.

✓ Yes ✗ No

Functionality tests for nm

Simple test

Test out ft_nm on the simple test's binary.
The output matches the nm in reference.

✓ Yes ✗ No

Less simple test

Test out ft_nm on the less simple test's binary.
On both the 32 and 64-bit binaries, the ft_nm output is always identical to the nm in reference, no matter the test done (no output with diff).

✓ Yes ✗ No

Format and order

The ft_nm output is always identical to the nm in reference, no matter which test is done (no exit with diff).

✓ Yes ✗ No

Multiple arguments

ft_nm manages multiple arguments.

✓ Yes ✗ No

Objet files

Test out ft_nm on a 32 and 64-bits version of an object file (.o).
The output matches the nm in reference.

✓ Yes ✗ No

Dynamic librairies

Test out ft_nm on a dynamic library (*.dylib *.so) (see in /usr/lib/).
The exit matches the nm in reference.
The order of the symbols for identical names is arbitrary and can differ.

✓ Yes ✗ No

Universal binary

Test out ft_nm with an universal binary (exemple: /usr/bin/python).
Check that it is indeed universal with the command "file ".
The output matches the nm in reference.
The order of the symbols for identical names is arbitrary and can differ.

✓ Yes ✗ No

Functionality tests for otool

Simple test

Test out ft_otool on the simple test's binary.
The output matches the the one of otool -t.

✓ Yes ✗ No

Format and order

On both 32 and 64-bit binaries the ft_otool output is identical to
the one of otool -t, no matter the test done (no output with diff).

✓ Yes ✗ No

Multiple arguments

ft_otool manages multiple arguments.

✓ Yes ✗ No

Objet files

Test out ft_otool on a 32 and 64-bits version of an object file (.o).
The output matches the one of otool -t.

✓ Yes ✗ No

Dynamic librairies

Test out ft_otool on a dynamic library (*.dylib *.so) (see in /usr/lib/).
The exit matches the one of ft_otool -t.

✓ Yes ✗ No

Universal binary

Test out ft_otool with an universal binary (exemple: /usr/bin/python).
Check that it is indeed universal with the command "file ".
The output matches the one of ft_otool -t.

✓ Yes ✗ No

Bonus

Bonus

Count in this section the different bonuses. You can grade up to 5
distinctive bonuses.

Each bonus must be :
- At the very least useful (up to you)
- Well implemented and 100% functional

Rate it from 0 (failed) through 5 (excellent)

○

Ratings

Don't forget to check the flag corresponding to the defense

✓ Ok ★ Outstanding project

Empty work Incomplete work No author file Invalid compilation

Norme Cheat Crash Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!